Executive Summary

The Web is constantly evolving into an unprecedented and continuously growing source of knowledge, information and services, potentially accessed by anyone at anytime and from anywhere. Yet, the current uptake rates of the Web have not really reached their full potential, mainly due to the design of modern Web-based interfaces, which fail to satisfy the individual interaction needs of target users with different characteristics. A common practice in contemporary Web development is to deliver a single user interface design that meets the requirements of an "average" user. However, this "average" user is in fact an imaginary user. Often the profiles of a large portion of the population, especially that of people with disability, elderly people, novice users and users on the move, differs radically from that of the average user.

This work aims at supporting and providing the means for the development of inclusive Web-based interfaces that are capable to adapt to multiple and significantly different user profiles and contexts of use. To this purpose, the *Unified Web Interfaces* (UWIs) method is proposed as an alternative approach to the design and development of Web-based applications. This novel method builds on well-established *Design for All* principles and on the *Unified User Interfaces* methodology.

Furthermore, *EAGER*, an advanced toolkit, is provided as a means to facilitate Web developers in following the proposed UWI method. EAGER allows *Microsoft® .NET* developers to create interfaces that conform to W3C accessibility guidelines, and which are able to adapt to the interaction modalities, metaphors and user interface elements most appropriate to each individual user, according to profile information based on user and context specific parameters. As a means to validate the proposed toolkit, a prototype, fully functional Web portal was implemented by means of the EAGER toolkit and evaluated in terms of its accessibility and usability to all. Overall, this development confirmed that EAGER offers significant benefits to developers and ensures the delivery of accessible, usable and satisfying Web-based interfaces. The process of employing EAGER is significantly less demanding in terms of time, experience and skills required from the developer than the typical process of developing Web interfaces for the "average" user. Finally, this report discusses potential extensions of this work.

Acknowledgments

Part of the work reported here, especially the development of the new EDeAN portal, has been carried out in the framework of the European Commission funded Coordination Action DfA@eInclusion[1] (*Design for All for eInclusion*, Contract no. 0033838).

[1] http://www.dfaei.org/

Contents

Executive Summary ..... **i**

Acknowledgments ..... **iii**

Contents ..... **v**

List of Figures ..... **ix**

List of Tables ..... **xv**

CHAPTER 1 INTRODUCTION **1**

1.1 Computers and User Interfaces 1
1.2 About User Interface adaptation 2
1.3 Web-based User Interfaces: Opportunities and Challenges 3
1.4 Towards Unified Web-based Interfaces 4
1.5 Structure and contents of this report 5

CHAPTER 2 RELATED AND BACKGROUND WORK **7**

2.1 Web accessibility 7
2.1.1 Definition of accessibility *7*
2.1.2 Assistive technology and accessibility features in Operating Systems *8*
2.1.3 User agents (Web browsers) *10*
2.1.4 Web content accessibility - Evaluation tools *11*
2.1.5 Development environments and tools *13*
2.2 Web usability 13
2.2.1 Definition of usability *13*
2.2.2 Principles and heuristics *14*
2.3 Adaptation techniques for the Web 15
2.3.1 Alternative User Agents *15*
2.3.2 Intermediary Agents *16*
2.3.3 Self-adapting Web-based systems *18*

CHAPTER 3 METHODOLOGY **21**

3.1 User Interfaces for All 21
3.2 Unified User Interfaces 22
3.3 Proposed implementation approach 24

CHAPTER 4 TOWARDS UNIFIED WEB-BASED INTERFACES **27**

4.1 Adaptive and Adaptable behaviour 27
4.2 The Unified Web-based User Interfaces (UWI) methodology 28
4.2.1 User Information Component *29*
4.2.2 Context Information Component *31*
4.2.3 Decision Making Component *32*
4.2.4 Dialogue Controls Component *33*
4.2.5 Designs Repository *34*
4.3 Extensibility 36

CHAPTER 5 EAGER: A TOOLKIT FOR WEB DEVELOPERS **37**

5.1 User Information Component (UIC) implementation 37
5.1.1 User Profiles Repository *37*
5.1.2 User Components Repository *38*
5.1.3 Profiling Module *41*
5.1.4 Interaction Monitoring Module *42*
5.1.5 User Profile Statistics UI Module *42*
5.1.6 User Profile Selection UI Module *42*

5.1.7 User Profile Administration UI Module *43*
5.2 Context Information Component (CIC) implementation 43
5.2.1 User Context Profiles Repository *43*
5.2.2 Context Monitoring Module *44*
5.3 Decision Making Component (DMC) 44
5.4 Dialogue Controls Component (DCC) implementation 48
5.4.1 Dialogue controls hierarchy *50*
5.4.2 Examples of dialogue controls implementation *51*
5.4.2.1 Content adaptation 51
5.4.2.2 Layout adaptation 54
5.4.2.3 Navigation adaptation 57
5.4.2.4 Interaction adaptation 60
5.5 Designs Repository 64
5.5.1 Content-related alternative designs *64*
5.5.2 Layout-related alternative designs *68*
5.5.3 Navigation-related alternative designs *69*
5.5.4 Interaction-related alternative designs *72*

CHAPTER 6 EAGER VS. VISUAL STUDIO® ALONE 79

6.1 Initial development stages 79
6.2 Using Microsoft's .Net technologies alone 80
6.3 Using EAGER [in combination with .Net] 86
6.4 Porting EAGER into existing Web applications 90
6.5 Conclusions 91

CHAPTER 7 THE EDEAN CASE STUDY 93

7.1 HERMES 93
7.2 The new EDeAN Portal 94
7.2.1 Public area *94*
7.2.1.1 General settings 95
7.2.1.2 Applying different Accessibility & Interaction settings 96
7.2.2 Subscribed Area *102*
7.2.2.1 User Profiling (Usability – Accessibility settings) 104
7.2.2.2 Special Interest Groups 135
7.2.2.3 DfA Knowledge tools 153
7.2.3 Network Administrator's Interface *161*
7.2.3.1 Profile Statistics 162
7.2.3.2 Profile Administration 165

CHAPTER 8 EVALUATION - VALIDATION 167

8.1 Accessibility evaluation of UI elements 167
8.1.1 Procedure *167*
8.1.2 Results (Conformance to guidelines) *168*
8.1.3 Results (colour effectiveness) *175*
8.2 User-experience evaluation 177
8.2.1 Procedure *177*
8.2.1.1 System description 178
8.2.1.2 Inspection team 178
8.2.1.3 Target user groups 179
8.2.1.4 Functions per user group 179
8.2.2 Inspection results *180*

CHAPTER 9 CONCLUSIONS AND FUTURE WORK 185

REFERENCES 187

APPENDIX A: Development of alternative designs 191

Template alternatives styles – size *191*
Window alternatives *193*

Navigation alternatives *194*
Links - Buttons alternatives *199*
Image alternatives *201*
Textboxes alternatives *202*
Tabs alternatives *203*
Fieldset alternatives *204*
Tables alternatives *205*
Charts alternatives *207*
Module options alternatives *207*
Functions alternatives *209*
Paging representations - Single items representations *210*
Text entry alternatives *212*
Formatted text entry *215*
Date entry alternatives *215*
Upload files *218*
Upload images *220*
Files alternatives *223*
Images alternatives *224*
Search alternatives *226*

APPENDIX B: The EAGER toolkit (in detail) 229

Templates *230*
Header Templates *231*
Header Container Templates *231*
Header Templates *232*
Header Sub - Templates *234*
Footer Templates *236*
Windows *238*
Navigation *239*
Top navigation *240*
Bottom navigation *240*
Main navigation *241*
Full navigation *242*
Favourites navigation options *244*
Links *244*
Buttons *245*
Single selection controls *246*
Labels *246*
Multiple selection elements *247*
Images *247*
Tabs *248*
Field Sets *249*
Tables *250*
Charts *251*
Module options *254*
Functions *255*
Paging- Single items *256*
Text entry *258*
Formatted text entry *260*
Date entry *260*
Upload files *261*
Upload images *262*
File displayer *264*
Image displayer *264*
Search *265*

Appendices References i

List of Figures

Figure 1: **ISO framework for specifying and measuring the Usability of products** [15] ... 14
Figure 2: **The concept of User Interfaces for All** ... 22
Figure 3: **Overview of the U²I Architecture** [46] ... 23
Figure 4: **Proposed software architecture for implementing UWIs** ... 25
Figure 5: **Architecture of Unified Web-based Interfaces (UWIs)** ... 29
Figure 6: **User Information Component architectural model** ... 30
Figure 7: **Example of User Profile Instance** ... 30
Figure 8: **Context Information Component architectural model** ... 31
Figure 9: **Decision Making Component architectural model** ... 33
Figure 10: **Dialogue Controls Component architectural model** ... 33
Figure 11: **The polymorphic task hierarchy concept** ... 34
Figure 12: **Physical design alternatives (for *File uploader*)** ... 35
Figure 13: **Project EAGER – results of code line counter** ... 37
Figure 14: **User profile basic attributes specified prior to initiation of interaction** ... 38
Figure 15: **The Database diagram of the table to hold the basic user attributes** ... 38
Figure 16: **User profile extended attributes specified during interaction (1/3)** ... 39
Figure 17: **User profile extended attributes specified during interaction (2/3)** ... 39
Figure 18: **User profile extended attributes specified during interaction (3/3)** ... 39
Figure 19: **The database schema used for storing the extended user characteristics** ... 40
Figure 20: **Class diagram of *UserInformationComponent* and *UserFinalSettings*** ... 41
Figure 21: **The class diagram of the *InteractionMonitoringModule* class** ... 42
Figure 22: **User Context Profiles Repository Database scheme** ... 43
Figure 23: **Context Specific Attributes** ... 43
Figure 24: **The class diagram of the *ContextMonitoringModule* class** ... 44
Figure 25: **The class diagram of the *DecisionMakingComponent* class** ... 45
Figure 26: **The class diagram of the *DecisionLogic* class** ... 47
Figure 27: **The class diagram of the *DialogueControlComponent* class** ... 50
Figure 28: **The ics.Adaptation.DialogueControls namespace** ... 51
Figure 29: **Images hierarchy** ... 52
Figure 30: **Image displayer diagrams** ... 53
Figure 31: **The hierarchy of the potential template variations** ... 55
Figure 32: **Full navigation sub template control attributes** ... 58
Figure 33: **Categories navigation diagram** ... 59
Figure 34: **Sub categories navigation diagram** ... 59
Figure 35: **Option navigation diagram** ... 60
Figure 36: **File uploader diagram** ... 61
Figure 37: **Date entry diagram** ... 62
Figure 38: **Image alternative representations** ... 64
Figure 39: **Images representations** ... 65
Figure 40: **Display thumbnail, download link and estimated download time** ... 65
Figure 41: **Display full image info** ... 66
Figure 42: **Display thumbnail, download link and size** ... 66
Figure 43: **Display as slide show (version 1)** ... 67
Figure 44: **Display as a slide show (version 2)** ... 67
Figure 45: **Template representation styles** ... 68
Figure 46: **Template alternatives according to device resolution** ... 69

Figure 47: **Navigation alternatives** 70
Figure 48: **Navigation linearized (novice)** 70
Figure 49: **Navigation linearized (moderate)** 71
Figure 50: **Navigation linearized (expert)** 71
Figure 51: **Upload files alternatives** 72
Figure 52: **File upload (indirect manipulation)** 72
Figure 53: **File upload (direct manipulation)** 73
Figure 54**: File upload (mixed mode manipulation)** 74
Figure 55: **Date entry alternatives** 75
Figure 56: **Date entry (drop-downs, automated date enforcement)** 75
Figure 57: **Date entry (drop-downs)** 76
Figure 58: **Date entry (text – boxes, automated date enforcement)** 76
Figure 59: **Date entry (text – boxes, manual checking)** 76
Figure 60: **Date entry (text box & graphical calendar)** 77
Figure 61: **Visual studio: The New Project dialog** 80
Figure 62: **Visual studio: New project solution explorer** 80
Figure 63: **Mock-up of the page** 81
Figure 64: **Writing HTML (1)** 82
Figure 65: **Writing HTML (2)** 83
Figure 66: **Code for uploading file** 84
Figure 67: **Code for presenting uploaded files** 85
Figure 68: **Code to remove uploaded files** 85
Figure 69: **Task oriented mock-up of the page** 86
Figure 70: **Writing EAGER mark-up** 87
Figure 71: **Code for uploading file** 88
Figure 72: **Post topic (two examples of alternative representations)** 89
Figure 73: **Developing Ariadne with Visual Studio alone vs. with the EAGER toolkit** 90
Figure 74: **Total number of lines of pure code constituting the EAGER toolkit** 91
Figure 75: **The old HERMES portal** 93
Figure 76: **Total number of pure code constituting the new EDeAN portal** 94
Figure 77: **The public area (various skins)** 94
Figure 78: **Accessibility and Interaction options for portal visitors** 95
Figure 79: **Selection from a number of predefined profiles** 96
Figure 80: **The "Blind, no Assistive Technology, High Expertise" profile** 97
Figure 81: **The "Low vision, no Assistive Technology, Moderate Expertise" profile** 99
Figure 82: **The "Motor Impaired, two Switches, Low Expertise" profile** 100
Figure 83: **The "Colour Blind (Protanope), Low Expertise" profile** 101
Figure 84: **The "Colour Blind (Deuteranope), Moderate Expertise" profile** 101
Figure 85: **The "Colour Blind (Tritanope), High Expertise" profile** 102
Figure 86: **The Home Page for high display resolution (1024X768 or higher)** 103
Figure 87: **The Home Page for low display resolution (800X600 or lower)** 103
Figure 88: **The basic settings interface** 104
Figure 89: **The language settings interface** 105
Figure 90: **The disability settings interface** 105
Figure 91: **The Web Familiarity settings interface** 106
Figure 92: **The Device and Display resolution settings interface** 107
Figure 93: **The Input device selection interface** 108
Figure 94: **The Assistive Technology selection interface** 109
Figure 95: **The interaction preferences administration interface** 110
Figure 96: **The File uploading style selection interface** 111

Figure 97: **The File displaying style selection interface** 112
Figure 98: **The Paging style selection interface** 113
Figure 99: **The Date selection style interface** 114
Figure 100: **The Image uploading style selection interface** 115
Figure 101: **The Image displaying style selection interface** 116
Figure 102: **The Module function style selection interface** 117
Figure 103: **The Module options style selection interface** 118
Figure 104: **The Tabs style selection interface** 119
Figure 105: **The Text editing selection interface** 120
Figure 106: **The Search variations interface** 121
Figure 107: **The Favourite navigation options selection interface** 122
Figure 108: **The Navigation style selection interface** 122
Figure 109: **The Skin selection interface** 123
Figure 110: **The Font Family selection interface** 124
Figure 111: **The Accessibility preferences administration interface** 125
Figure 112: **Selecting the Template Linearization scheme** 126
Figure 113: **Selecting whether to display Quick Access Links** 127
Figure 114: **Selecting whether to enable Dynamic Adaptation** 127
Figure 115: **Selecting whether to display Section Breaks** 128
Figure 116: **The Table Linearization settings interface** 129
Figure 117: **The Chart settings interface** 130
Figure 118: **The Image settings interface** 131
Figure 119: **The Font settings interface** 131
Figure 120: **The Colour settings interface** 132
Figure 121: **The Text entry settings interface** 133
Figure 122: **The Fieldset presentation selection interface** 134
Figure 123: **The message board interfaces** 136
Figure 124: **A subset of possible adaptations for postings new topics** 137
Figure 125: **The Documents Area module (default view)** 139
Figure 126: **The Documents Area module (adaptation example)** 140
Figure 127: **The Chat facility (default view)** 141
Figure 128: **The Chat facility (adaptation example)** 142
Figure 129: **Link browsing process (default view)** 143
Figure 130: **Link browsing process (adaptation example)** 144
Figure 131: **Browsing Notifications (default view)** 145
Figure 132: **Browsing Notifications (adaptation example)** 146
Figure 133: **The Terms module (default view)** 147
Figure 134: **The Terms module (adaptation example)** 148
Figure 135: **The Email facility (default view)** 149
Figure 136: **The Email facility (adaptation example)** 150
Figure 137: **SIG Members facility (default view)** 151
Figure 138: **SIG Members facility (adaptation view)** 152
Figure 139: **EDeAN Training Course Browsing Facilities (default view)** 154
Figure 140: **Personal courses (default view)** 155
Figure 141: **Adaptation examples for the EDeAN online Training Course Navigator** 156
Figure 142: **Ariadne Browsing Facilities (default view)** 158
Figure 143: **Ariadne: Search Resources** 159
Figure 144: **Ariadne: Resource Details (default view)** 159
Figure 145: **Ariadne: Resource Details (adaptation example)** 160
Figure 146: **Home page of Network Administrators (various scins)** 161

Figure 147: **Statistics regarding the popularity of settings** 162
Figure 148: **Statistics regarding the popularity of preferences** 163
Figure 149: **Statistics regarding the popularity of custom accessibility settings** 164
Figure 150: **The Profile Administration facility** 165
Figure 151: **Template alternative styles** 191
Figure 152: **Template alternatives according to device resolution** 192
Figure 153: **Windows alternative styles** 193
Figure 154: **Navigation alternatives** 194
Figure 155: **Default – Quick navigation alternatives** 195
Figure 156: **Step by step navigation alternative** 196
Figure 157: **Navigation linearized (novice) alternative** 196
Figure 158: **Navigation linearized (moderate) alternative** 197
Figure 159: **Navigation linearized (expert) alternative** 197
Figure 160: **Navigation hiding and sorting based on frequency of use** 198
Figure 161: **Favourites navigation alternatives** 199
Figure 162: **Colour contrast** 200
Figure 163: **Colour wheel** 200
Figure 164: **Link – Button representations** 201
Figure 165: **Image alternative representations** 202
Figure 166: **Textboxes representations** 203
Figure 167: **Tabs representations** 204
Figure 168: **Fieldset alternatives** 204
Figure 169: **Table alternatives** 205
Figure 170: **No table linearization** 205
Figure 171: **Table linearization by row (no headings repeat)** 205
Figure 172: **Table linearization by row (repeat headings every line)** 206
Figure 173: **Table linearization by row (repeat headings every cell)** 206
Figure 174: **Table linearization by column (no headings repeat)** 206
Figure 175: **Table linearization by column (repeat headings every line)** 206
Figure 176: **Table linearization by column (repeat headings every cell)** 206
Figure 177: **Charts alternative representations** 207
Figure 178**: Module options alternative styles** 207
Figure 179**: Top left navigation window – Top right navigation window** 208
Figure 180: **Tab styles options (Inline navigation window vs. tabs without graphics)** 208
Figure 181: **Functions alternatives** 209
Figure 182: **Functions with default help alternative** 209
Figure 183: **Function with optional help alternative** 210
Figure 184: **Paging alternatives** 211
Figure 185: **Simple paging alternatives** 211
Figure 186: **Web-based Virtual Keyboard (overview)** 213
Figure 187: **Standard phone-like layout** 214
Figure 188: **Frequency based phone-like layout** 214
Figure 189: **Optimised layouts for 2-, 3- and 5- key text entry** 214
Figure 190: **Editor Alternatives** 215
Figure 191: **Date entry alternatives (1/2)** 216
Figure 192: **Date entry alternatives (2/2)** 217
Figure 193: **Upload files alternatives** 218
Figure 194: **File upload indirect manipulation alternative** 218
Figure 195: **File upload direct manipulation alternative** 219
Figure 196: **File upload mixed mode manipulation alternative** 220

Figure 197: **Upload images alternatives**.....221
Figure 198: **Image upload indirect manipulation alternative** 221
Figure 199: **Image upload direct manipulation alternative** 222
Figure 200: **Image upload mixed mode manipulation alternative** 223
Figure 201: **Files representations** 224
Figure 202: **Images representations** 224
Figure 203: **Display thumbnail, download link and estimated download time** 225
Figure 204: **Display full image info** 225
Figure 205: **Display thumbnail, download link and size** 225
Figure 206: **Display as a slide show (version 1)**.....225
Figure 207: **Display as a slide show (version 2)**.....226
Figure 208: **Search function alternatives** 227
Figure 209: **The hierarchy of the potential template variations** 230
Figure 210: **Header container diagram**.....232
Figure 211: **Header template diagrams** 233
Figure 212: **Header template control diagram**.....234
Figure 213: **Header sub templates diagram** 235
Figure 214: **Header sub templates controls** 236
Figure 215: **Footer template diagram**.....237
Figure 216: **Footer template control** 238
Figure 217: **Widget component diagram**.....239
Figure 218: **Top navigation diagram** 240
Figure 219: **Bottom navigation diagram**.....241
Figure 220: **Main navigation diagram**.....241
Figure 221: **Full navigation sub template control attributes**.....242
Figure 222: **Categories navigation diagram** 243
Figure 223: **Sub categories navigation diagram**.....243
Figure 224: **Option navigation diagram** 244
Figure 225: **Favourites navigation options** 244
Figure 226: **The alternative link controls** 245
Figure 227: **The icsButton controls** 245
Figure 228: **The class diagram representing the alternative selection controls** 246
Figure 229: **The alternative label controls**.....247
Figure 230: **The icsDropDown control** 247
Figure 231: **Images hierarchy** 248
Figure 232: **Tabs diagram**.....249
Figure 233: **Field set diagram** 250
Figure 234: **Tables diagram** 251
Figure 235: **BarChart diagram** 252
Figure 236: **Pie chart diagram**.....253
Figure 237: **Colour chart diagram**.....254
Figure 238: **Module options diagram** 255
Figure 239: **Functions diagram**.....256
Figure 240: **Paging diagrams**.....257
Figure 241: **Textbox diagram**.....258
Figure 242: **Web virtual keyboard diagram** 259
Figure 243: **Editor Diagram**.....260
Figure 244: **Date picker diagram**.....261
Figure 245: **File up loader diagram** 262
Figure 246: **Image uploader diagram** 263

Figure 247: **File displayer diagram** 264
Figure 248: **Image displayer diagrams** 265
Figure 249: **Image uploader diagram** 266

List of Tables

Table 1. **Types of Assistive technology and corresponding target users** 9
Table 2. **Web usability heuristics by K. Instone [**18**]** 15
Table 3: **An instance of design rationale documentation** 35
Table 4: **The *UserInformationComponent* class** 41
Table 5: **The *InteractionMonitoringModule* class** 42
Table 6: **The *ContextMonitoringModule* class** 44
Table 7: **The *DecisionMakingComponent* class** 45
Table 8: **The *DecisionLogic* class** 46
Table 9: **The *DialogueControlComponent* class** 48
Table 10: ***icsImageControlBase* class properties – methods** 52
Table 11: **Image control class hierarchy** 53
Table 12: ***icsImageDisplayerBase* class properties – methods** 53
Table 13: **Image displayers control library** 54
Table 14: ***icsMasterPageBase* class** 56
Table 15: **The available descendors of the *icsMasterPageBase* control** 57
Table 16: ***icsFileUploaderBase* class properties – methods** 61
Table 17: **File uploader class hierarchy** 62
Table 18: **'icsDatePickerBase' class properties – methods** 63
Table 19: **Date picker class hierarchy** 63
Table 20: **Design rational of the images alternatives** 64
Table 21: **Design rational of images representations** 67
Table 22: **Design rationale of the template styles** 68
Table 23: **Design rationale of the templates alternatives according to device resolution** 69
Table 24: **Design rationale of the step by step navigation (novice, moderate, expert)** 71
Table 25: **Design rationale of the upload files alternatives** 74
Table 26: **Design rationale of date input alternatives** 77
Table 27: **Black-Pink-Yellow-White colour setting evaluation** 176
Table 28: **Yellow-Blue-Blue-Black colour setting evaluation** 176
Table 29: **Blue-Pink-Yellow-White colour setting evaluation** 176
Table 30: **White-Blue-White-Black colour setting evaluation** 176
Table 31: **Overview of the quality of user experience of the EDeAN prototype** 183
Table 32: **Design rationale of the template styles** 192
Table 33: **Design rationale of template alternatives according to device resolution** 193
Table 34: **Design rationale of window alternative styles** 193
Table 35: **Design rationale of default and quick navigation alternatives** 195
Table 36: **Design rationale of the step by step navigation (novice, moderate, expert)** .. 197
Table 37: **Design rationale of navigation hiding, sorting, bookmarking** 199
Table 38: **Design rationale of Link alternatives** 201
Table 39: **Design rationale of Buttons alternatives** 201
Table 40: **Design rational of the images alternatives** 202
Table 41: **Design rationale of textbox representations** 203
Table 42: **Design rationale of tab alternative styles** 204
Table 43: **Design rationale of fieldset styles** 204
Table 44: **Design rationale of table styles** 206
Table 45: **Design rationale of chart representations** 207
Table 46: **Design rationale of module options alternative styles** 208

Table 47: **Design rational of functions alternatives** 210
Table 48: **Design rational of paging alternatives 1/2** 211
Table 49: **Design rational of paging alternatives 2/2** 212
Table 50: **Design rationale of text entry alternatives** 214
Table 51: **Design rational of the editor alternatives** 215
Table 52: **Design rationale of the date entry alternatives** 217
Table 53: **Design rationale of the upload files alternatives** 220
Table 54: **Design rationale of the upload image alternatives** 223
Table 55: **Design rationale of files representations** 224
Table 56: **Design rational of images representations** 226
Table 57: Design **rational of the search functions alternatives** 227
Table 58: **Conventions followed by the Microsoft Visual Studio 2005 class designer** ... 229

Chapter 1
INTRODUCTION

1.1 Computers and User Interfaces

The *user interface* (UI), as the aggregate of means by which people (the users) interact with computer programs, provides means of: (a) input, allowing the users to manipulate a system, and (b) output, allowing the system to produce the effects of the users' manipulation.

Through their first decades, UIs were mainly implemented as *command line interfaces* (CLIs), which allowed the user to interact with an operating system using a command line interpreter. Even after the introduction of additional interaction styles, commonly referred as *Question and Answer*, *Menu Selection*, *Function Key Selection*, and *Form Fill-In* [1], the human-computer dialogue reflected the computer's preferences, which demanded rigid, typed input through keyboards. At that time, a computer's ability to deal with human communication was inversely related to what was easy for people to do and thus exploited only by programmers, system administrators in engineering, scientific environments, and by technically advanced personal computer users. Finally, in 1970s, another dialog alternative surfaced in reaction to the steep learning curve of CLIs. *Graphical user interfaces* (GUIs), using a form of human gesturing and employing the mouse for pointing and selecting as the primary human-computer communication method. Although GUIs, provided ways to communicate with computers that were easier to learn and remember than with CLIs, they still required from the users to follow strict paths, and employ manual dexterities. Nonetheless, at that time, GUIs reflected a task-oriented approach to UI design paying low attention to important user characteristics and needs. A significant amount of research was then put into *human-computer interaction* (HCI) and in finding ways to make modern UIs more flexible and user-centered.

Recently, computer-based products have become associated with a great amount of daily user activities, such as work, communication, education, entertainment, etc. Their target population has changed dramatically. Users are no longer only the traditional able-bodied, skilled and computer-literate professionals. Instead, users are potentially all citizens of an emerging *Information Society* and they demand customised solutions to obtain timely access to any application, irrespective of where and how it runs. At the same time, the type and context of use of interactive applications is radically changing due to the increasing availability of novel *Information and Communication Technologies* (ICT, e.g., personal digital assistants, kiosks, cellular phones and other network-attachable equipment) which progressively enable nomadic access to information [3]. As characteristic result of this evolution, modern UIs have radically changed in order to cope with the increasing volume and variance of user needs and requirements. Today, several years later, the way that user interfaces are perceived has changed dramatically. Notably, the amount of programming code devoted to the user interface nowadays exceeds 50 percent [2].

1.2 About User Interface adaptation

In computing, the notion and importance of *adaptation*, as the ability to adapt a system to the user's needs, expertise and requirements was only recently recognised. In a natural environment, the survival of all living organisms is often, if not constantly, subject to their ability of constantly changing, adjusting and functioning according to the surrounding environment.

In similar terms, the "survival" of users (in other words, system adoption) could also be considered as subject to:

(a) their ability to adapt to a given system (environment), on the one hand, and
(b) on the other hand, to their freedom to modify the settings of the system in question.

Clearly, early forms of UIs were rather based on the first case alone. This was mainly due to the fact that early interfaces had to be taken as de facto, restricting their users to make convenient changes to the style, presentation and behaviour of a given UI. A radical change in this pattern has recently emerged with the introduction of adaptation in user interfaces: the computationally empowered environment can adapt itself, at various degrees, to its 'inhabitants', thereby reducing drastically the amount of effort required from the user's part.

Methods and techniques for *user interface adaptation* meet significant success in modern interfaces. Some popular examples include the desktop adaptations in *Microsoft Windows XP*, offering, for example, the ability to hide or delete unused desktop items. *Microsoft Windows Vista* also offer various personalisation features of the desktop based on personal preferences of the user, by adding helpful animations, transparent glass menu bars, live thumbnail previews of open programs and desktop gadgets (like clocks, calendars, weather forecast, etc.). Similarly, *Microsoft Office* applications offer several customisations, such as toolbars positioning and showing/hiding recently used options.

However, adaptations integrated into modern systems merely focus on usability and aesthetics. In terms of accessibility, for instance to people with disability, only a limited number of adaptations have been proposed, and developed a posteriori, including keyboard shortcuts, size and zoom options, changing colour and sound settings, automated tasks, etc.

According to [4], a better approach to ensure accessibility and usability and meeting the individual target user requirements, abilities, preferences of the user population at large, including disabled and elderly people, is to provide *User Interfaces for All* by following the principles of *design for all* in HCI.

The *Unified User Interfaces* (U^2I) methodology, which involves UI adaptation, was then conceived and verified [3] as a vehicle to efficiently and effectively ensure during the interface development process the accessibility and usability of UIs to users with diverse characteristics, supporting also technological platform independence, metaphor independence and user-profile independence.

1.3 Web-based User Interfaces: Opportunities and Challenges

Web-based user interfaces (WUIs), constitute a particular type of UIs that accept input and provide output by generating Web pages that are transported via the Internet and viewed by the user using a Web browser program. Despite the universality of the Web and the predominant role of WUIs in the evolving Information Society, current approaches in Web design hardly embrace the notion of adaptation and principles of *design for all*. In fact, only very recently, few Web applications started providing some sort of adaptation to their users. Indicatively, the *iGoogle*[2] website and the *Microsoft Sharepoint portal server*[3] offer users the ability to customise the UI, such as repositioning, minimising and maximising webpage elements, defining the number of search results to be displayed, and personalising the colour settings. Clearly, although some of these features are well appreciated by some users, they can not alone support the delivery of *qualitative user experience for all*, regardless of the user's (dis)abilities, skills, preferences, and context of use. For instance, the particular ways in which most of these features are implemented, render the produced webpages completely inaccessible for blind individuals that use screen readers.

Admittedly, the Web can play a key role in many aspects of our everyday life serving as an unprecedented resource of knowledge, information and services, including news and information, commerce, entertainment, classroom education and distance learning, job searching and remote collaboration, social participation and government services. Web, thanks to its universality, has the potential of reaching an enormous number of individuals, with diverse characteristics, populations and under various contexts (personal computers, mobile phones with small display screens, web-TV, kiosks etc). Consequently, today the main challenge for WUI designers and researchers is to come-up with approaches that can meet diverse user needs and requirements in various contexts. Contemporary users desire and expect the delivery of interfaces that ensure personalisation and satisfying levels of freedom to decide which way to interact with the Web, rather than compromise on de facto, designs for "average" users.

In addition to those limitations of current WUIs, which actually concern the end user, one needs consider that the design and development of a Web application that meets the needs and requirements of as many users as possible is evolving into an increasingly difficult and demanding task for Web developers. As a result, a vast majority of developers today compromise on designing a Web application for the "typical" or "average" user, taking this as the best solution to cater the needs of the broadest possible population. Unfortunately, this approach leads in excluding various categories of users, such as non-expert IT users, the very young and elderly people, people with disability, etc. [5]. Specialised designs for one user group often constrain the capabilities of another still important group. As a result, design needs to be equally targeted towards all potential users [5].

[2] http://www.google.com/ig

[3] http://sharepoint.microsoft.com/sharepoint/

1.4 Towards Unified Web-based Interfaces

The work presented in this report is motivated from the emerging need to support individuals to fully participate in the knowledge society, especially people at risk of exclusion. To offer means to address challenges associated to population aging and high disability rates[4] all over the world. In these terms, the main goal of this work is to contribute to the collective vision to mainstream and radically improve the accessibility, usability and, in general, user experience and acceptance of computer-based products and services, and thereby also help in reducing the 30% of the European population currently not using ICT.

In particular, this work focuses on WUIs, and proposes a novel approach, based on *Web portal technologies* and UI adaptation techniques, to embed personalised accessibility features deep into Web design. A portal typically includes a number of facilities for access and navigation of information, socialisation, collaboration and trade. It can be defined as a website acting as a gateway on the way to access a multitude of online destinations that are somehow related and which can be referred to as the contents of the portal. For example: (a) Web pages, websites and other Web-based applications (including other portals); (b) people connected on the Web[5]; and (c) digital resources[6], such as documents, multimedia, software, etc. Typically, portals include also various *portlets*, which are reusable Web components that display relevant information to portal users such as news, email, weather information, and discussion forums.

An adaptation of the U^{2}I methodology mentioned in 1.2 is proposed here for developing Web-portals following a design for all approach and for ensuring the delivery of *Unified Web-based Interfaces*[7] (UWIs) that support adaptation to diverse user needs, (accessibility) requirements and contexts of use.

In order to further facilitate Web developers in following the proposed UWI method in practice, this work has also focused on the development of an advanced developer's toolkit, namely *EAGER*[8]. By means of EAGER, a developer can produce Web portals that have the ability to adapt to the interaction modalities, metaphors and UI elements most appropriate to each individual user, according to profile information containing user and context specific parameters. The characteristics of the EAGER toolkit are presented in this report along with guidance for Web developers. Finally, a prototype portal, namely the EDeAN portal, developed for validation purposes, is presented here as a case study serving Web developers as a use guide of EAGER.

[4] According to the European Commission (see FP7, ICT work programme 2007-08), there is a growing part of the population that is over 60 and at the same time a 15% of the EU population has a disability, of which 70% will be over 60 by 2020.

[5] In these case we usually use the notion *community portals*

[6] In these cases we usually use the notion of *digital library portals*

[7] The notion of "Unified Web-based Interfaces" is used here to refer to Web-based UIs that comply with the *Unified User Interfaces* (U^{2}I) methodology [3].

[8] EAGER stands for "toolkit for **e**mbedding **a**ccessibility, **g**raceful transformation and **e**ase of use in Web–based products".

1.5 Structure and contents of this report

The rest of this report is organised as follows.

Chapter 2 provides background information required for understanding the context of this work. It introduces the reader to the basics of accessibility, usability and adaptation techniques. More specifically an overview of existing solutions for providing Web accessibility, mainly to people with disability, is discussed both from a user and a developer's perspective.

Chapter 3 presents the theory behind this work. It starts with an introduction to the concept of User Interfaces for All, a proactive paradigm for the development of systems accommodating the broadest possible end-user population, followed by an overview of the U^2I methodology, an approach for developing interfaces for all. The latter, in the context of this work, is adapted in order to incorporate the specific characteristic of the Web and of portal-based applications, leading to the concept of UWIs for developing Web portals for all. Finally, this Chapter presents the approach for validating the UWI methodology and the derived toolkit for Web developers (EAGER).

Chapter 4 describes in detail the proposed UWI methodology. To this effect, the concept of adaptive and adaptable behaviour is presented followed by an overview of the architectural solution proposed. The Chapter concludes providing an overview of the extensibility options supported by this method.

Chapter 5 provides an overview of the EAGER toolkit for Web developers (a detailed description of EAGER is available in Appendix B - see pp. 229). More specifically, this section describes the proposed toolkit in relation to the UWI methodology. The Chapter concludes with the definition of the design space of the EAGER toolkit and the presentation of indicative design scenarios for the repository of alternative interaction dialogues (a full overview of the alternative designs currently supported by EAGER is available in Appendix A – see pp. 191).

Chapter 6 compares the process of developing a portal with Microsoft Visual Studio IDE alone, with the process of developing it by means of the EAGER toolkit (in combination with Microsoft Visual Studio). The Chapter concludes by presenting the benefits of using the EAGER toolkit, both to developers and end-users.

Chapter 7 presents the EDeAN portal, a prototype developed by means of EAGER. More specifically, we present the profile-related facilities offered to end-users followed by a complete presentation of the modules developed for a fully functional portal. Examples of alternative personalisation schemes of these modules are presented highlighting the adaptive behaviour inherited by the EAGER toolkit.

Chapter 8 describes the evaluation conducted for validating the results of this work. On the one hand, the accessibility characteristics of the alternative UI elements produced were evaluated according to processes proposed by the World Wide Web Consortium (W3C) and, on the other hand, the overall user-experience of the EDeAN portal, including its usability, was investigated using an evaluation framework recently introduced in the international literature.

Finally, Chapter 9 discusses the conclusions of this work. It provides a summary of the processes followed for the design and development both of EAGER and of the prototype portal, and highlights the potential techniques that could be applied in the context of future amendments for improving the effectiveness and efficiency of the EAGER toolkit. In conclusion, some other future work directions are outlined.

Chapter 2
RELATED AND BACKGROUND WORK

One of the main goals of the work presented in this report is to support the design and development of accessible and usable web-based interfaces the allow adaptation to individual user needs and preferences, since this task can be very difficult, time and resources demanding, and may in many occasions result in interfaces that generate disorientation, confusion and upsetting. Previous related work mainly focuses on three complementary research directions: *web accessibility*, *web usability* and *adaptation techniques for the Web*.

2.1 Web accessibility

This section focuses on the concept of accessibility as a perceived characteristic and quality measurement of computer-based systems, including and Web-based applications. In general, accessibility is considered as prerequisite for usability, since there cannot be optimal interaction if there is not the possibility of interaction in the first place. An overview of existing practices, methods, and tools for achieving accessibility is also provided.

2.1.1 Definition of accessibility

Accessibility is traditionally associated with disabled and elderly people and reflects the efforts devoted to the task of meeting prescribed code requirements for use by people with disabilities [6, 7]. However, due to recent technological developments (e.g., proliferation of interaction platforms, such as wireless computing, wearable equipment, user terminals), the range of the population which may gradually be confronted with accessibility problems extends beyond the population of disabled and elderly users.

According to [5] "*accessibility implies the global requirement for access to information by individuals with different abilities, requirements and preferences, in a variety of contexts of use; the meaning of the term is intentionally broad to encompass accessibility challenges as posed by diversity in:*

a) the target user population profile (including people with special needs) and their individual and cultural differences;

b) the scope and nature of tasks (especially as related to the shift from business tasks to communication and collaboration intensive computer-mediated human activities);

c) the technological platforms and associated devices through which information is accessed."

Several efforts towards accessibility of computer-based applications and services for disabled and elderly people have been done through a posteriori adaptation of interactive software. This amounts to a "reactive" approach, whereby Assistive Technology experts attempted to respond to contemporary technological developments by building accessibility features into interactive applications, as a

result of specific user requirements. Although the reactive approach to accessibility may be the only viable solution in certain cases [8], it suffers from some serious shortcomings, especially when considering the radically changing technological environment, and, in particular, the emerging Information Society technologies.

2.1.2 Assistive technology and accessibility features in Operating Systems

Assistive Technology (AT) is a generic term that includes assistive, adaptive, and rehabilitative devices and the process used in selecting, locating, and using them. AT promotes accessibility and greater independence for people with disability by enabling them to perform tasks that they were formerly unable to accomplish, or had great difficulty accomplishing, by providing enhancements to or changed methods of interacting with the technology needed to accomplish such tasks. According to disability advocates, technology is often created without regard to people with disability, creating unnecessary barriers to hundreds of millions of people. AT products are designed to provide additional accessibility to individuals who have physical or cognitive difficulties, impairments, and disabilities.

Clearly, assistive technology includes products that are placed “on top” of an existing system and used in a complementary way. Therefore, when selecting AT products, it is crucial to find products that are compatible with the computer operating system and programs on the particular computer being used.Table 1, provides an overview of the existing categories of AT products, along with their corresponding target users.

Further to the definition of accessibility given in section 2.1.1, accessibility is subject, among others, to the technological platforms through which information is accessed. In these terms, before reviewing the accessibility of Web-based systems, one needs to be aware of the existing accessibility aids embedded in operating systems. In many cases, throughout all these years, AT products that were proved to be widely useful were also integrated into mainstream operating systems. The term “alternative access based systems” is used to define those systems that support additional input and output devices and provide alternative interaction metaphors in terms of the operating system or the graphic environment. These systems are widely used in conjunction with AT by the majority of disabled users for accessing the web. Most modern operating systems support a number of services of this category in order to make possible at least the initiation of interaction with these systems.

The facilities offered by operating systems include aids for visual impairments, such as *screen magnifiers* and *screen readers-narrators.* Screen magnifiers enable users who are partially sighted to view selected areas of the screen, in a manner similar to using a magnifying glass by magnifying the screen up to 20x and by automatically following the mouse cursor. Screen readers or narrators are text-to-speech utilities that read what is displayed on the screen such as the contents of the active window, menu options, or text that has been typed. Additionally, some operating systems support adjusting the screen to white on black or gray scale for improve the contrast of the screen.

Table 1. **Types of Assistive technology and corresponding target users**

Types of AT	Purpose	Main users supported
Alternative keyboards	Featuring larger- or smaller-than-standard keys or keyboards, alternative key configurations, and keyboards for use with one hand.	Motor impaired
Electronic pointing devices	Used to control the cursor on the screen without use of hands.	Motor impaired
Sip-and-puff systems	Activated by inhaling or exhaling, as a means for selection.	Motor impaired
Wands and sticks	Worn on the head, held in the mouth or strapped to the chin and used to press keys on the keyboard.	Motor impaired
Joysticks	Manipulated by hand, feet, etc. and used to control the cursor on screen.	Motor impaired
Trackballs	Movable balls on top of a base that can be used to move the cursor on screen.	Motor impaired
Touch screens	Allow direct selection or activation of the computer by touching the screen, making it easier to select an option directly rather than through a mouse movement or keyboard.	Motor impaired
Braille embossers	Transfer computer generated text into embossed Braille output.	Blind
Keyboard filters	Constitute typing aids, such as word prediction utilities and add-on spelling checkers, which reduce the required number of keystrokes.	Motor impaired
Light signaler alerts	Monitor computer sounds and alert the user with light signals.	Deaf
On-screen keyboards	Provide an image of a standard or modified keyboard on the computer screen that allows the user to select keys with a mouse, touch screen, trackball, joystick, switch, or electronic pointing device.	Motor impaired
Reading tools and learning disabilities programs	Include software and hardware designed to make text-based materials more accessible for people who have difficulty with reading	Visually or cognitively impaired
Refreshable Braille displays	Provide tactile (Braille) output of information represented on the computer screen.	Blind
Screen magnifiers	Work like a magnifying glass for the computer by enlarging a portion of the screen which can increase legibility and make it easier to see items on the computer.	Visually impaired
Screen readers	Are used to verbalise everything on the screen including text, graphics, control buttons, and menus into a computerised voice that is spoken aloud. In essence, a screen reader transforms a GUI into an audio interface.	Blind
Speech recognition programs	Allow people to give commands and enter data using their voices rather than a mouse or keyboard.	Motor impaired
Text-to-Speech (TTS) or speech synthesizers	Receive information going to the screen in the form of letters, numbers, and punctuation marks, and then "speak" it out loud in a computerised voice.	Blind

Targeting to motor impairments, modern operating systems support on *screen keyboards*, *sticky keys, key commands* and *speech recognition*. *On-screen keyboard* displays a virtual keyboard with all the standard keys on the computer screen that allows people to type data by using a pointing device or joystick. *Sticky keys* allow the user to enter "key chords" sequentially. With sticky Keys active, the operating system displays each key in the sequence on screen so the user can correct and verify the sequence before executing it. *Key commands* are used to navigate through menus, windows using only the keyboard. The user has the ability to assign and reassign key commands used system wide or in individual applications. *Speech recognition* enables

user to interact with the computer using only voice while maintaining, or even increasing, productivity.

Finally, in the case of hearing impairments, operating systems support flashing the screen instead of playing an audio system beep, as well as quick-access key commands to adjust the system volume.

Popular operating systems *Microsoft Windows Vista*[9,10], *Microsoft Windows XP*[11,12], *Linux* [9] and *MacOs X*[13] include all the aforementioned built-in accessibility services that make it easier for computer users to see, hear, and use their computers.

Services offered by modern operating systems enabled users with disability to enter the world of computing. All the issues that arise in the context of accessibility are based on the fact that the user has the ability to initiate interaction with a computer system, and these facilities play a fundamental role for enabling this to happen. However, web accessibility involves totally different issues. Often, facilities offered by operating systems and the incorporation of additional input or output devices can not cope with the barriers set by the modern Web. Web portals and sites can not be accessed using only built-in facilities, due to the diversity and complexity of the technologies and approaches followed by the Web. For example, some Web applications use client-side scripting to such an extent that they are rendered inaccessible, even with the help of AT in conjunction with the aforementioned operating system features. It is therefore clear that the accessibility options provided by modern operating systems are valuable for enabling the initiation of interaction with a computer system, but can not to cope with the numerous barriers set for people with disabilities by modern Web applications.

2.1.3 User agents (Web browsers)

Towards the same direction, popular *user agents* (Web browsers), such as *Internet Explorer*, *Netscape Navigator*, *Firefox* and *Opera* encapsulate several accessibility settings and features. These accessibility features include support on 'zooming in' on a Web page to magnify text, images, and controls on the page. Additionally, users may choose colours, text size, and font style used on Web pages to make Web pages easier to see by changing the text, background, link and hover colours. Users may also choose to ignore colours, font styles, and font sizes specified on Web pages to make the pages easier to see or may choose to format documents using their own style sheet. Advanced accessibility settings include expanding alternative text for images, moving system caret with focus/selection changes, resetting text size to medium for new windows and tabs. Users may also turn off or turn on pictures, animations, and videos, music and other sounds included in Web pages. Some web pages provide a rich interactive experience with Java applets. However, users who rely on keyboard navigation may experience problems with some Java applets that automatically set focus and do not provide a way to "break out" of the applet and navigate to the rest of the web page. Modern browsers support particular setting in order to disable Java and JavaScript. Finally, some browsers include keyboard and mouse shortcuts in order to reduce the need for special users to reach mouse or keyboard.

---

[9] Accessibility Tutorials for Windows Vista: http://www.microsoft.com/enable/products/windowsvista/default.aspx

[10] Accessibility in Windows Vista: http://www.microsoft.com/enable/products/windowsvista/default.aspx

[11] Windows XP Accessibility Resources: http://www.microsoft.com/enable/products/windowsxp/default.aspx

[12] Windows XP Accessibility Tutorials: http://www.microsoft.com/enable/training/windowsxp/default.aspx

[13] MacOs X Users Manual: http://www.apple.com/macosx/features/universalaccess/

People with disability may encounter several difficulties while trying access the Web, regardless the fact that web browsers accessibility settings are properly configured because some websites are designed to defeat browser settings. Browser settings should work when websites meet some basic web accessibility guidelines and are designed for accessibility, flexibility (see next section). As an example, Web browser's text resizing feature does not work well in websites that do not meet accessibility guidelines and define text size using "hard-coded" or "absolute" sizes.

2.1.4 Web content accessibility - Evaluation tools

As mentioned in the previous section, AT or other alternative access systems alone are not in the position to ensure accessibility to Web-based products and services. Web-content needs to be implemented in appropriate ways that allow graceful transformation of the content in alternative modalities. Towards this direction, a number of guidelines collections for Web accessibility have been developed [10, 11, 12, 13]. The *Web Content Accessibility Guidelines* (WCAG) [10] developed by the *World Wide Web Consortium* (W3C) provide recommendations for making Web content accessible to people with disability. Web "content" generally refers to the information in a webpage or Web application, including text, images, forms, sounds, etc. WCAG 1.0[14] provides 14 guidelines that are general principles of accessible design. Each guideline has one or more checkpoints that explain how the guideline applies in a specific area. On the other hand WCAG 2.0[15] provides guidelines and success criteria that are organized around to four principles, *perceivable, operable, understandable* and *robust,* that lay the foundation necessary for anyone to access and use Web content. W3C has also defined 3 levels of compliance (A, AA and AAA) to the provided WCAG. Each level requires a stricter set of conformance to guidelines, such as different versions of HTML (Transitional vs. Strict) and other techniques that need to be incorporated into code before accomplishing validation. Another well established source of web accessibility guidance comes from the US government: Section 508 of the US Rehabilitation Act. It is a comprehensive set of rules designed to help Web designers make their sites accessible.

In general, for a website to comply with accessibility standards, they should at least have the following characteristics:

- Valid (X)HTML
- Valid CSS for pages layout
- At least WAI-A (preferably AAA) compliance with the WAI's WCAG
- Compliance with Section 508 of the US Rehabilitation Act

The process of using, or testing conformance to, widely accepted accessibility guidelines is time consuming and requires additional developer expertise in accessibility. To address this issue, several tools have been developed enabling the semi automatic checking of html documents. Such tools make easier the development of accessible web content especially due to the fact that the checking of conformance does not rely solely on the expertise of developers. Developers with limited experience in web accessibility can use such tools for evaluating web content and without the need to go through a large number of check – lists. These tools can be separated to tools that check accessibility of (a) CSS (b) HTML – XHTML (c)

[14] http://www.w3.org/TR/WCAG10/

[15] http://www.w3.org/TR/WCAG20/

images. There is a great amount of such tools that are provided as (a) online services, (b) hosted services, (c) desktop applications and (d) browser plug-ins. Some tools provide have the ability to generate reports in HTML, XML or in simple text.

The *W3C XHTML validator*[16] is a free service published by W3C for checking the generation of valid mark-up language. On the other hand *W3C CSS Validation Service*[17] is a free service that checks Cascading Style Sheets (CSS) in (X)HTML documents or standalone for conformance to W3C recommendations. Some popular web accessibility tools include *Watchfire Bobby* that checks compliance with WCAG and Section 508 guidelines is a desktop application and offers automatically generated reports. *WebXACT*[18] is a free online service that also enables users to test single pages of web content for accessibility, quality and privacy issues. The *AccessColour*[19] utility analyses the internal and external CSS of a web page to test colour brightness and the colour contrast between the text and background colours using the algorithm recommended by the W3C. *The Lynx Viewer*[20] allows webmasters to see what their pages will look like when viewed with Lynx, a text-mode web browser and helps developers to determine if web pages are accessible to the vision impaired.

The usage of guidelines and tools is today the most widely adopted process by web authors for creating accessible web content. This approach has proven valuable for bridging a number of barriers faced today by people with disabilities. Unfortunately, many limitations arise due to a number of reasons. Guidelines themselves pose several issues regarding their ease of use and adoption by developers. On the other hand accessibility tools as application or online services interpret each and every accessibility guideline literally, without having the ability to check it as part of the page. For example[21], one of the W3C accessibility guidelines states that a table must include a hidden summary to be read aloud to screen reader users before reading through the table content. However, there may be a heading directly before the table to describe what the table is about, so the hidden summary will just repeat what the previous heading said.

Accessibility tools cannot check the websites content structure thus a website may be perfectly coded and conform to the highest coding standards by an accessibility tool but it will be inaccessible for some special needs web users when its content is poorly structured. Automated accessibility tools check typically a guideline but this is not adequate. For example, when a website includes alternative text for all the images, an accessibility tool will report a pass for this guideline, but it has not checked if the alternative texts are descriptive for the images. At last accessibility tools provide warnings apart form errors which are basically guidelines that the automated tool can not check and the evaluator has to check them manually. As a result automated accessibility tools can be useful and save a large amount of time but must be used with caution and not as stand alone guides. Developers that wish to deliver products that align with Web content accessibility guidelines must be educated themselves and spend much of their personal resources for this task. It is therefore clear that, in order to achieve web accessibility, effort must be paid by large industries for delivering conformance with guidelines in their development frameworks.

---

16 http://validator.w3.org/

17 http://jigsaw.w3.org/css-validator/

18 http://webxact.watchfire.com/

19 http://www.accesskeys.org/tools/colour-contrast.html

20 http://www.yellowpipe.com/yis/tools/lynx/lynx_viewer.php

21 http://www.webcredible.co.uk/user-friendly-resources/web-accessibility/automated-tools.shtml

2.1.5 Development environments and tools

Commercially available frameworks for developing web applications have not paid enough attention to incorporating the aforementioned guidelines into their products. Most frameworks rely only to producing XHTML valid mark-up and not on making guidelines a part of the logic used for producing the output. Additionally, the race of making web applications more client dependent and therefore decrease the server side load has produced solutions that largely rely on client side scripting, thus worsening interaction for disabled users. The most recent web developer's tool that is promising for accessibility *Visual Studio 2005*[22] only includes some controls that can be used for accessibility compliance such as skip navigation functions and images that support an alternative text and an extended description. Additionally, this developer's tool encapsulates an accessibility validation program that checks if the code generated by the pages is firstly valid XHTML and then adheres to the Web Accessibility Initiative (WAI) accessibility standards of W3C.

2.2 Web usability

This section focuses on the concept of usability as a perceived characteristic and quality measurement of computer-based and Web-based systems. An overview of existing practices, methods, and tools for achieving accessibility is also provided.

2.2.1 Definition of usability

Usability was first introduced and recognised long time before the appearance of computer systems and digital technologies ("*It is not the utility, but the usability of a thing which is in question*" [14]). It was as automation and machines started to be integrated into everyday life that usability started to be investigated more in depth. The emergence of digital systems led to the establishment of usability as a scientific filed of inquiry dealing with quality aspects of such systems (e.g., effectiveness, efficiency and aesthetics). Building on the numerous approaches in bibliography to define usability, usability is defined in [15] as: *"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"*.

The same ISO standard provided a more evolved framework (see Figure 1) for specifying and measuring the usability of products. According to this framework:

- *Usability is measured by the extent to which the intended goals of users are achieved (effectiveness), the resources that have been expended to achieve these goals (efficiency) and the extent to which the users find the use of the product acceptable (satisfaction).*
- *Emphasis is given to the context of use and to the fact that the level of usability depends on the specific circumstances in which the product is used.*

The proposed framework consists of three components: (a) the *context of use* (the users, the equipment, the environment, the goals & the tasks); (b) the *usability*

[22] http://www.it-analysis.com/content.php?articleid=13021

measures (effectiveness, efficiency and satisfaction); and (c) the users' *goals* of use of the product.

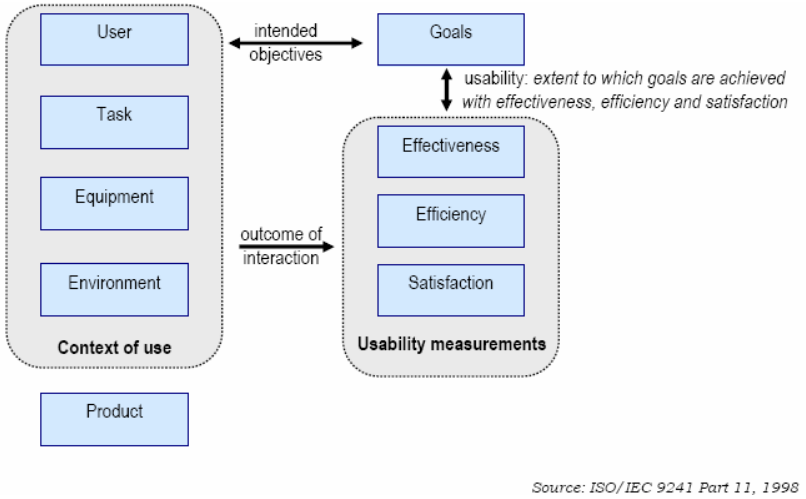


Figure 1: **ISO framework for specifying and measuring the Usability of products [**15**]**

J. Nielsen [16] and B. Shneiderman [17] have both described a system's acceptability, where usability is a part of "usefulness" and is composed of:

- Learnability (e.g., intuitive navigation)
- Efficiency of use
- Memorability
- Few and non-catastrophic errors
- Subjective satisfaction

2.2.2 Principles and heuristics

Usability is now recognised as an important software quality attribute, earning its place among more traditional attributes, such as performance and robustness. Usability plays a key role in order to make the Web friendly to all target users. General usability principles are applied to the Web by means of usability heuristics proposed by experts. This method is not always productive, because the concept of desktop applications is different from the Web. As an example, in most desktop applications there is a main working area along with a toolbar for the necessary supporting tools. A heuristic rule would be: "place all the appropriate tools for user tasks in a toolbar on the top". If we use this guideline for the Web, this will lead to inappropriate designs. Web applications are more task sequencing oriented in conjunction to desktop application that are multi-tasking oriented. Therefore, a more proper guideline would be: "place only the appropriate tools for the current user task in a toolbar" to support user for the current task without inducing confusion.

Generally, there are several resources that include usability principles and guidelines in detail but are interspersed in books, web and, papers. Additionally, usability is a subjective issue, therefore in order to ensure web site's usability, after applying several usability principles, a usability testing process have to be applied too with real users. A web usability test is an essential element of quality assurance and a true test of how people actually use a web site. An example of the most established usability heuristics for Web were developed by K. Instone [18] building on ten general usability heuristics defined by J. Nielsen [16], and are presented Table 2.

Table 2. **Web usability heuristics by K. Instone [**18**]**

Web Usability Heuristics	Description
Visibility of system status	*Each page has to be branded with the section it belongs to, and links to other pages should be clearly marked.*
Match between system and the real world	*Language used has to be simple and comprehensive to serve people form diverse backgrounds.*
User control and freedom	*Site has to provide several controls to users in order to assist customisations of the site and navigation in it.*
Consistency and standards	*Content and links wording has to be used consistently to avoid user confusion. Web "standards" HTML specifications, accessibility guidelines and etc has to be followed.*
Error prevention	*User input has to be checked before submitting to prevent errors, but have to be double checked after submission.*
Recognition rather than recall	*Labels and descriptive links have to be used in order to inform users about where they are, by looking on current page.*
Flexibility and efficiency of use	*Pages have to be easily bookmarked and offer intelligent book marking to specific corner of the site.*
Aesthetic and minimalist design	*Content of web pages has to be separated in different level of detail and provide alternative ways to access them. Content has to be also separated to relevant content chunks and provide access to them by different links.*
Help users recognize, diagnose, and recover from errors	*For every error message a solution (or a link to a solution) has to be provided on the error page.*
Help and documentation	*Help and documentation has to be integrated in the site. Help has to be context-sensitive for each page of the site.*

2.3 Adaptation techniques for the Web

2.3.1 Alternative User Agents

To ensure seamless access to the World Wide Web, several approaches involved the development of special purpose user agents (web browsers).

The *AVANTI Web Browser* [19] facilitates static and dynamic adaptations in order to adapt to the skills, desires and needs of each user including people with visual and motor disabilities. The AVANTI's unified interface can adapt itself to suit the requirements of three user categories: able-bodied, blind and motor impaired. Adaptability and adaptivity are used extensively to tailor and enhance the interface respectively, in order to effectively and efficiently meet the target of interface individualisation for end users. Special purpose input/output devices have been integrated into the system to support blind and motor-impaired individuals. Additionally, the unified browser interface implements features, which assist and enhance user interaction with the system. Such features include enhanced history control for blind and sighted users, link review and selection acceleration facilities, document review and navigation acceleration facilities, enhanced intra-document searching facilities etc.

WebAdapter [20] is a Web agent that provides accessibility functionalities for blind, visual and physically impaired people. This agent is based on the idea to include these functionalities within a browser without affecting the UI for non disabled users. In particular, the adaptations provided by WebAdapter include adaptation for physically impaired users (customization of images sizes), adaptations for visually impaired users (turning-off of background images), and finally, adaptations for blind users (sequential presentation of tables). WebAdapter uses also an integrated speech synthesizer.

pwWebSpeak32[23] is a commercially available web agent designed and developed by SoundsLink for users who wish to access the Internet in a non-visual or combined auditory and visual way. This includes blind or partially sighted users, people with dyslexia or learning difficulties, and users who are learning new languages. pwWebSpeak32 is designed to provide direct audio access to the information contained in a web page via TTS (Text-To-Speech). The greatest advantage of this agent is the use of intelligent techniques for transforming visual object to their non – visual representation. The intelligence built into pwWebSpeak32 understands the HTML constructs and automatically presents them in a logical sequence, greatly simplifying navigation of even complex documents.

Another special purpose agent is proposed in [21] including sophisticated adaptation mechanisms to provide context-aware behaviour and user interfaces. The aforementioned agent adapts the interface it presents to the user according to the type of the display that is used. In particular, fonts and widgets are adjusted to match the size of the screen. Additionally, the browser conveys information about the display to HTTP servers using the context header field, enabling the servers to adapt their responses accordingly. If the display type changes at run-time, the browser dynamically alters its interface and behaviour in response.

In general, the concept of special purpose agents is a very promising field in web accessibility and usability. Special purpose agents can deliver a number of facilities that include alternative interaction modalities and UI elements, support for a number of assistive technologies and input output devices, text to speech and speech recognition facilities. These agents offer the advantages of desktop based processing together with the positive features of intermediary frameworks acting themselves as a proxy between the web and the user. However, these approaches are limited by the fact that the user must have the actual product installed on the computer used to gain access to the Web. This issue would not be a barrier if all agents were actually available in the market. Unfortunately, many of the most promising approaches are products of research that are not widely available. It is therefore clear that these facilities must be either presented as commercial products or be embedded in existing mainstream Web browsers in order for the aforementioned benefits to be reach their actual beneficiaries.

2.3.2 Intermediary Agents

Intermediary agents can be considered as filtering and transformation tools that are used to build alternative versions of webpages based on disability category, user preferences, or heuristic rules.

WebFace [22] constitutes a representative intermediate agent supporting accessibility with respect to physical or perceptual disabilities, or combinations thereof. It is a tool that enables dynamic transformation of webpages to their personal counterparts according to each user special need. It is designed to produce webpages that are in conformance with the WCAG of WAIof the W3C and with the Section 508 of the Federal Information Technology Accessibility Initiative. Accessibility transformations are performed both at the physical level of interaction (e.g., fonts and colours), and at the syntactic level of interaction (i.e., re-structuring) of web documents with supporting some supplementary features such as: (a) linearized page rendering, in a top-down manner, (b) removing tables used for page layout, (c) “tab”

[23] http://www.soundlinks.com/pwgen.htm

browsing support based on a tab index that is assigned to every active element of the page (d) HTML anchor elements which are added to demarcate specific sections of the HTML document and associated with a specific access key that enables users to jump to these sections with a single key press.

Web Adaption Technology[24] developed by IBM Research is investigating a method of making Web pages accessible without requiring the use of assistive technologies. Through a standard browser, users will have the ability to access Web pages reformatted in a manner most usable by them. Web Adaptation Technology transforms the results of HTTP requests using a combination of both client and server technologies, taking advantage of the major strengths of each. Key features that are supported by the aforementioned technology include enlarging of content on web pages, tailoring the text to meet user needs, reducing visual clutter on web pages by removing backgrounds, stopping animations, not displaying images and by reformatting the page layout so that content is presented in a single column. Keyboard and mouse adaptations are supported too.

Several other intermediary frameworks are specifically designed for people with vision impairments and focus on removing sticky user interface elements or on transforming them to exploitable elements by assistive technologies. More specifically, these frameworks transform a web page from a graphics-heavy and inaccessible version, to a text-only version that is easily accessible by visually impaired users. By following the W3C's Web Content Accessibility Guidelines, they remove elements that screen readers cannot handle, allows user customization of fonts and background colours, remove Java applets, javascript code, and graphics, fix ALT attributes, replace shockwave, flash, and other plug-in applications etc. More specifically, the aforementioned systems handle frames, by serializing them and tables by linearising them. Some of the most known systems that fall into category are the *Personalizable Accessible Navigation* [23], the *Access Gateway* system [24], the *Textualise* system[25], the *Accessibility Transformation Gateway* [25], the *IBM system* described in [26], *BETSIE*[26], *Crunch* [27, 28], *Muffin*[27] and *RabbIT*[28].

Web-Based Intermediaries (WBI) [29, 30, 31], is a special dynamic framework that includes a Text-To-Speech service [32] that allows the speaking of the text of HTML pages during their displaying towards end users. Moreover, the Text-To-Speech Service can help the comprehension of documents that are written in foreign languages, since a reader that is partially familiar with the spoken language and not with its written form can be supported in getting, at least, the meaning of the information contained in the document.

Web Page Transformation [33] is another special framework that offers a webpage transformation algorithm to browse webpages on small devices that consists of three steps. It analyses the HTML Document Object Model (DOM) tree and detects the high-level content blocks (which contain location and size information for headers, footers, sidebars, and the body). Then, it analyses the content inside each high-level content block to identify explicit separators to determine where to split the blocks. Finally, it detects implicit separators to help split the blocks further. The overall goal for page analysis is to help split webpages into appropriate blocks so that

[24] http://www.webadapt.org/

[25] http://aquinas.venus.co.uk/

[26] http://www.bbc.co.uk/education/betsie/

[27] http://muffin.doit.org

[28] http://rabbit-proxy.sourceforge.net

users can browse page blocks on small devices instead of scrolling and browsing through large webpage.

Concluding the concept of intermediary agents is considered as a very promising approach for enabling universal accessibility on the Web. The concept of maintaining a proxy that can dynamically retrieve and "fix" a Web document in order to make it accessible was followed by a number of approaches discussed previously. However, the practical exploitation of this concept highlighted a number of issues that tend to reduce the universality of the approach. Many websites do not produce html code for each element appearing in a webpage. There are examples of websites were the rendered code is entirely in a client scripting language. Additionally, malformed html code tends to make these websites not readable by proxies (due to the issues arising in the process of parsing malformed documents). A solution to this issue was the development of specialised filters for each website or portal parsed by an intermediary agent. This approach enabled these agents to interpret a number of different technologies and web programming approaches, but at the same time made it clear that the current situation of the web does not allow a universal approach. Finally the diversity of the target user population cannot be address by just "fixing" the rendered html output. Sometimes there is a need to perform additional operations such as replacement of interaction elements and modalities in order to cope with a diverse target user population. This is clearly not supported by these systems, and therefore these solutions can only be considered as frameworks that automatically test and fix issues of compliance with the web content accessibility guidelines.

2.3.3 Self-adapting Web-based systems

Except from the intermediate agents that were presented above there are other web systems or algorithms that encapsulate or support adaptation correspondingly.

E-Victor [34] is an eCommerce system that has been developed as a component based application. Components are grouped into services that offer functionality to application users (user services) or to other services (internal service). A separation of concerns has been realised by decomposing each service into separate components for content, layout, navigation, user-interaction and processing. Both the selection of services and the type of content, layout, navigation and user-interaction component used by each service are adapted within E-Victor. During the adaptation process for each user a set of service components is selected from the set of available service components applying the ranking relations derived from the user profiles. For each service components, navigation, layout and user-interaction components are selected according to availability and ranking. Finally the selected components are instantiated. E-Victor system is a custom system that supports alternative navigation, layout and user interaction components using WebComposition Markup Language. Although this approach has important results, the use of a specific markup language reduces the advantage to be used as a general solution for web development. Additionally, this approach focuses on a specific system and doesn't present a generic framework that would be possible to be applied for the majority of web sites or sites developed with a specific technology. Finally, E-Victor doesn't support accessibility adaptations, embarrassing users with disabilities to access it.

In [35], a finite state machine algorithm was proposed, which gathers information on the clicking styles of the user and categorises them into predefined profiles during progress through the task. For example, a user consistently clicking on the hypertext would be classified as a text profile; whereas mixing text, images and

speech in any order would spawn a multimedia profile. A default profile gets weighted by the modality used at each step of the navigation. Finally, the system applies predefined presentation templates for every step of the process, progressively adapting to the interaction style of the user. A visual profile (using mainly images) would receive a shorter description with many images, while a text profile would get only one image and a longer text. A multimedia profile would get a video description, a speech profile a spoken description and an iconic profile a digest style including bullet points and iconic information. This approach uses statistical values in order to choose among predefined patterns of interaction styles that fit to specific user interaction styles. Although this algorithm can derive conclusions about user presentation preferences (e.g., image style, text style) it can not infer interaction preferences. Additionally, this method supports only alternative information representation and does not offer the means to be extended to support alternative UI elements, dialogue controls, etc.

Chapter 3
METHODOLOGY

In this Chapter, the rationale, principles and the methodology followed during the design and implementation of proposed UWI framework and the derived EAGER toolkit are described. In more detail, this chapter emphasises to the principles of *User Interfaces for All*, as well as to the *Unified User Interface Design* method, adapted here for the Web. Subsequently, implementation issues of EAGER are presented. The approach followed to validate the presented framework and toolkit is also described.

3.1 User Interfaces for All

The term *User Interfaces for All* denotes an effort to unfold and reveal the challenges of usability and accessibility, as well as to provide insights and instrument appropriate solutions in the HCI field. The fundamental vision of *User Interfaces for All* is to offer an approach for developing computational environments that cater for the broadest possible range of human abilities, skills, requirements and preferences. The roots of *User Interfaces for All* are traced in the notions of *Design for All* and *Universal Access* [36].

Design for All in the Information Society is the conscious and systematic effort to proactively apply principles and methods, and employ appropriate tools, in order to develop IT & T products and services which are accessible and usable by all citizens thus avoiding the need for posteriori adaptations, or specialised design [5]. Supplementary *Universal Access* to computer-based applications and services implies more than direct access or access through add-on (assistive) technologies, since it emphasizes the principle that accessibility should be a design concern, as opposed to an afterthought. *Universal Design* and *Design for All* in *HCI* recognises, respects, values and attempts to accommodate the broadest possible range of human abilities, requirements and preferences, eliminates the need for 'special features' and fosters individualisation and end-user acceptability.

User Interfaces for All fosters a proactive strategy, postulating that accessibility and quality of interaction need to be embedded into a product at design time. Proactive strategies entail a purposeful effort to build access features into a product, as early as possible (e.g., from its conception, to design and release). In the context of HCI, *User Interfaces for All* advocates such a proactive paradigm for the development of systems accommodating the broadest possible end-user population [37]. In other words, *User Interfaces for All* seeks to minimize the need for a posteriori adaptations and deliver products that can be adapted for use by the widest possible end user population (adaptable user interfaces). This implies the provision of alternative interface manifestations depending on the abilities, requirements and preferences of the target user groups. The main objective in such a context is to ensure that each end-user is provided with the most appropriate interactive experience at run-time. Producing and enumerating distinct interface designs through the conduct of multiple design processes would be an impractical solution, since the overall cost for managing in parallel such a large number of independent design processes, and for separately implementing each interface version, would be unacceptable [38].

Therefore, the *Unified User Interface Development* (U^2I) methodology, which is part of the Unified User Interface development framework, supports a process that leads to a single system that appropriately structures multiple designs and their underlying user and context related parameters, therefore facilitating on the one hand the mapping of design to a target software system implementation; and on the other hand the maintenance, updating and extension of design itself. The U^2I method is followed here and adapted appropriately for the Web. For this reason, in this work we refer to this adapted approach as *Unified Web-based Interfaces* (UWI).

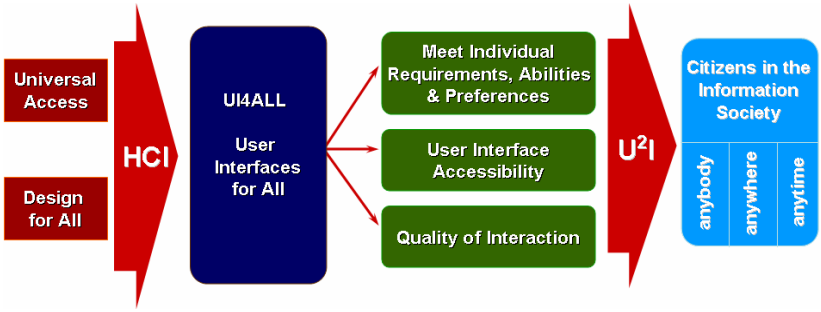


Figure 2: **The concept of User Interfaces for All**

3.2 Unified User Interfaces

Until recently, adaptive techniques have had limited impact on the issue of Universal Access. The U^2I development methodology [3, 39,40,41,42,43] has been proposed as a complete technological solution for supporting universal access of interactive applications and services. U^2Is convey a new perspective into the development of user interfaces, providing a principled and systematic approach towards coping with diversity in the target user requirements, tasks and environments of use [44]. The notion of a U^2I originated from research efforts aiming to address the issues of accessibility and interaction quality for people with disabilities [45]. A U^2I comprises a single (unified) interface specification that exhibits the following properties [39]:

(i) It embeds representation schemes for user- and usage-context- parameters and accesses user- and usage-context- information resources (e.g., repositories, servers), to extract or update such information.

(ii) It is equipped with alternative implemented dialogue patterns (i.e., implemented dialogue artefacts) appropriately associated to different combinations of values for user- and usage-context- related parameters. The need for such alternative dialogue patterns is identified during the design process, when, given a particular design context, for differing user- and usage-context- attribute values, alternative design artefacts are deemed as necessary to accomplish optimal interaction.

(iii) It embeds design logic and decision making capabilities that support activating, at run-time, the most appropriate dialogue patterns according to particular instances of user- and usage-context- parameters, and is capable of interaction monitoring to detect changes in parameters.

As a consequence, a unified interface realises:

- User-adapted behaviour (user awareness), i.e., the interface is capable of automatically selecting interaction patterns appropriate to the particular user.

- Usage-context adapted behaviour (usage context awareness), i.e., the interface is capable of automatically selecting interaction patterns appropriate to the particular physical and technological environment.

From a user perspective, a U^2I can be considered as an interface tailored to personal attributes and to the particular context of use, while from the designer perspective it can be seen as an interface design populated with alternative designs, each alternative addressing specific user- and usage-context- parameter values. Finally, in an engineering perspective, a U^2I is a repository of implemented dialogue artefacts, from which the most appropriate according to the specific task context are selected at run-time by means of an adaptation logic supporting decision-making [39].

At run-time, the adaptations may be of two types:

a) adaptations driven from initial user- and context- information known prior to the initiation of interaction, and
b) adaptations driven by information acquired through interaction monitoring analysis.

The former behaviour is referred to as *adaptability* (i.e., initial automatic adaptation) reflecting the interface's capability to automatically tailor itself initially to each individual end-user in a particular context. The latter behaviour is referred to as *adaptivity* (i.e., continuous automatic adaptation), and characterizes the interface's capability to cope with the dynamically changing or evolving user and context characteristics. Adaptability is crucial to ensure accessibility, since it is essential to provide, before initiation of interaction, a fully accessible interface instance to each individual end-user [3]. Furthermore, adaptivity can be applied only on accessible running interface instances (i.e., ones with which the user is capable of performing interaction), since interaction monitoring is required for the identification of changing / emerging decision parameters that may drive dynamic interface enhancements. This combination of adaptation characteristics and behaviour make U^2Is suitable and appropriate for supporting Universal Access [3].

The concept of U^2I is supported by a specifically developed architecture [46]. This architecture consists of independent communicating components, possibly implemented with different software methods and tools that cooperate to perform the types of adaptation mentioned above.

Figure 3 (from [46]) depicts the components of the unified interface architecture.

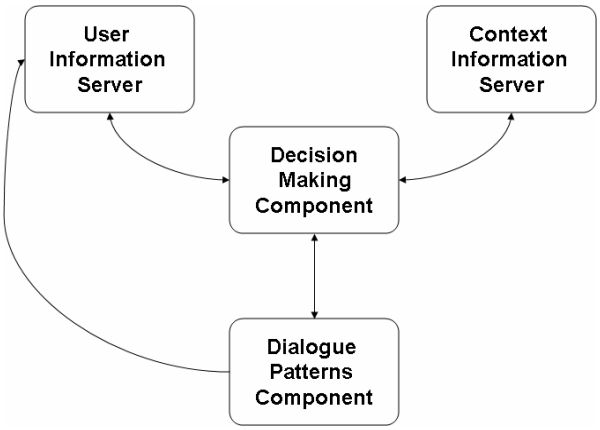


Figure 3: **Overview of the U²I Architecture [46]**

The **Dialogue Patterns Component** (DPC) is responsible for supplying the software implementation of the dialogue artefacts identified in the design process as belonging to a unified interface. Such components may be common across different user- and usage-context- attribute values, in the case no adaptation is needed, or dialogue artefacts that are appropriate for specific attribute values, in case alternative designs for adaptation have been identified.

The **Decision Making Component** (DMC) has the role of deciding at run-time the necessary automatic adaptation actions, and to subsequently communicate such decisions to the DPC, which applies them.

The **User Information Server** (UIS) supplies user attribute values either known off-line, without performing interaction monitoring analysis (e.g., motor / sensory abilities, age, nationality, etc., or detected on-line, from real-time interaction-monitoring analysis (e.g., fatigue, loss of orientation, inability to perform the task, interaction preferences, etc.).

The **Context Information Server** (CIS) supplies context attribute values (machine and environment) either invariant, i.e., unlikely to change during interaction (e.g., peripheral equipment), or dynamically changing during interaction (e.g., environment noise, failure of particular equipment, etc.). This component does not support device independence, but device awareness, and enables the Decision Making Component to select those interaction patterns, which, apart from fitting the particular end-user attributes, are also appropriate for the type of equipment available to the end-user.

The U^2I Development method is not prescriptive regarding how each component is to be implemented [46]. For example, the UIS component may employ alternative ways of representing user-oriented information. A repository of user profiles can serve as a central database of individual user information (i.e., registry). A quite simple but powerful and flexible approach is to represent profiles as a typical list of attributes and related values. More sophisticated user representation and modelling methods can be also employed, including support for stereotypes of particular user categories.

3.3 Proposed implementation approach

In this section, the implementation architecture followed to support the development of UWIs, including the prototype portal presented in Chapter 7 is discussed. This architecture, called three- or multi-tier architecture, separates the application logic from data, introducing an additional distinction of responsibilities at the back-end side. The presence of one or more distinct application tiers enables the implementation of advanced architectures that integrates the traditional HTTP protocol and client-server application distribution protocols for better performance, scalability, reliability, and security [51]. The advantages of a multi-tier architecture mainly derive from the separation of the various independent parts of a system. The software architecture used includes three basic layers: *Data Access Layer*, *Application (business) Layer* and *Presentation Layer* as shown in Figure 4.

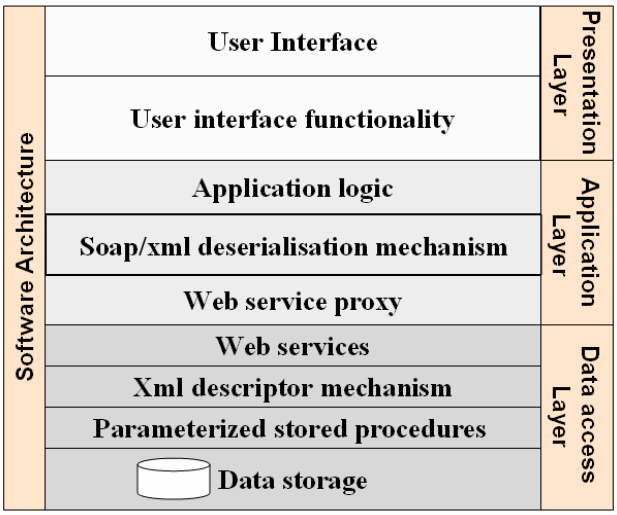


Figure 4: **Proposed software architecture for implementing UWIs**

The Database implementation supports multilingualism both in terms of user interface and application content using a design method for the separation of multilingual and non multilingual content through the division of each table in two separate tables, one containing the non multilingual content and the other containing the multilingual one. Additionally, the database stored procedures are incorporated for faster retrieval and insertion in the database, reducing the amount of client side processing. These stored procedures are accessed by web services that are subsequently available to the business logic layer. The implementation of these web services incorporates an *XML* (eXtensible Markup Language) query descriptor mechanism [52] that undertakes to connect to the database management system, call the appropriate predefined query using the available query description, pass the suitable parameters and return the acquired data. The objective of the XML query descriptor library module is to provide a consistent mechanism for accessing the underlying sub-layers using a unique and secure way of connecting to the database. Apart from the connection string, this module requires all the appropriate information to call a specific parameterized predefined query from the database. For achieving library reusability and database independence, an XML query description file is used to retrieve such information. The aforementioned Web services are responsible for providing the appropriate mechanism to transmit data from the data access layer to the application (business) layer. Web services in general provide several advantages, since they are loosely coupled to the clients and stateless serving thus several clients at the same time, and allowing immediate switching between servers [53].

The Business logic layer (application) incorporates classes that form a higher level ontology specification of the database schema. The aim of this layer is to transform the data received by the web services of the data access layer to instances of the ontology specification. To this purpose, special methods are used to deserialise the data received and transform them into meaningful instances of the ontology. This strategy is followed in order to make the development of the higher levels easier and closely coupled with the UI functionality. The implementation of the business logic layer is totally independent from the implementation of specific parts of the data access layer, and allows the replacement of the data access layer without redeveloping the business layer. The presentation layer is developed using UI components that provide the ability to each UI module to dynamically adapt to the requirements set by the basic and extended characteristics of the User.

Chapter 4
TOWARDS UNIFIED WEB-BASED INTERFACES

4.1 Adaptive and Adaptable behaviour

Web users are potentially all citizens. In this context, WUI adaptations must take into account a wider collection of parameters, such as context and user specific attributes (e.g., input-out devices, disabilities, user attitude towards technology, etc.). The adaptation of an application can occur in different ways and can cover a number of aspects of the application or its environment

The adaptation of an application can be classified according to which aspects of the application are adapted. Generally, Web application model described in [58] distinguishes five orthogonal aspects of an application:

1. **content:** Adaptation of content affects the content such as text, graphics or any other media type or data used or displayed by the application. This type of adaptation is most common on the Web. UWI supports adaptations, which automatically modify the presentation and conceived behavioural attributes of interactive elements. As an example images can be transformed as normal images, as simple text containing image's alternative text and as a hyperlink that downloads the image and has as text the image's alternative text.
2. **navigation:** Adaptation of navigation adapts the navigational structure of a web application hiding or modifying links. UWI supports navigation adaptations. Some examples include the linearization of the whole navigation of the portal in a top navigation bar in order to facilitate blind users or the step by step navigation which reduces the number of links that motor-impaired user has to scan.
3. **layout:** Adaptation of layout changes the way information is presented to a user visually. This can be done to accommodate different types of displays or to satisfy preferences of aesthetic, cultural or other nature a user may have. The proposed framework supports layout adaptations, as long as it offers alternative templates layouts depending on screen resolution, disability etc.
4. **user-interaction:** Adaptation of user-interaction changes the way the user interacts with the application. An application might adapt offering a wizard based interface to less experienced users and a single page form to other users. UWI framework supports conditional activation and deactivation of multiple interaction modalities based on the user profile including alternative task structures, alternative syntactic paradigms, task simplification and adaptable and adaptive help facilities - runtime task guidance.
5. **processing:** Adaptation of processing changes the way user-input is processed. For example, a product request of a person that has placed many large orders in the past might be processed differently than that of a previously unknown person. UWI framework doesn't support such adaptations. These kinds of adaptations can not be address by a generic framework and rely solely on the implementation of each web application.

On the other hand, the adaptation of an application can also be classified according to the way that is achieved in three distinct categories:

- Manual adaptation can either take place during the development cycle of an application where a developer can modify the application to meet different user requirements or run on a different platform. It can also take place through manual reconfiguration of an application. UWI framework supports this kind of adaptation by providing user with a mechanism in order to extend his/her generic profile to a more extended version. In this version user selects the exact adjustments of the Web application.
- Automated adaptation does not require any effort on the part of a user, administrator or developer. It is done automatically by the application or the framework the application runs in and can take place in regular time intervals or event driven. The proposed framework supports such a kind of adaptation, as an example based on statistic values identify situations that user can not complete a task and activates help provision component.
- Semi-automated adaptation requires the user or another person to provide certain information through questionnaires, feedback buttons, etc., to be able to adapt the application. The information obtained does not directly describe how the application should be adapted but is used by the system to determine a suitable application configuration. UWI supports this kind of adaptation by providing user with a generic profile where stores general information such as disability, language, etc. It also supports decision logic in order to decide about the adjustments of the web application based on the general information.

The next sections provide a detailed discussion of the proposed architecture for design and implement UWI.

4.2 The Unified Web-based User Interfaces (UWI) methodology

The UWI methodology is derived from the aforementioned architectural structure for enabling the development of U^2I. Towards this direction, several adaptations were made to apply this architecture in the context of web applications. Figure 5 presents a general overview of the UWI architecture followed and the engaged communication channels. The basic components involved in the architecture are:

- **The User Information Component:** responsible for collecting and propagating user specific attributes.
- **The Context Information Component**: responsible for collecting and propagating attributes varying by the context of use.
- **The Decision Making Component:** in charge of the overall decision making regarding the conditional activation – deactivation of interaction elements.
- **The Designs Repository:** a repository of the alternative interaction styles design to be implemented in the Dialogue Controls Component.
- **The Dialogue Controls Component:** a repository of alternative interaction styles to be conditionally activated of deactivated to form the final User Interface.

The engaged communications channels are used for propagating user and context specific parameters from the User and Context Information Component to the Decision Making Component. On the other hand, the Decision Making Component is responsible for propagating its decisions to the Dialogue Controls Component in order to inform about the activation – deactivation of interaction elements. The aforementioned communication channels are bidirectional in order for the Decision Making Component to be able to propagate decisions that may result to changes on the user or context specific parameters. The overall process results in the activation of the appropriate interaction elements to be used for rendering the final interactive front-end.

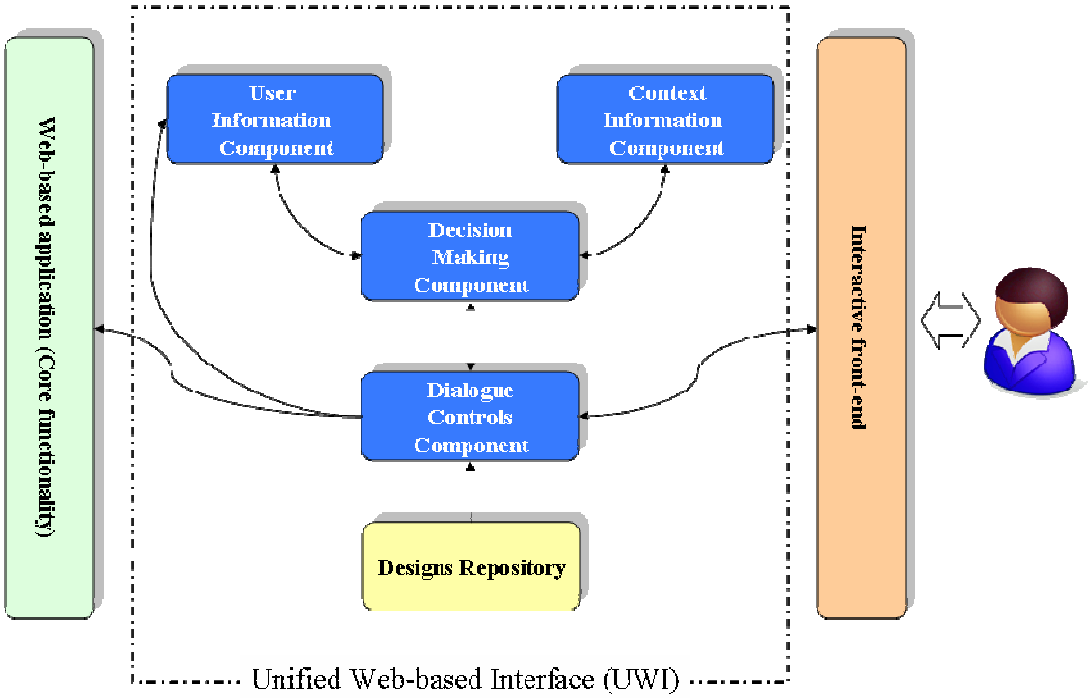


Figure 5: **Architecture of Unified Web-based Interfaces (UWIs)**

Each part of the presented architecture is described in detail in the following sections in order to clarify its role in the adaptation process and its internal behaviour.

4.2.1 User Information Component

The scope of User Information Component (UIC) is to provide information regarding user profiles. A User profile initially contains attributes specified by the user prior to the initiation of interaction or during interaction. In order to collect user attributes during interaction, a specific mechanism is used inside the UIC.

In Figure 6 the detailed UIC architecture is presented. As shown in this figure, the User Profiles Repository is a database containing all the user specific parameters. On the other hand the User Components Repository collects information regarding the conditional activation- deactivation of interactive elements per user as propagated by the Decision Making Component. To achieve the transmission of data from and forth the various modules to the User Profiles and Components repositories specialised Web Services and Logic is incorporated acting as a proxy class to their underlying implementation. The Interaction Monitoring Module incorporated in the UIC

architecture provides the appropriate mechanisms to monitor and record user interactions and propagate data related with the interaction to the User Profiles Repository. The data propagated by this module include the successful or unsuccessful completion of actions, the popularity of navigation options, etc.

The core element of the UIC is the Profiling Module which is responsible for propagating User Profile information to the DMC, and additionally acts as a front end providing all the required functionality to be used by the specialised Profiling UI modules.

These modules include:

- **User Profiles Statistics UI module:** Presents statistics regarding the popularity of the available settings.
- **User Profile Selection UI module:** Allows the selection among predefined user profiles or the manual configuration of the available settings.
- **User Profiles Administration UI module:** Allows the administration of the predefined profiles.

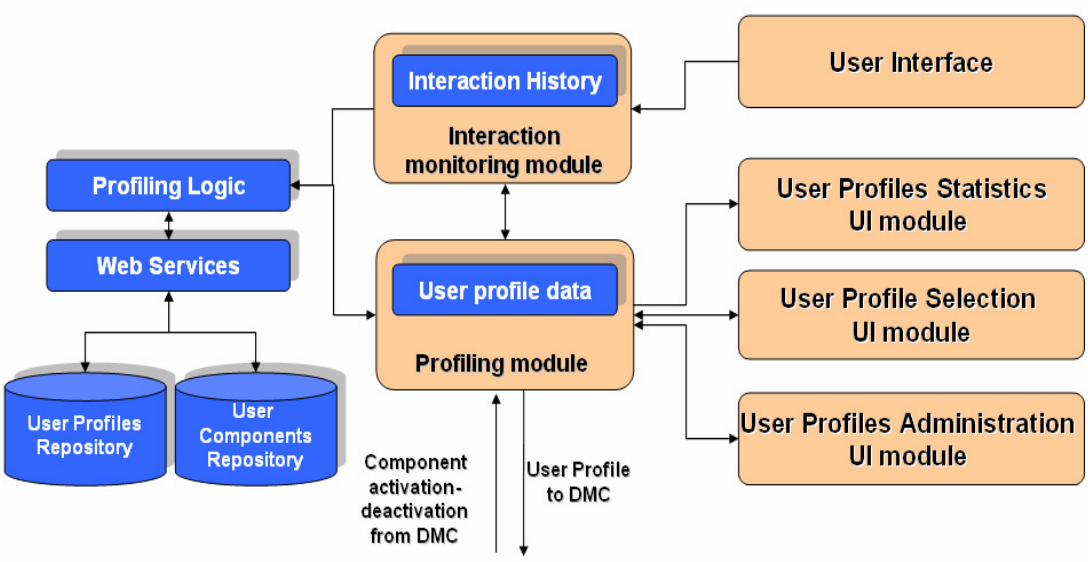


Figure 6: **User Information Component architectural model**

Figure 7 depicts an example of the attribute value based user profile model. Similar considerations hold for the Context Information Component presented in the next section.

Parameters	Values				
Language	English	Greek			
Disability	None	Blind	Low vision	Color-blind	Motor-impaired
Web familiarity	Low	Moderate	High		

Figure 7: **Example of User Profile Instance**

4.2.2 Context Information Component

The Context Information Server of the derived architecture was initially intended to collect and propagate context attribute values (machine and environment) of two types:

(i) (potentially) invariant, meaning unlikely to change during interaction, e.g., peripheral equipment; and

(ii) (ii) variant, dynamically changing during interaction (e.g., due to environment noise, or the failure of particular equipment, etc.).

This component was therefore intended to support device awareness. In the context of web application, the attributes to be supported dynamically by this component are radically decreased due to the lack of methods for capturing and propagating such information from the client side.

Despite of the aforementioned limitations, the role of this module in the extended architecture remains significant and its architectures is presented in Figure 8. As shown in this figure, the CIC shares functionality with the UIC. The profiling module is also used to collect and propagate context specific parameters to different subsystems. The Context Monitoring Module presented in this architecture has the responsibility to monitor context changes and propagate this information to the User Profiling Module. This module in turn enriches the User Context Profile Repository with these context specific attributes to be used in the process of Decision Making.

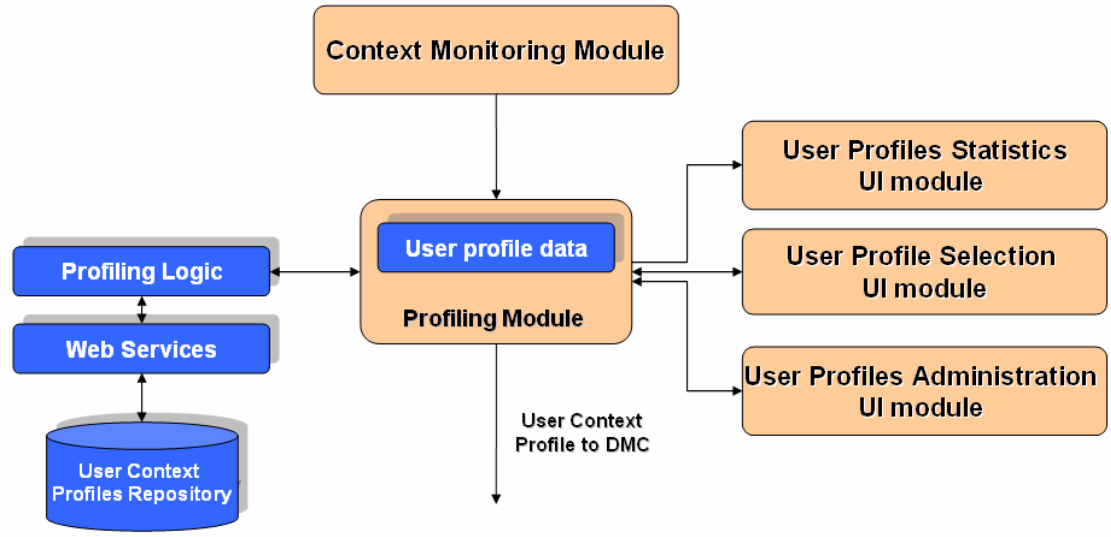


Figure 8: **Context Information Component architectural model**

In case dynamic user attribute detection is desired, the content may include dynamically collected interaction monitoring information, design information and knowledge processing components.

Figure 7 depicts an example of the attribute value based user profile model and instance of the EAGER toolkit; similar considerations hold for the CIS.

4.2.3 Decision Making Component

The Decision Making Component (DMC – see Figure 9) decides regarding the conditional activation and deactivation of UI components based on the user attributes provided by the UIC and the repository of alternative dialog patterns. The core of this component consists of a number of implemented rules representing the design space of the user interface. More specifically, this module performs the overall decision making of when, why and how adaptation will occur. The adaptation logic incorporated in this module contains the mapping of user attributes to selections among alternative dialog patterns.

For example, the decision logic for presenting links can be the following:

- "web knowledge" in {very good, good} → "underlined text"
- "web knowledge in {normal, low} → "push buttons"

Below there is an example of the decision login of that can be implemented based on "if…then…else" logic.

Decision making logic – an excerpt

```
switch(disability)
{
      case (long) Disablity.Blind:
     {
              return DisplayStatistics.Table;
     }
    case (long) Disablity.ColourBlindProtanope:
    {
       return DisplayStatistics.GraphicalProtanope;
    }
   case (long) Disablity.ColourBlindDeuteranope:
   {
       return DisplayStatistics.GraphicalDeuteranope;
   }
   case (long) Disablity.ColourBlindTritanope:
   {
       return DisplayStatistics.GraphicalTritanope;
   }
   case (long) Disablity.MotorImpaired:
   {
       return DisplayStatistics.Graphical;
   }
   case (long) Disablity.LowVision:
   {
       return DisplayStatistics.Table;
   }
   case (long) Disablity.None:
   default:
   {
              return DisplayStatistics.Graphical;
    }
}
```

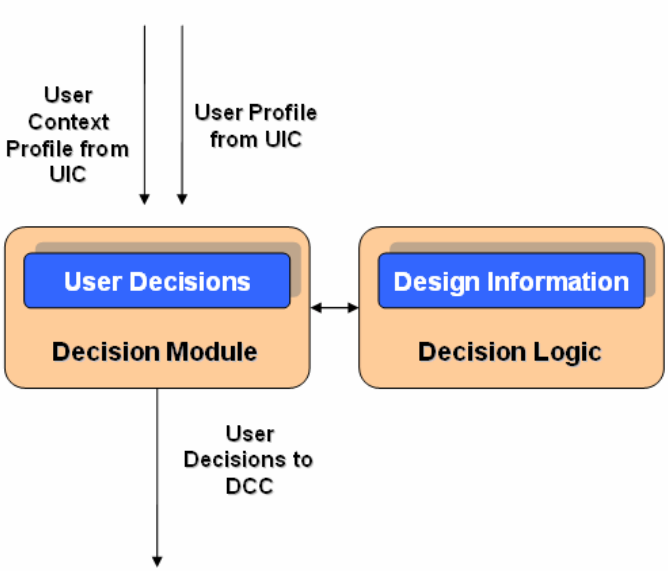


Figure 9: **Decision Making Component architectural model**

4.2.4 Dialogue Controls Component

The role of the Dialogue Controls Component (DCC) is to apply interface adaptation decisions propagated by the DMC and additionally to structure the final interface using the selected dialog components. More specifically, this component:

(i) provides the implementation of the alternative dialog components of a self-adapting interface in the form of dynamic libraries
(ii) moderates and administrates the alternative dialog components
(iii) maintains a record of user interaction with alternative dialog components (see Figure 10).

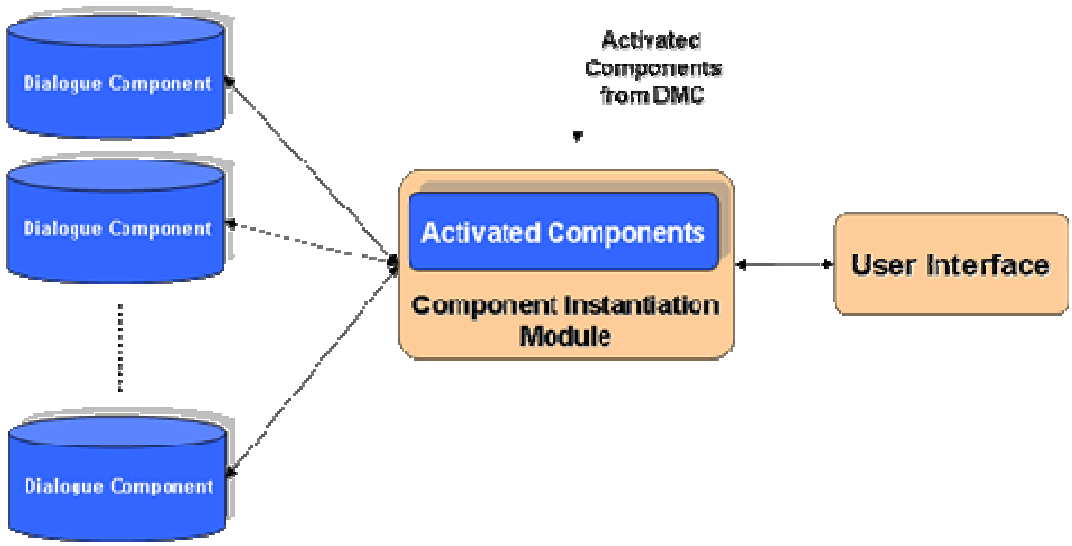


Figure 10: **Dialogue Controls Component architectural model**

4.2.5 Designs Repository

The Design Repository component is developed during the design cycle and provides the Dialogue Controls Component with the designs of the alternative dialogues controls in a form of abstract design and polymorphism. Polymorphic decomposition leads from abstract design pattern to a concrete artefact. Three categories of design artefacts may be subject to polymorphism on the basis of user- and usage-context-parameter values, and namely:

- **User tasks**, relating to what the user has to do; user tasks are the centre of the polymorphic task decomposition process.
- **System tasks**, representing what the system has to do, or how it responds to particular user actions (e.g., feedback); in the polymorphic task decomposition process, they are treated in the same manner as user tasks.
- **Physical designs**, which concern the UI components on which user actions are to be performed; physical interface structure may also be subject to polymorphism.

User tasks, and in certain cases, system tasks, are not necessarily related to physical interaction, but may represent abstraction on either user- or system- actions. System tasks and user tasks may be freely combined within task "formulas", defining how sequences of user-initiated actions and system-driven actions interrelate. The physical design, providing the interaction context, is associated with a particular user task, and provides the physical dialogue pattern associated to a task-structure definition. Hence, it plays the role of annotating the task hierarchy with physical design information. The U^2I design emphasises capturing of the more abstract structures and patterns inherent in the interface design, enabling hierarchical incremental specialisation towards the lower physical-level of interaction, and making therefore possible to introduce design alternatives as close as possible to physical design [41]. This makes it easier to update and extend the design space, since modifications due to the consideration of additional values of design parameters (e.g., considering new user- and usage-context- attribute values) can be applied locally to the lower-levels of the design, without affecting the rest of the design space.

Figure 11 depicts an example of polymorphic task hierarchy, illustrating how two alternative dialogue styles for an "upload file" task may be designed. Alternative decomposition "styles" are depicted in the upper part of the figure, and an exemplary polymorphic decomposition at the lower part. Figure 12 includes physical design annotation corresponding to the alternative styles.

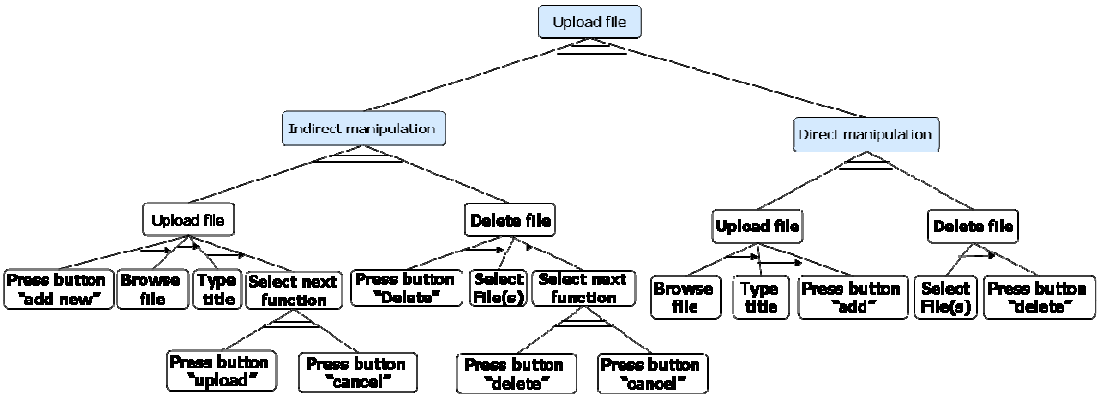


Figure 11: **The polymorphic task hierarchy concept**

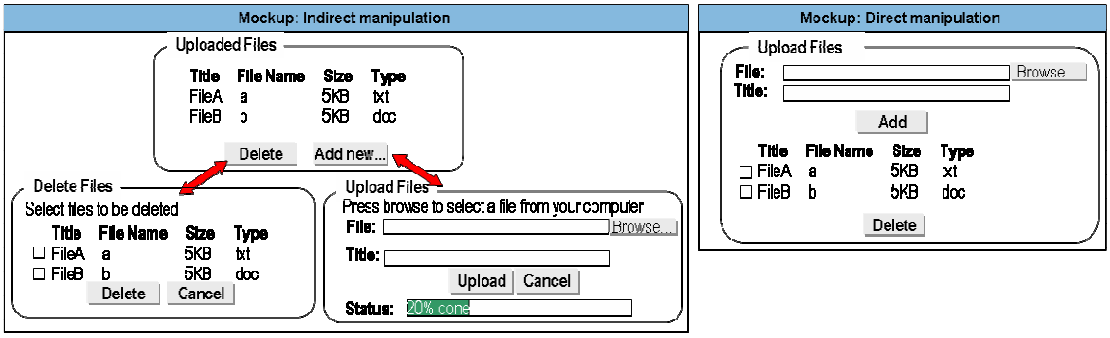


Figure 12: **Physical design alternatives (for *File uploader*)**

In the depicted process, the following primary decisions need to be taken:

- at which points of a task hierarchy polymorphism should be applied, based on the considered (combinations of) user- and usage-context- attributes; and
- how different styles behave at run-time. This is performed by assigning to pair(s) of style (groups) design relationships.

These decisions need to be documented in a design rationale recorded by capturing, for each sub-task in a polymorphic hierarchy, the underlying design logic, which directly associates user- / usage-context- parameter values with the designed artefacts.

As a minimum requirement, such a rationale should document [38]:

- related task
- design targets leading to the introduction of the style
- supported execution context
- style properties
- design relationships with competing styles.

In Table 3, an instance of such documentation record for the task "upload files" is depicted, adopting a tabular notation.

Table 3: **An instance of design rationale documentation**

Task: Upload files		
Style:	Indirect manipulation	Direct manipulation
Targets:	Simplicity, guided steps	Speed, effectiveness
Parameters:	User (novice)	User (expert)
Properties:	Upload file: Press button 'add new' first, browse file next, type title next, press button 'upload' Delete file: Press button 'delete' first, select file(s) next, press button 'delete'	Upload file: Browse file first, type title next, press button 'add Delete file: Select file(s) first, press button 'delete' next
Relationships:	Exclusive	Exclusive

4.3 Extensibility

Moreover, one of the most prominent features of UWI framework is its ability to extend, thus promoting code reusability and development efficiency. Towards this direction, the process of extending the framework to support new interaction elements entails the process of designing and coding the alternative interactions styles. The new interaction elements can be in turn easily incorporated in the framework by adding the required decision logic for supporting their conditional activation and deactivation. Additionally, the framework can be extended for supporting new interaction modalities and adjustments. For example, in order for the framework to be able to produce high quality results in a PDA, the first step to be followed is the evaluation of the existing interaction element in order to decide whether their output is sufficient for use with PDA devices. In the case this does not occur, new styles must be produced to facilitate the interaction and display requirement of a PDA. Additionally, whenever new functionality is required, the process for extending the framework to support new interaction elements must be applied. The overall process of decision making does not alter. The framework continues to operate with enriched decision logic applied whenever the context of use parameters specifies that a PDA device is used.

Chapter 5
EAGER: A TOOLKIT FOR WEB DEVELOPERS

This Chapter describes the implementation of the *EAGER toolkit* developed in Microsoft® Visual C# .NET and according to the UWI framework presented in previous Chapter.

The technologies that were used for the development of the EAGER toolkit include:

- EAGER toolkit together with Microsoft Visual C# .NET for the implementation of the UI modules.
- Microsoft Visual C# .NET together with XML for Business Logic and Web Services.
- Microsoft SQL server 2000 for the database implementation.

An overview of the code details of the EAGER toolkit are presented in Figure 13. Note that a 15% of the total number of lines constitutes comments specially targeted to Web developers.

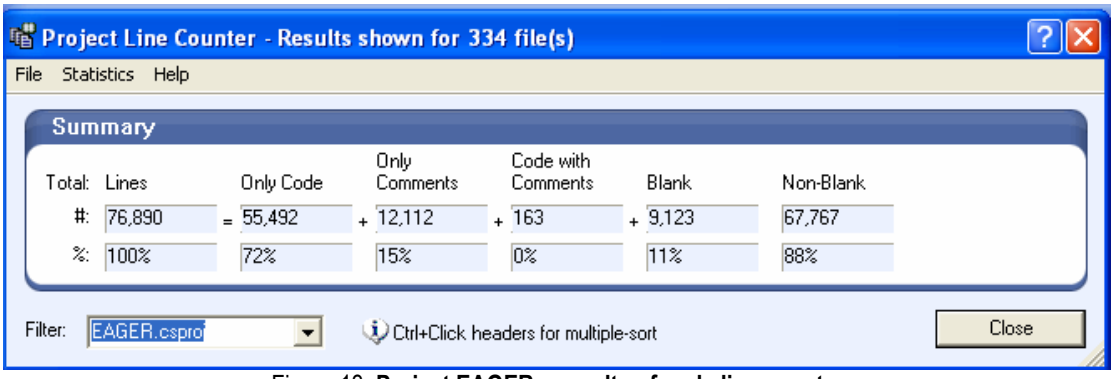


Figure 13: **Project EAGER – results of code line counter**

For each of the UWI components described in the previous Chapter, the corresponding elements of EAGER are presented in the following subsections.

5.1 User Information Component (UIC) implementation

As described in the previous Chapter, UIC consists of the following modules: User Profile Repository, User Components Repository, Profiling Module, Interaction Monitoring Module, User Profiles Statistics UI Module, User Profile Selection UI Module, and User Profiles Administration UI Module (see Figure 6). Their implementation with EAGER is presented in detail in the following subsections.

5.1.1 User Profiles Repository

The User Profiles Repository stores information about basic user attributes as presented in Figure 14. These attributes (Language, Disability, Web familiarity) can be

thought as the minimum level of information to be used for performing decision making by the EAGER toolkit.

Parameters	Values				
Language	English	Greek			
Disability	None	Blind	Low vision	Color-blind	Motor-impaired
Web familiarity	Low	Moderate	High		

Figure 14: **User profile basic attributes specified prior to initiation of interaction**

The Database architecture used for presenting the user information server is presented in Figure 15.

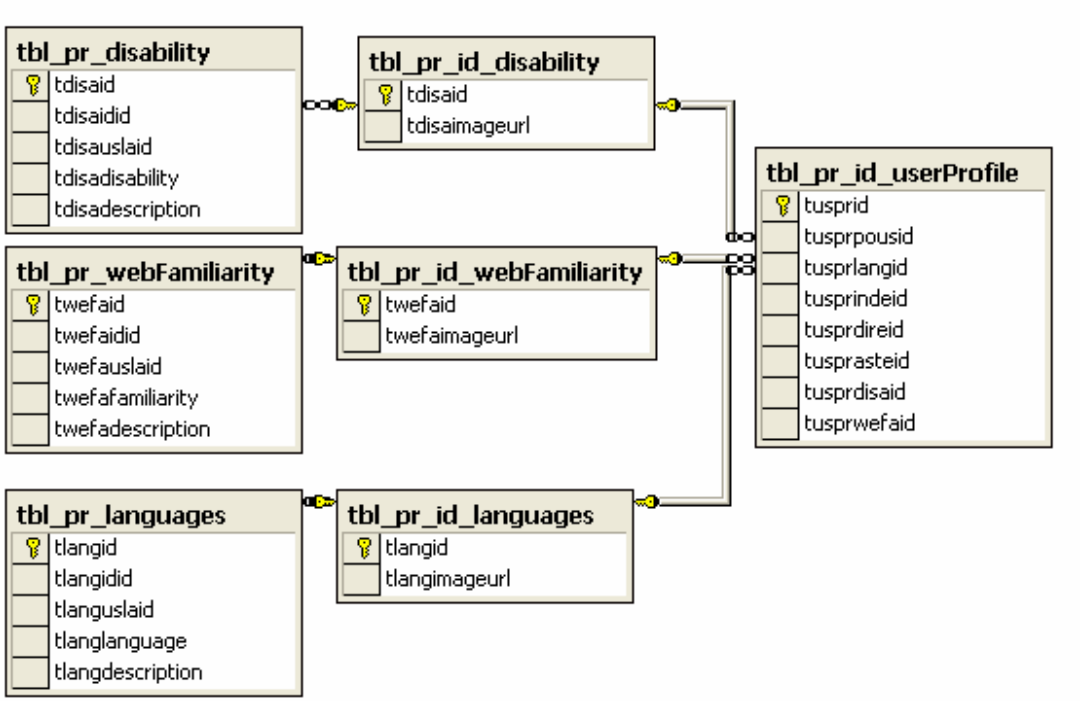


Figure 15: **The Database diagram of the table to hold the basic user attributes**

5.1.2 User Components Repository

As already discussed, the DMC propagates the decisions made regarding the conditional activation-deactivation of interaction elements to the UIC. Towards this direction, a more detailed scheme to present the specialized user selections regarding the conditional activation – deactivation of interaction elements is required. In this scheme, the decisions made are stored in order to prevent the per request invocation of the DMC. Additionally, each user is provided with the ability to manually override the default decision making and alter its personal setup for enriching its profile based on his personal interaction and accessibility preferences. The abstract representation of the extended user attributes is presented in Figure 16 to Figure 18.

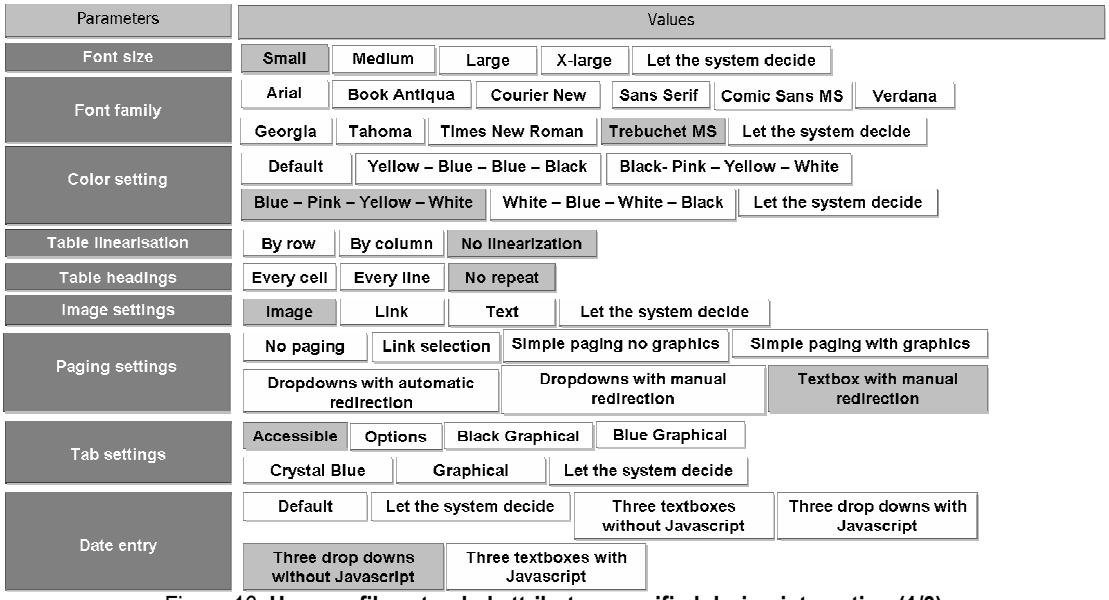


Figure 16: **User profile extended attributes specified during interaction (1/3)**

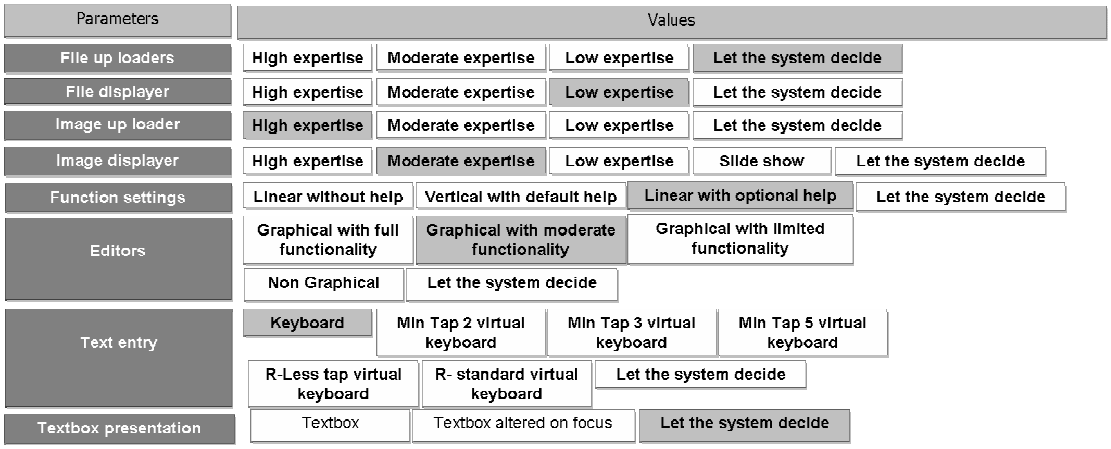


Figure 17: **User profile extended attributes specified during interaction (2/3)**

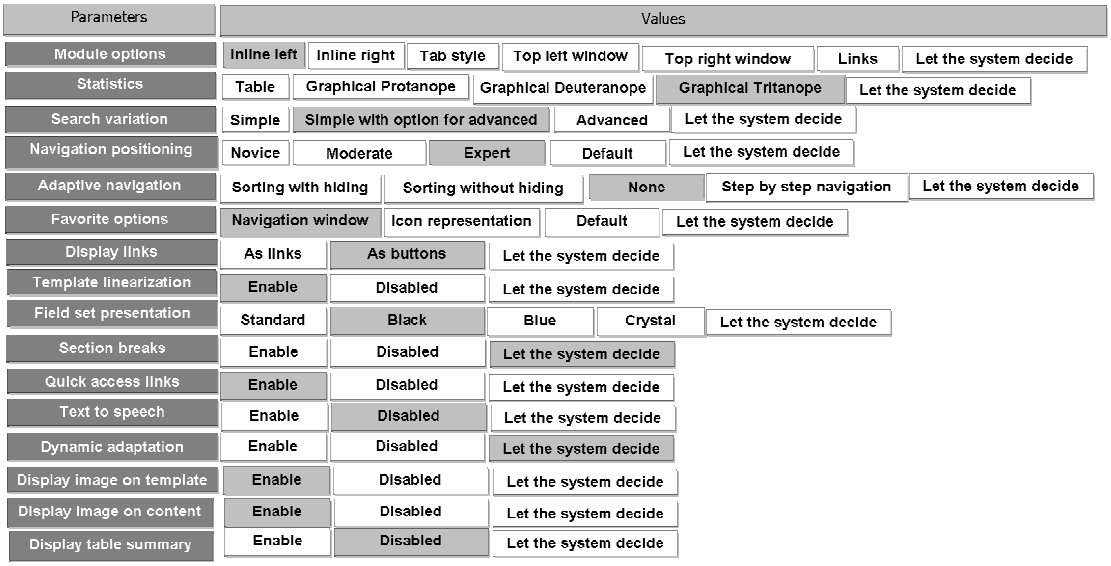


Figure 18: **User profile extended attributes specified during interaction (3/3)**

To support the aforementioned enhancement, the way that user profile attributes are stored the UIC database schema was enriched to incorporate the specific user extended characteristics as presented in Figure 19.

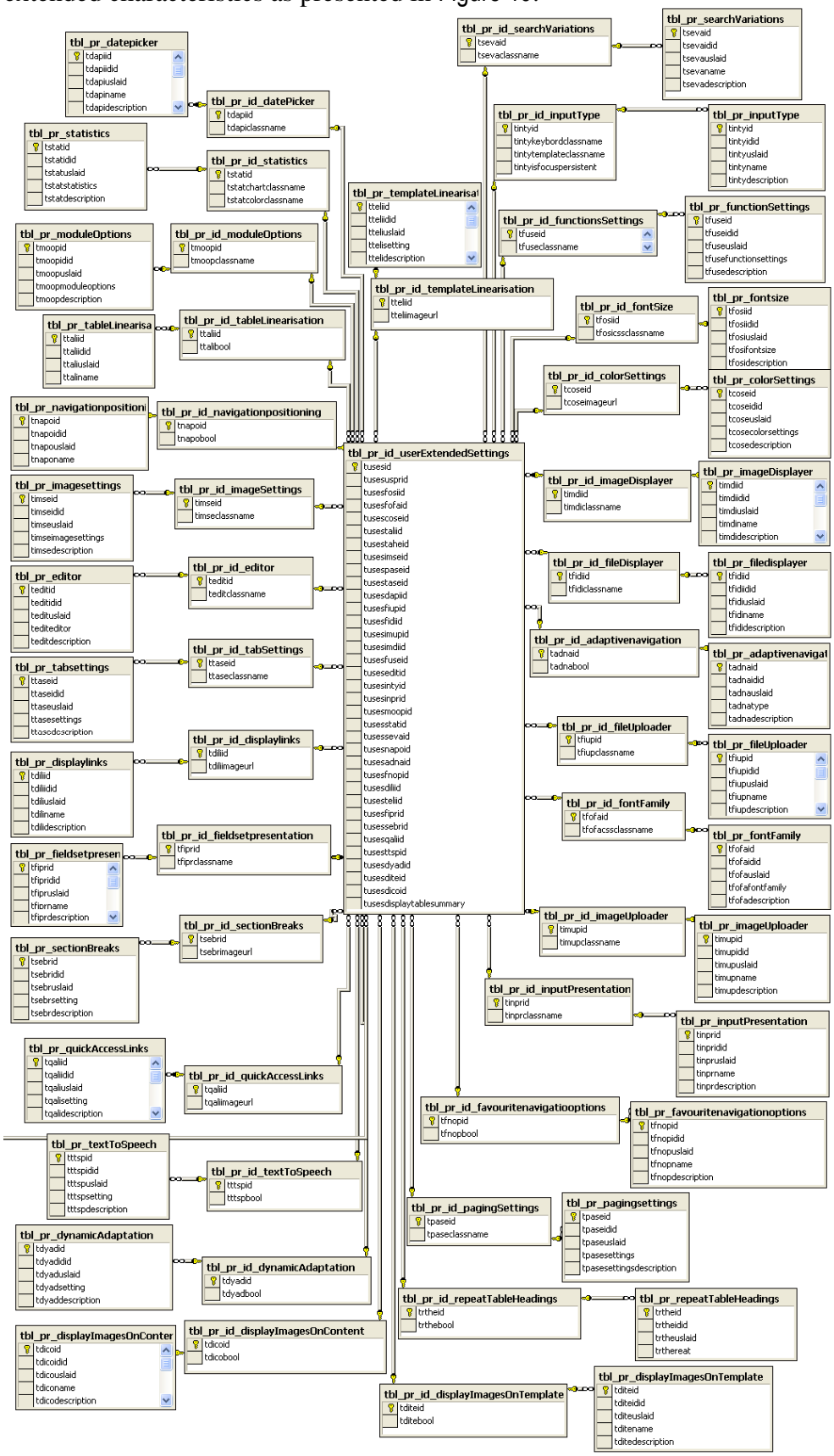


Figure 19: **The database schema used for storing the extended user characteristics**

5.1.3 Profiling Module

The basic class used for propagating user attributes is the UserInformationComponent that exposes functionality for enabling the retrieval of basic and extended user characteristics as presented in Table 4. The representation of the User Information Component together with the class used for storing the final user settings propagated by the DMC is presented graphically in Figure 20.

Table 4: **The *UserInformationComponent* class**

UserInformationComponent	
Methods	
getUserBasicCharacteristics	This function returns the user basic settings (language, disability, etc).
getUserExtendedCharacteristics	Functions used for getting the user manually extended activations – deactivations o interactive elements.
getUserHasExtendedCharacteristics	Returns whether the user has extended characteristics.
getUserBasicCharacteristics	Returns whether the user has basic characteristics.

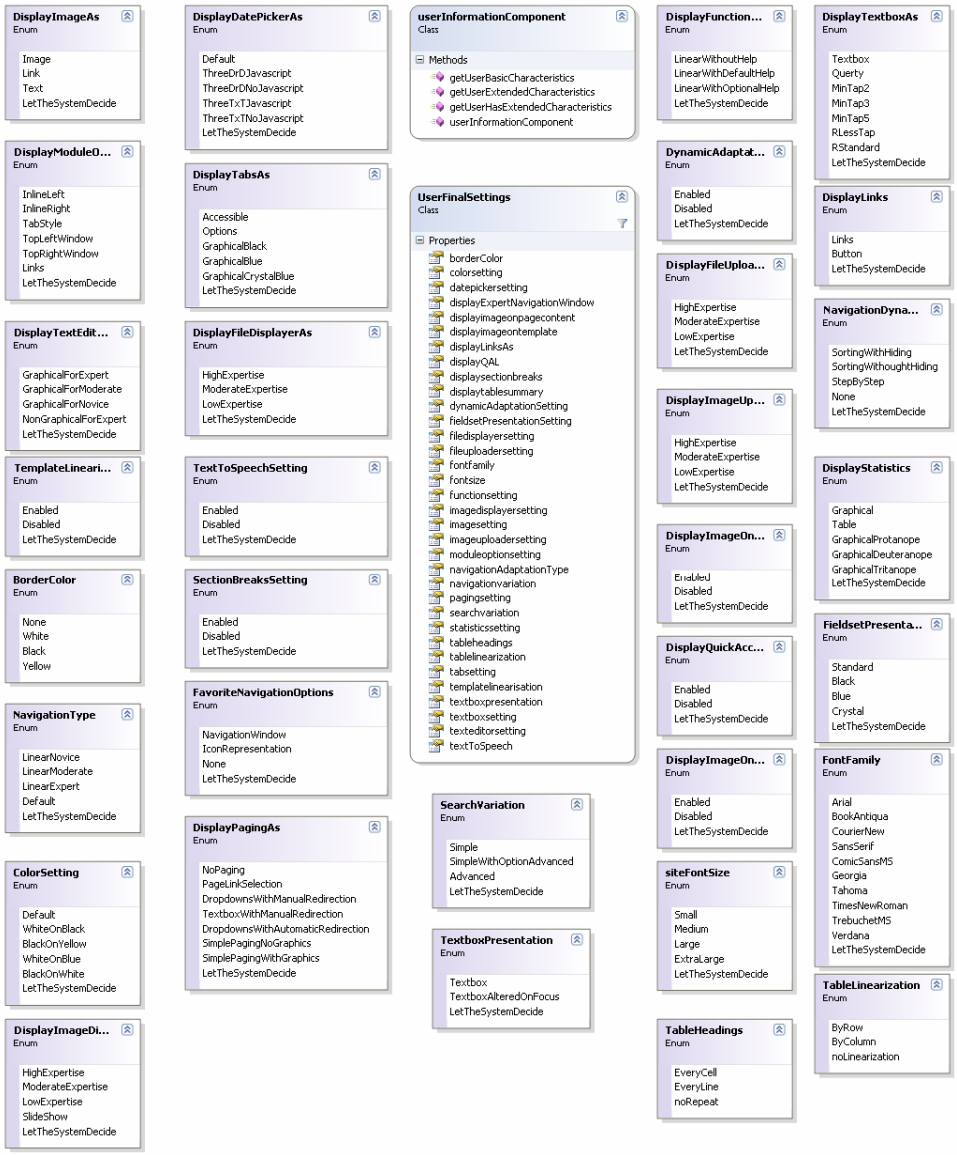


Figure 20: **Class diagram of *UserInformationComponent* and *UserFinalSettings***

5.1.4 Interaction Monitoring Module

The Interaction Monitoring Module enables the collection and propagation of attributes regarding user actions. The module responsible for performing these operations exposes functionality for monitoring the outcome of user actions as presented in Table 5.

Table 5: **The *InteractionMonitoringModule* class**

InteractionMonitoringModule	
Methods	
increaseSuccessfullActionRepetitions	This function is used to increase the successful repetitions for a specific user and a specific action.
increaseUnsuccessfullActionRepetitions	This function is used to increase the unsuccessful repetitions for a specific user and a specific action.

The Interaction monitoring class diagram is presented in Figure 21.

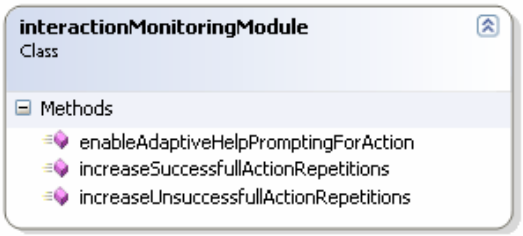


Figure 21: **The class diagram of the *InteractionMonitoringModule* class**

5.1.5 User Profile Statistics UI Module

This module is responsible for generating statistics based on the data collected from user profiles. More specifically, the statistics generated by this module include at a first level the popularity of selections regarding the User Information Component. Additionally, a more detailed overview of specific user setting is provided by generating statistics regarding the popularity of specific activations or deactivations of interactive elements. The information provided by this module can be used to understand the behaviour and preferences of actual users in order to get feedback and enrich the decision logic of the framework. A detailed presentation of the User Profile Statistics UI module is presented in section 7.2.3.1.

5.1.6 User Profile Selection UI Module

The user profile selection UI module acts as a front end to the final users of applications developed using the EAGER toolkit. The UI module enables end users to enter their profile information in order for the decision making to take place. Additionally specific UI interfaces are provided for manual overriding system's decisions. Using these interfaces all the user specific settings regarding activation or deactivation of interactive elements can be set. It is therefore prominent that the eager framework provides not only the underling infrastructure, but also specific user based or system based administrative facilities and in depth presentation of the User Profile Selection UI module in terms of functionality, user interface features and its incorporation into an actual portal implementation is presented in section 7.2.2.1.

5.1.7 User Profile Administration UI Module

The User Profiles Administration UI module is an administrative facility that enables the generation of predefined user profiles. These profiles can be in turn be published and used by the application end users to perform a quick setup of their interface. In that way a novice user, for example, can select among a descriptive predefined user profile instead of using the User Profile Selection UI module to edit specific user profile parameters. A detailed presentation of the User Profile Administration UI module is presented in section 7.2.3.2.

5.2 Context Information Component (CIC) implementation

As described in the previous Chapter, CIC consists of the User Context Profiles Repository and the Context Monitoring Module (see Figure 8). Their implementation with EAGER is presented in detail in the following subsections.

5.2.1 User Context Profiles Repository

The User Context Profiles Repository stores information such as the technical equipment used for accessing a web site (desktop, palmtop, mobile phone, etc.) or the assistive technologies that are used (screen reader, screen magnifier etc), as presented in Figure 23. This kind of information is of particular importance for enabling the Decision Making Component (described in next section) to infer the presentation characteristics and limitations of each individual user request.

The Database architecture used for the User Context Profiles Repository is presented in Figure 22.

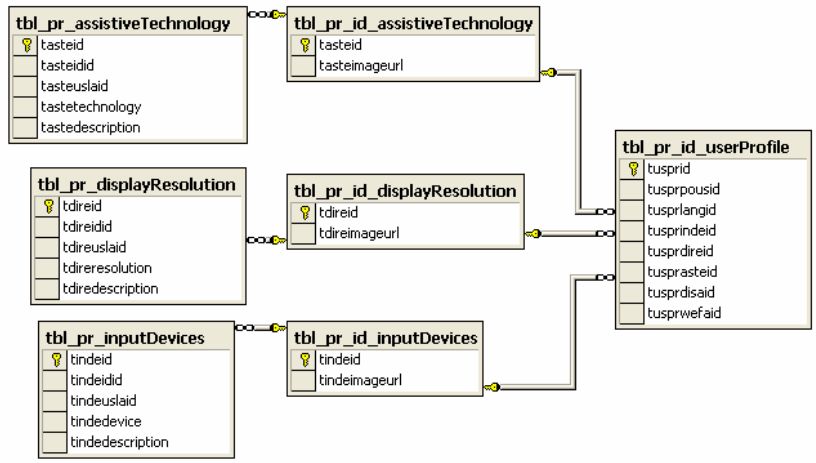


Figure 22: **User Context Profiles Repository Database scheme**

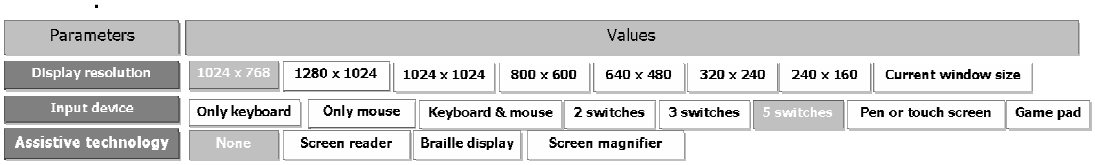


Figure 23: **Context Specific Attributes**

5.2.2 Context Monitoring Module

The Context Monitoring Module plays an important role for identifying the presentation characteristics of each client in terms of the device and display resolution used. Towards this direction the architecture implemented (see Figure 24) incorporates the appropriate functions for requesting information by the Content Monitoring Module.

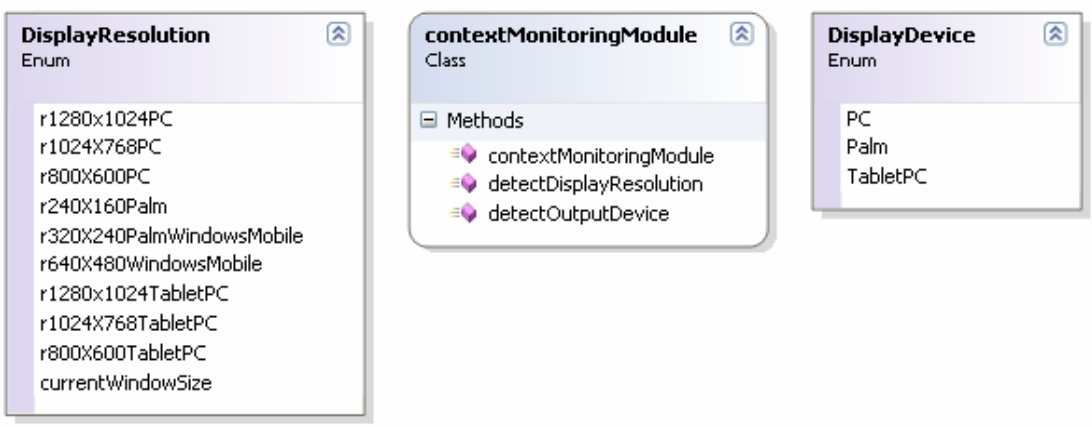


Figure 24: **The class diagram of the *ContextMonitoringModule* class**

A complete presentation of the API offered by this module is presented in Table 6.

Table 6: **The *ContextMonitoringModule* class**

Context Monitoring Module	
Methods	
contextMonitoringModule	Constructor: Responsible for the initialisation of the Context Monitoring Module
detectDisplayResolution	This function is responsible for detecting the client's display resolution
detectOutputDevice	This function is responsible for detecting the device used for accessing the Web Application such as PDA, PC, etc.

5.3 Decision Making Component (DMC)

The repository of mappings contains design information regarding the user interface for ensuring the usefulness of occurring adaptations. More specifically, the DMC has the following functionality. Initially, if only the basic user and context attributes are available, it performs decision making based on these attributes and the incorporated repository of mappings. The process of decision making is more complex when specific user interaction and accessibility setting are present. In this case, extended settings are overridden only if not explicitly selected by the user. In order for the decisions to be propagated by the DMC component after each decision making session is completed, a new repository of final settings is created for each user. The DecisionMakingComponent class is responsible for generating the final set of user characteristics and additional for propagating the decisions made.

The functionality exposed by the DecisionMakingComponent is presented in Table 7.

Table 7: **The *DecisionMakingComponent* class**

Decision Making Component	
Methods	
DecideAboutUserCharacteristics	Overloaded. Generate the collection of decisions using data stemming from the user information server.
desideAboutNavigationSortingType	Gets the navigation sorting type to be used
displayAccessibleChat	Gets whether to display Accessible chat.
displayFavoritesNavigationOptionsIcon	Gets whether to display Favourites navigation option.
displayQAL	Gets whether to display Quick Access Links.
displaySectionBreaks	Gets whether to display Section breaks.
getBodyColourCssClassName	Gets the css class to be used for applying the body colour.
getBodyColourSetting	Gets the colour setting to be used for applying the body colour.
getBodyFontFamilyCssClassName	Gets the css class to be used for the selected font family.
getBodyFontSizeCssClassName	Gets the css class to be used for the selected font size.
getBorderCssClassName	Gets the css class to be used on borders.
getButtonCssClassName	Gets the css class to be applied on buttons.
getDisplayImageOnTemplate	Gets whether to display images on template.
getErrorLabelCssClassName	Gets the css class to be applied on error labels.
getFinalUserCharacteristics	Returns the final collection of user settings.
getFontFamily	Gets the font family to be used for rendering portal content.
getLinkCssClassName	Gets the css class to be applied on links.
getNavigationPositioning	Gets the position where navigation shall be displayed.
getTextBoxCssClassName	Gets the css class to be applied on text boxes.
getTextBoxOnFocusCssClassName	Gets the css class to be applied on text boxes on focus.
isTTSEnabled	Gets whether Text to Speech output is enabled
templateLinearizationEnabled	Gets whether to linearize tables.

See the corresponding class diagram representation in Figure 25.

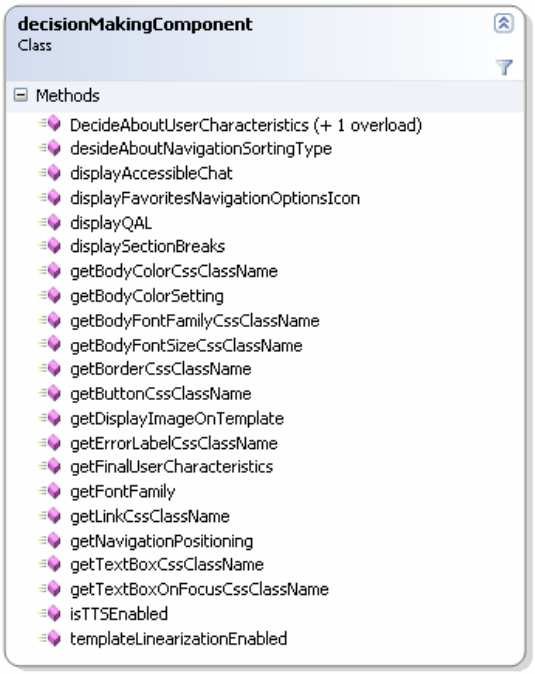


Figure 25: **The class diagram of the *DecisionMakingComponent* class**

The decision logic for performing the Decision Making in order to generate the final set of user characteristics is encapsulated in the DecisionLogic class. This class exposes functionality for deciding about specific setting, as presented in Table 8.

Table 8: **The *DecisionLogic* class**

Decision Logic	
Methods	
decideAboutAdaptiveNavigationType	DecideAboutAdaptiveNavigationType
decideAboutAdaptiveNavigationWindow	Decide about whether to display and Adaptive Navigation
decideAboutBorderColour	Performs decision making regarding the border colour
decideAboutColourSetting	Performs decision making regarding the colour setting
decideAboutDatePickerSetting	Performs decision making regarding the date picker
decideAboutDynamicAdaptationEnable	Decide about whether to enable Dynamic Navigation
decideAboutFontFamily	Performs decision making regarding the font
decideAboutFontSize	Performs decision making regarding the font size
decideAboutFunctionSetting	Performs decision making regarding the module functions style
decideAboutImageOnPageContent	Performs decision making regarding whether to display images on page content
decideAboutImageOnTemplate	Performs decision making regarding whether to display images on template
decideAboutImageSetting	Performs decision making regarding the way that images shall be displayed
decideAboutImageUploaderSetting	Performs decision making regarding the style of image uploader
decideAboutModuleOptionSettings	Performs decision making regarding the style of module options
decideAboutNavigationVariation	Performs decision making regarding the navigation style
decideAboutPaging	Performs decision making regarding the paging style
decideAboutQAL	Performs decision making regarding whether to display Quick Access Links
decideAboutSearchVariation	Performs decision making regarding the search
decideAboutSectionBreaks	Performs decision making regarding whether to display Section Breaks
decideAboutStatisticSettings	Performs decision making regarding the way that chart shall be displayed
decideAboutTableHeadings	Performs decision making regarding the repetition of table headings
decideAboutTableLinearization	Performs decision making regarding the linearisation of tables
decideAboutTableSummary	Performs decision making regarding whether to display Table Summary
decideAboutTabSetting	Performs decision making regarding tab presentation
decideAboutTemplateLinearizaton	Performs decision making regarding whether to perform template linearisation
decideAboutTextBoxPresentation	Performs decision making regarding the text box style
decideAboutTextBoxSetting	Performs decision making regarding the text box style
decideAboutTextEditorSetting	Performs decision making regarding the Editor style
decideAboutUploader	Performs decision making regarding the file up loader style

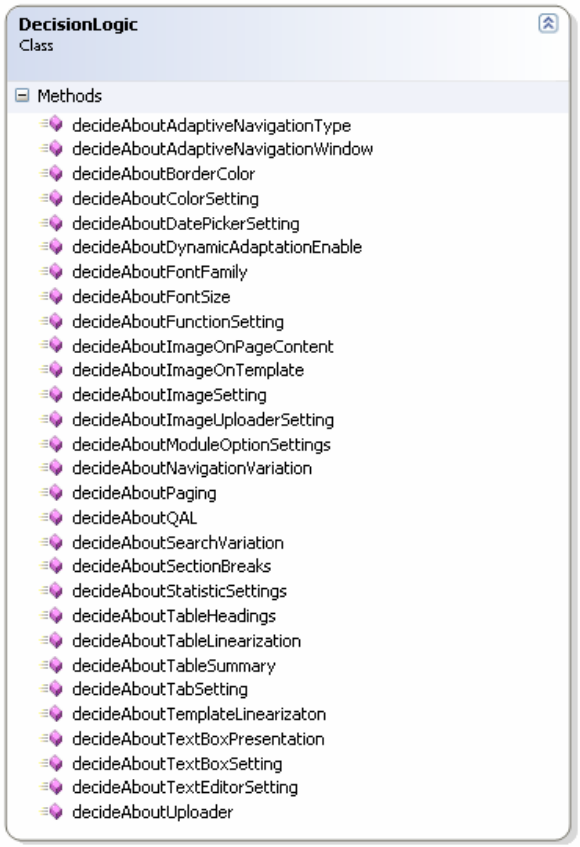


Figure 26: **The class diagram of the *DecisionLogic* class**

5.4 Dialogue Controls Component (DCC) implementation

The class exposed by the DCC for generating the appropriate instances of the control hierarchy according the specific control activation – deactivation settings exposes the functionality presented in Table 9.

Table 9: **The *DialogueControlComponent* class**

DialogueControlsComponent class	
Methods	
CreateTemplate	Function exposed for generating an instance of the table linearisation templates UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateAdaptiveHelpProvisionControl	Function exposed for generating an adaptive help control.
generateBarChartControl	Function exposed for generating an instance of the Bar Charts UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateBottomNavigationControl	Function exposed for generating an instance of the Bottom Navigation UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateCalendarControl	Function exposed for generating an instance of the Date Pickers UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateCategoriesNavigationControl	Function exposed for generating an instance of the Categories Navigation UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateChartColours	Function exposed for generating an instance of the Chart Colours Components Hierarchy based on the decision taken by the Decision Making Component.
generateEditorControl	Function exposed for generating an instance of the Editors UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateFavoritesNavigationControl	Function exposed for generating an instance of the Favourites Navigation UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateFileDisplayer	Function exposed for generating an instance of the File Displayers UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateFileUploader	Function exposed for generating an instance of the File Up loaders UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateFullNavigationControl	Function exposed for generating an instance of the Full Navigation UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateFunctionsContainerControl	Function exposed for generating an instance of the Module Functions UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateICSWidjet	Function exposed for generating an instance of the icsWidgets UI Components Hierarchy based on the decision taken by the Decision Making Component.
GenerateImageControl	Function exposed for generating an instance of the Images UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateImageDisplayer	Function exposed for generating an instance of the Image Displayer UI Components Hierarchy based on the decision taken by the Decision Making Component.

DialogueControlsComponent class	
Methods	
generateImageUploader	Function exposed for generating an instance of the Image Up loaders UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateMainNavigationControl	Function exposed for generating an instance of the main Navigation UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateMainNavigationOptionsControl	Function exposed for generating an instance of the Main Navigation UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateModuleOptionsControl	Function exposed for generating an instance of the Module Options UI Components Hierarchy based on the decision taken by the Decision Making Component.
generatePagingControl	Function exposed for generating an instance of the Paging styles UI Components Hierarchy based on the decision taken by the Decision Making Component.
generatePieChartControl	Function exposed for generating an instance of the Pie Charts UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateSearchControl	Function exposed for generating an instance of the Chart Colours Components Hierarchy based on the decision taken by the Decision Making Component.
generateSimplePagingControl	Function exposed for generating an instance of the Paging styles UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateSingleFileUploader	Function exposed for generating an instance of the Single File Up loaders UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateSingleImageUploader	Function exposed for generating an instance of the Single Image Up loaders UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateSubcategoriesNavigationControl	Function exposed for generating an instance of the Subcategories UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateTab	Function exposed for generating an instance of the Tab Styles UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateTabLinear	Function exposed for generating a linear tab.
generateTextBoxControl	Function exposed for generating an instance of the Text Boxes UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateTopNavigationNavigationControl	Function exposed for generating an instance of the top Navigation UI Components Hierarchy based on the decision taken by the Decision Making Component.
generateWizardControl	Function exposed for generating an instance of the Wizards UI Components Hierarchy based on the decision taken by the Decision Making Component.

The class diagram for this class is presented in Figure 27 below.

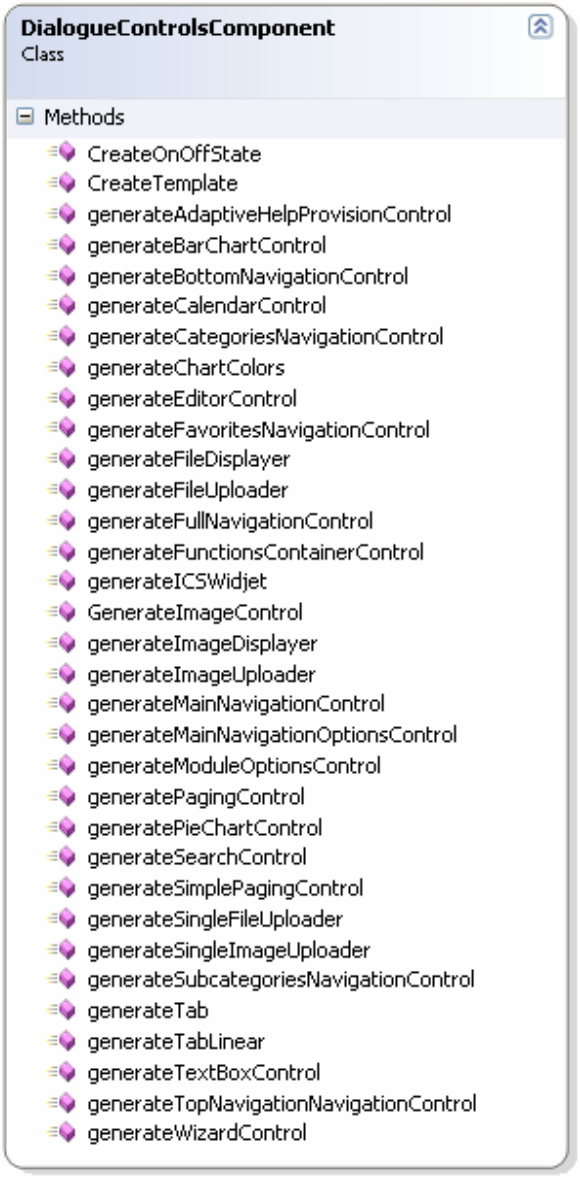


Figure 27: **The class diagram of the *DialogueControlComponent* class**

5.4.1 Dialogue controls hierarchy

This sub-section presents Dialogue Controls Hierarchy. More specifically, the namespace ics.Adaptation.DialogueControls contains all the designer enabled controls to be used by application developers for building adaptive applications. Each control contained in this namespace represents an interaction element that can be instantiated at runtime to a number of different styles. These styles are invisible to the end users of the adaptation library and are contained in a different namespace.

For example, all the variations of the paging controls are contained in:

"ics.Adaptation.DialogueControls.PagingStyles"

The following illustration (Figure 28) shows the hierarchy of controls in the **ics.Adaptation.DialogueControls** namespace and its derived namespaces.

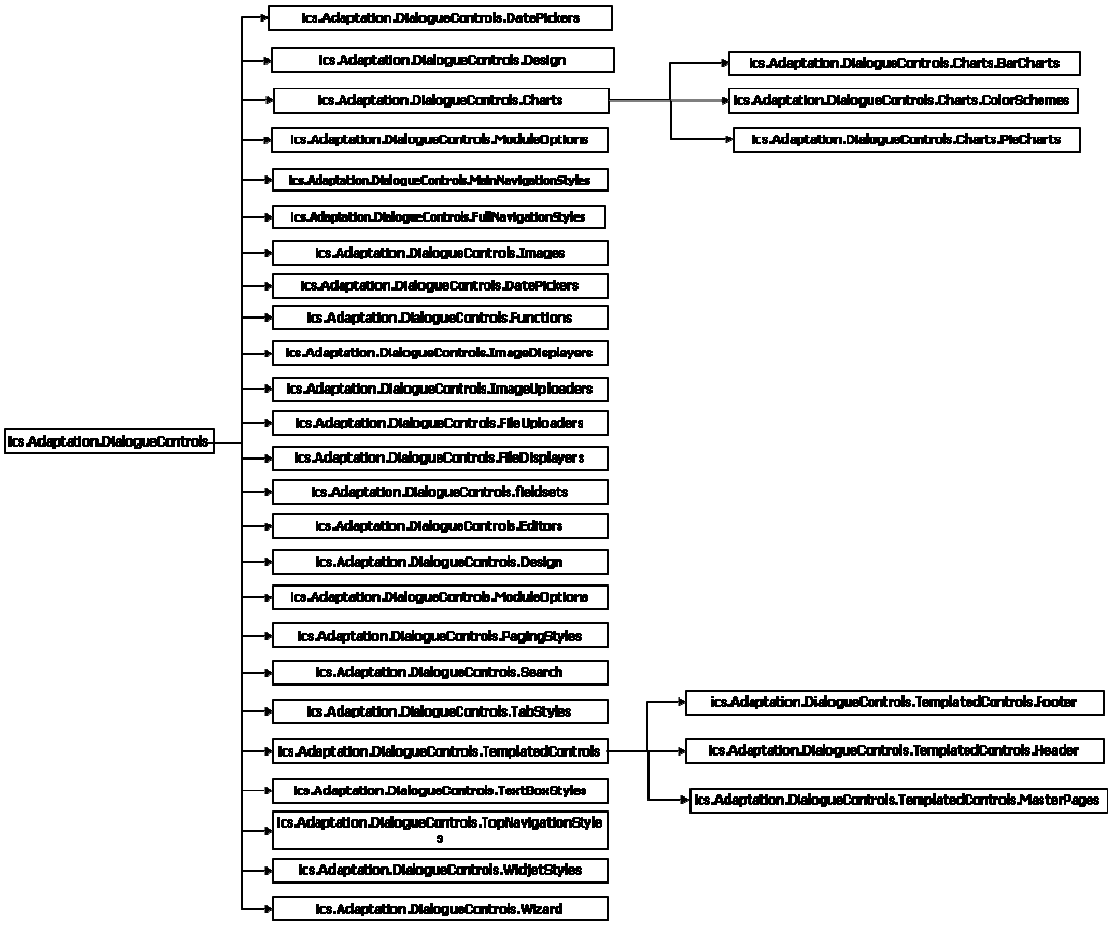


Figure 28: **The ics.Adaptation.DialogueControls namespace**

5.4.2 Examples of dialogue controls implementation

5.4.2.1 Content adaptation

Example: Images implementation

The way that a web application presents images can be altered according to the specific user characteristics and presentation requirements. The different image displaying schemes that stemmed from the design phase are facilitated by the framework through the image UI components hierarchy presented in Figure 29. The hierarchy is constituted by the icsImageControl base that acts as a common parent for all alternative image styles. This base class includes all the appropriate fields and methods (see Table 10) in order to be extended and used by the child classes to achieve specialized configuration of an image control.

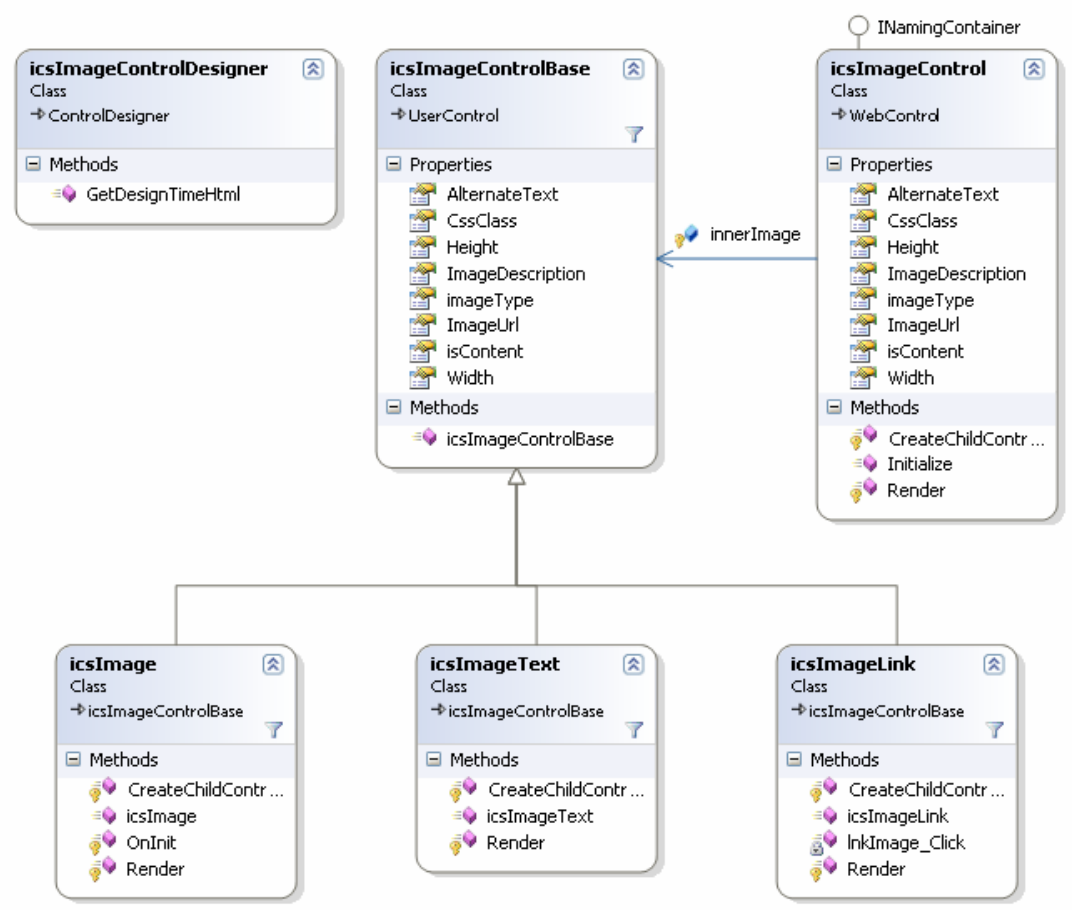


Figure 29: **Images hierarchy**

Table 10: ***icsImageControlBase*** **class properties – methods**

icsImageControlBase	
Properties	
AlternateText	Gets or Sets the image alternative text.
CssClass	Gets or Sets the CssClass to be used on the image control.
Height	Gets or Sets the image height.
ImageDescription	Gets or Sets the image description.
imageType	Gets or Sets the image type.
ImageUrl	Gets or Sets the image url.
isContent	Gets or Sets whether the control is intended for content.
Width	Gets or Sets the image width.

The 'icsImageControlBase' child classes are presented in Table 11 along with a summative description. 'icsImageControl' (Figure 29) encapsulates an 'icsImageControlBase' object that at runtime is instantiated in an object of the child classes, providing end user with a configurable image control.depending on the decision making component commands. 'icsImageControl' is placed on the Visual Studio toolbox, from where the developer drags and drops it in the Visual Studio designer. In order for this control to be represented, the 'icsImageControlDesigner' class is used, providing html code for rendering graphically the control in the designer.

Table 11: **Image control class hierarchy**

Class	Description
icsImage	This is the control used to render an image normally
icsImageControlBase	This is the base class for each of the Image Controls UI Components hierarchy.
icsImageLink	This is the control used for rendering an image as link
icsImageText	This is the control used for rendering the text representation of an image

Example: Images displayers implementation

Web portals include image lists too. During the design phase, alternative variations for presenting images were identified, based not only on the expertise of a given user, but also on specific interaction requirements. To facilitate these alternatives, the image displayers UI controls hierarchy was developed, as presented in Figure 30.

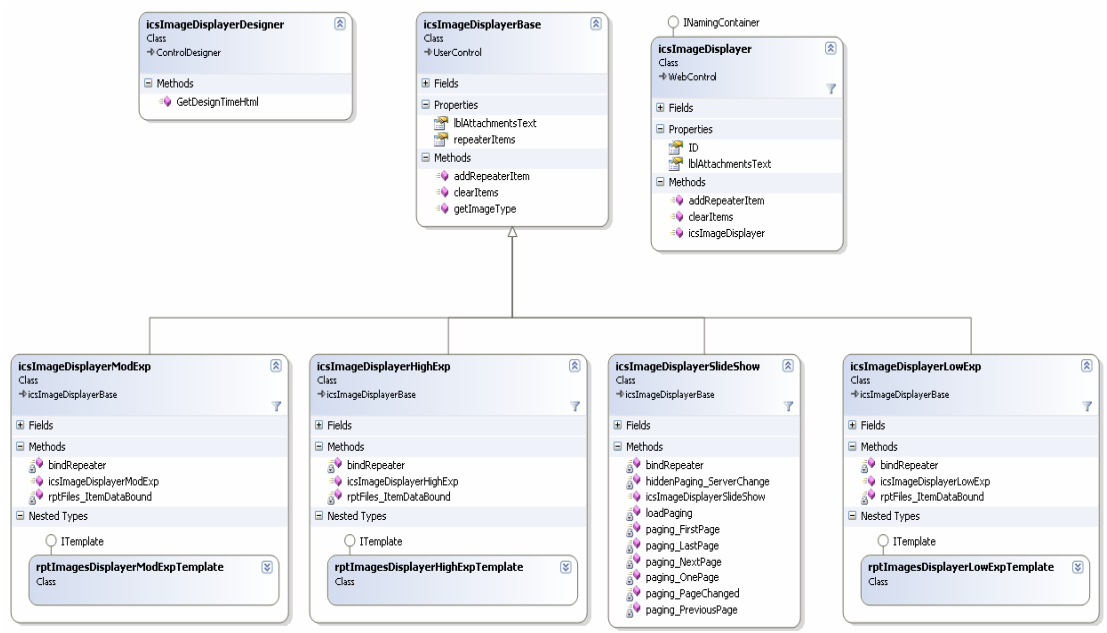


Figure 30: **Image displayer diagrams**

The 'icsImageDisplayerBase' class includes the basic properties and methods (see Table 12) that are implemented by the derived classes (see Table 13) in order to generate special image displayers, as shown in Figure 30.

Table 12: ***icsImageDisplayerBase* class properties – methods**

icsImageDisplayerBase	
Properties	
lblAttachmentsText	The text used for describing images.
repeaterItems	The collection of images to be displayed.
Methods	
clearItems	Function exposed for clearing uploaded items.
getImageType	Function exposed for getting the image type given a images file name.

Table 13: **Image displayers control library**

Class	Description
icsImageDisplayerBase	This is the base class for each of the Image Displayers UI Components hierarchy.
icsImageDisplayerHighExp	This is a control image displayer control that shows full information about each image. In this control images are displayed as a list.
icsImageDisplayerLowExp	This is a control image displayer control that shows basic information about each image. In this control images are displayed as a list.
icsImageDisplayerModExp	This is a control image displayer control that shows moderate information about each image. In this control images are displayed as a list.
icsImageDisplayerSlideShow	This is a control that displays images as a slide show with navigation facilities attached for navigating back and forth to the images collection.

The 'icsImageDisplayerDesigner' class renders in html the 'icsImageDisplayerBase' control for design time support. In order for a developer to add an image displayer to a web application, the 'icsImageDisplayer' control is offered from the Visual Studio toolbox area which encapsulates an 'icsImageDisplayerBase' object, and is instantiated to a distinct object of the derived classes as necessary in each case.

5.4.2.2 Layout adaptation

Example: Template implementation

Templates affect the basic characteristics of each page controlling the positioning to be applied on page content, the available containers of page content and the presentation characteristics used for rendering page content. To achieve this, a hierarchy of different template variations has been developed, as presented in Figure 31.

Figure 31: **The hierarchy of the potential template variations**

The icsMasterPageBase acts as the base class for all different template variations available exposing all the functionality to be overridden by its descendors for providing a specific implementation.

More specifically, as presented in Table 14, icsMasterPageBase provides access to a number of placeholder for rendering each specific template contents and additionally parameters for controlling template styles and functionality for rendering a template in variant ways (e.g., accessible presentation, design view presentation, etc.).

Table 14: ***icsMasterPageBase* class**

icsModuleOptionBase	
Properties	
backgroundImageStyleLeft	Gets or Sets the background to be applied on the left side of the template.
backgroundImageStyleMiddle	Gets or Sets the background to be applied on the middle of the template.
backgroundImageStyleRight	Gets or Sets the background to be applied on the right side of the template.
currentMode	Gets or Sets the template mode (preview, design etc).
designModeBorderStyle	Gets or Sets the style to be applied when rendering borders.
PanelMain	Gets the PanelMain.
PanelNavigator	Gets the PanelNavigator.
PlaceHolderContents	Gets the placeholder where fcontents are placed.
PlaceHolderContentsRight	Gets the placeholder where contents on the right are placed.
PlaceHolderContentsRight2	Gets the placeholder where contents on the righter are placed.
PlaceHolderFooter	Gets the placeholder where the footer is placed.
PlaceHolderFooterMenu	Gets the placeholder where the footer menu is placed.
PlaceHolderHeader	Gets the placeholder where header is placed.
PlaceHolderMenu	Gets the placeholder where menu is placed.
PlaceHolderNavigation	Gets the placeholder where navigation is placed.
PlaceHolderRight	Gets the placeholder for the right contents.
templatePaddingStyleLeft	Gets or Sets the style used for master page padding.
templatePaddingStyleRight	Gets or Sets the style used for master page padding.
templateWidthStyle	Gets or Sets the style used for master page width.
tempLoader	Gets or Sets the internal instance of the template loader calss.

icsModuleOptionBase	
Methods	
buildStylesForAccessibleMode	Function to be override for building styles when control is in accessible mode.
buildStylesForDesignMode	Function to be override for building styles when control is in design mode.
buildStylesForPreviewMode	Function to be override for building styles when control is in preview mode.
buildStylesForStandardMode	Function to be override for building styles when control is in standard mode.
createStyles	Function to be override for creating template styles.
ParentControl	Gets the parent control.

As shown in Figure 31, according to the specific user attributes, a page template can be rendered in the form of any of the icsMasterPageBase descendors presented in Table 15.

Table 15: **The available descendors of the *icsMasterPageBase* control**

Class	Description
icsTemplateFourColumns	Represents a four column template.
icsTemplateFourColumnsAlt	Represents a four column template.
icsTemplateOneColumn	Represents a one column template.
icsTemplateOneColumnAlternate	Represents a one column template.
icsTemplateThreeColumns	Summary description for PageUserControlThreeColumns.
icsTemplateThreeColumnsAlt	Summary description for PageUserControlThreeColumns.
icsTemplateThreeColumnsReverse	Represents a three column reverse template.
icsTemplateTwoColumns	Represents a two columns template.
icsTemplateTwoColumnsReverse	Represents a two columns reverse template.

5.4.2.3 Navigation adaptation

Example: Full navigation implementation

In many occasions, the existence of several navigation schemes in different screen positions may influence the usability of a web application, especially when the scanning of a complete page for each selection is required. In this case, an alternative way to navigate through pages is required. The full navigation scheme was implemented to provide a navigation alternative that would be available from a specific screen location.

The full navigation hierarchy is presented in Figure 32.

Figure 32: **Full navigation sub template control attributes**

The generation of the full navigation scheme is based on the composition of three alternative controls representing the navigation categories, subcategories and options. Each of the aforementioned controls is originated for a different UI components hierarchy:

- The icsCategories navigation hierarchy (see Figure 33): This hierarchy contains alternative categories representations according to the expertise of the user currently accessing an interface.
- The icsSubCategories navigation hierarchy (see Figure 34): This hierarchy contains alternative sub - categories representations according to the expertise of the user currently accessing an interface.
- The icsOption navigation hierarchy (see Figure 35): This hierarchy contains alternative options representations according to the expertise of the user currently accessing an interface.

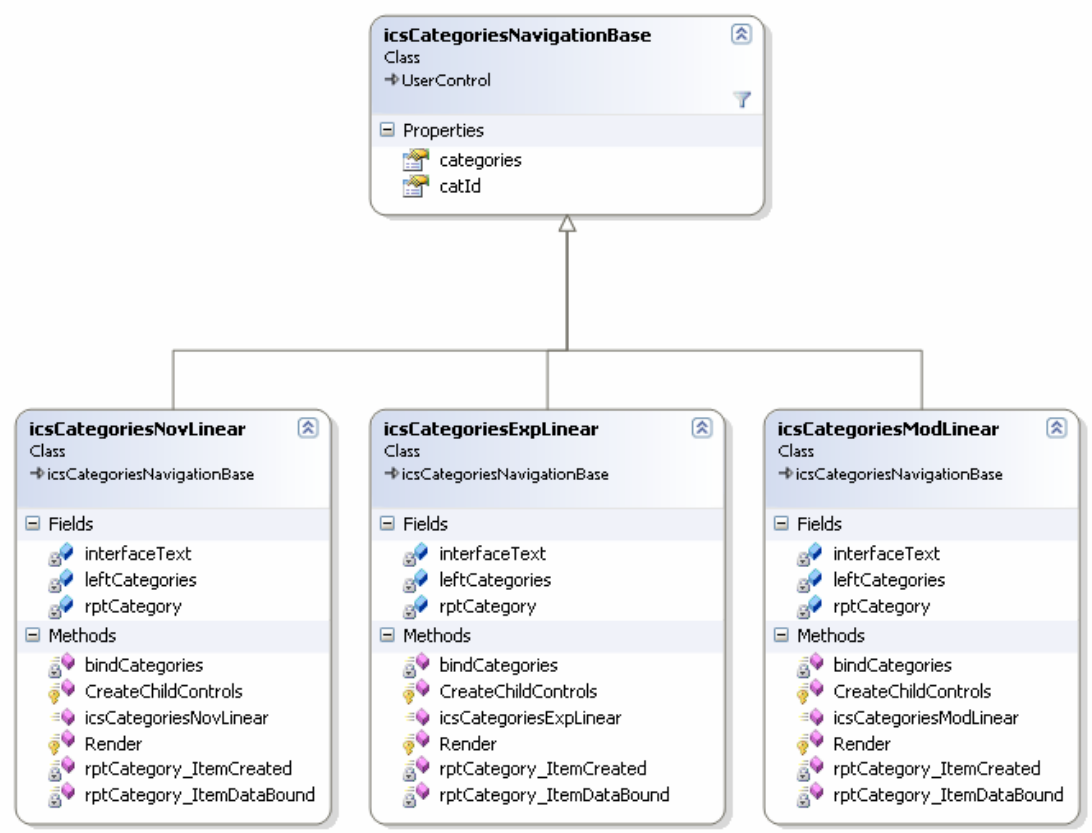


Figure 33: **Categories navigation diagram**

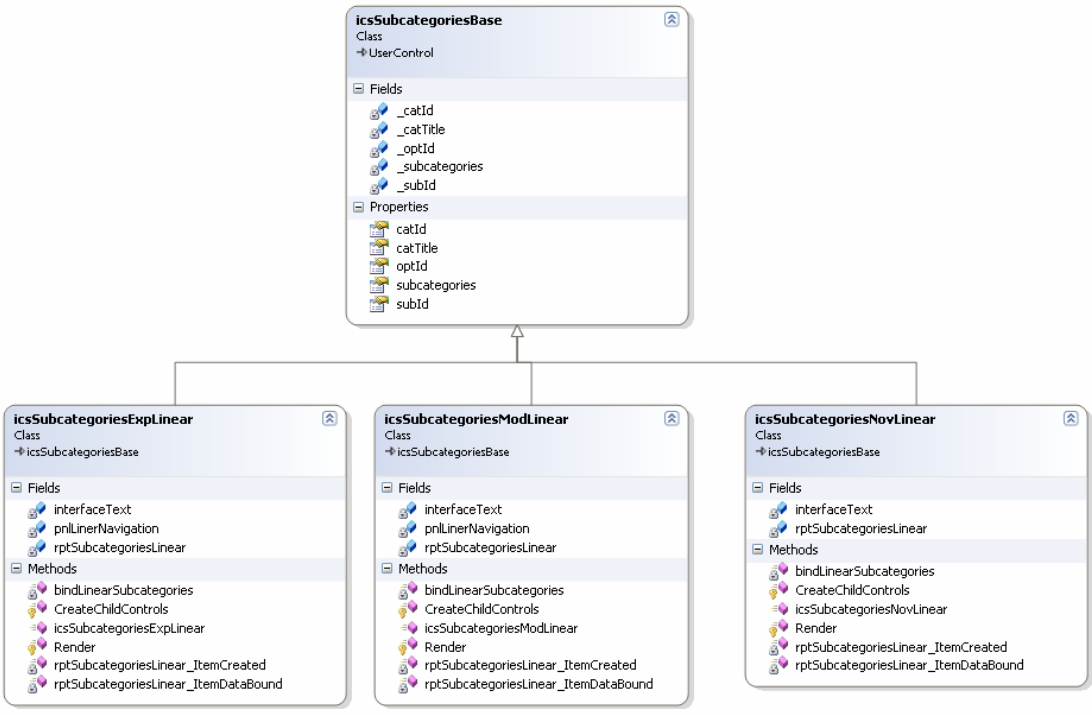


Figure 34: **Sub categories navigation diagram**

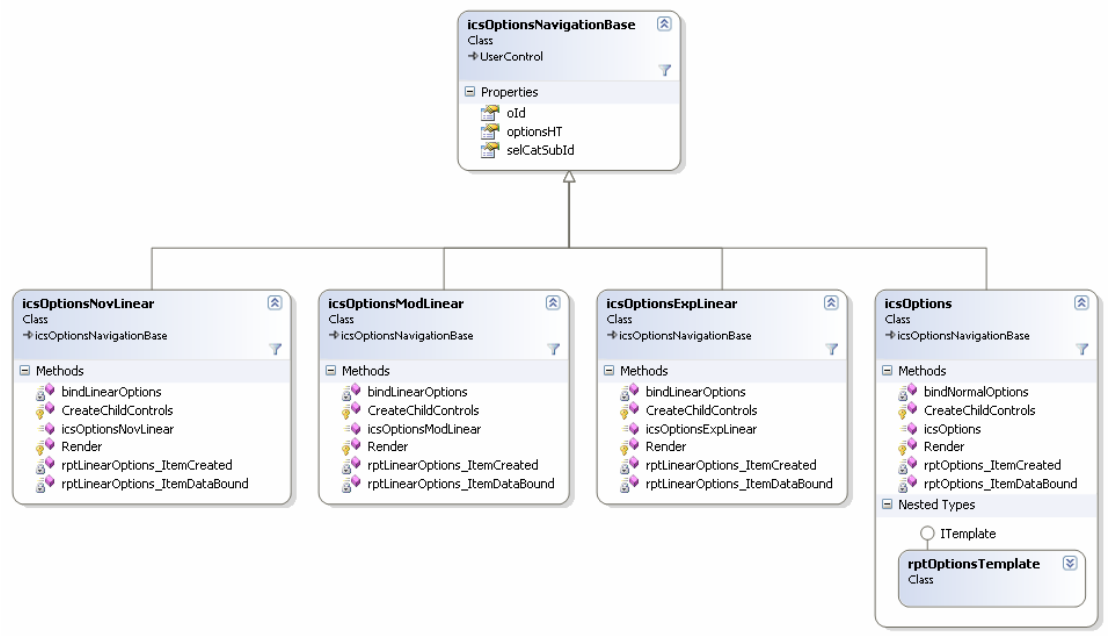


Figure 35: **Option navigation diagram**

5.4.2.4 Interaction adaptation

Example: File up loader implementation

Modern web applications typically introduce facilities, such as file sharing downloads, that mainly facilitate the concept of a common file system for a wide number of users. In this context, the process of uploading files is of particular importance, especially for users with limited experience with using web applications. In the framework, the different file uploaded styles identified are facilitated thought the file up loaders polymorphic hierarchy presented in Figure 36.

The 'icsFileUploaderBase' class, acting as a parent of all file up loaders, incorporates all the appropriate methods and functionality (see Table 16) to be shared among the derived classes. Several derived classes inherit from the base class in order to match different user and contexts of use needs and are summarised in Table 17. 'icsFileUploader' is used by the developers in order to place a file uploader object in a web page. The procedure from the systems' perspective is that 'icsFileUploader' includes an 'icsFileUploaderBase' object that at design time is presented in a designer following the 'icsFileUploaderDesigner' guidelines, and at runtime is transformed in an object of the derived classes depending on user and context of use attributes.

Figure 36: **File uploader diagram**

Table 16: ***icsFileUploaderBase* class properties – methods**

icsFileUploaderBase	
Properties	
destID	Gets or Sets the ID to be combined with the final path for generating a unique path.
fpath	Gets or Sets the final path of uploaded files.
isUpdateMode	Gets or Sets whether the control is intended for updating existing files.
lblFileText	Gets or Sets the text to be used for marking attached files.
repeaterItems	Gets or Sets the collection of uploaded items.
sourceID	Gets or Sets the ID to be combined with the source path for generating a unique path.
sourcePath	Gets or Sets the source path of uploaded files.

icsFileUploaderBase	
Methods	
clearItems	Method used for clearing uploaded files.
initialiseFiles	Overloaded. Method used for initialise file when on update mode.
writeToFinalLocation	Method used for writing uploaded files to their final location.

Table 17: **File uploader class hierarchy**

Class	Description
icsFileUploaderBase	This is the base class for each of the File Up loaders UI Components hierarchy.
icsFileUploaderDirectManipulation	Summary description for icsFileUploaderHighExp.
icsFileUploaderIndirectManipulation	Summary description for icsFileUploaderLowExp.
icsFileUploaderMixedModeManipulation	Summary description for icsFileUploaderModeExp.
icsSingleFileUploaderHighExp	Summary description for icsSingleFileUploaderHighExp.

Example: Date entry implementation

Date entry is a frequent task carried out by users of web applications. As identified during the design phase, a number of variations in style can be facilitated according to specific user characteristics. In order for these variations to be supported by the framework, the datepickers class hierarchy was developed, as presented in Figure 37.

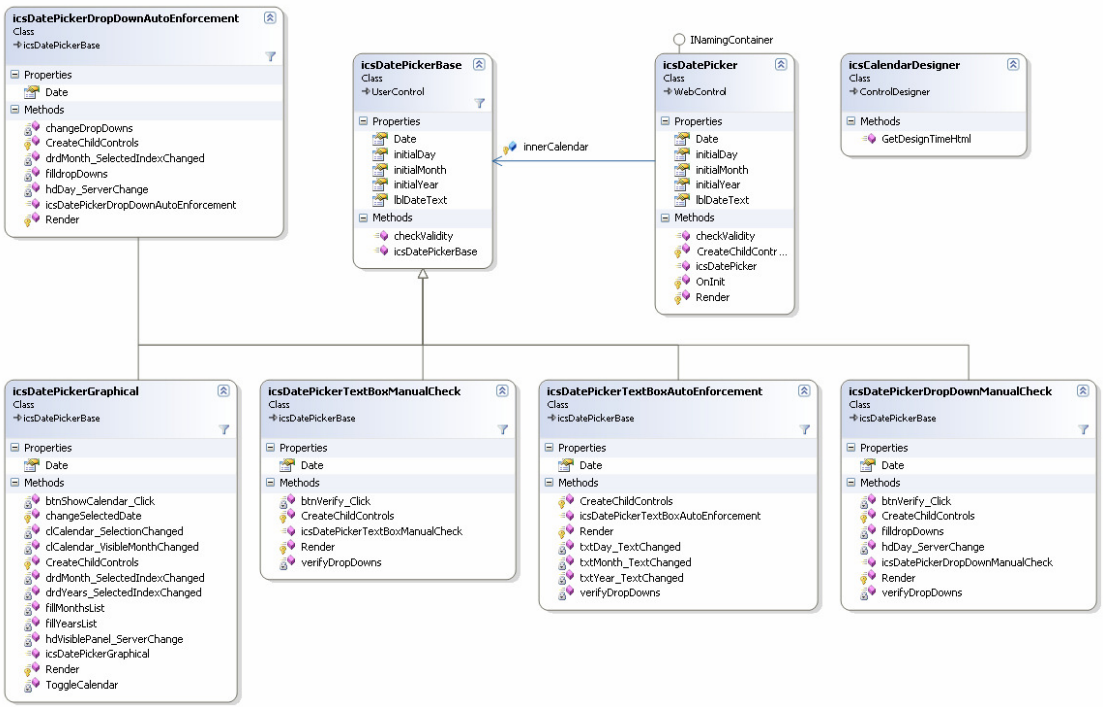


Figure 37: **Date entry diagram**

The 'icsDatePickerBase' constitutes the base class of the Datepickers hierarchy, including methods and properties appearing in Table 18, and being used and extended by the derived classes to meet individual user and contexts of use needs.

Table 18: **'icsDatePickerBase' class properties – methods**

icsDatePickerBase	
Properties	
Date	Represent the user selected date.
initialDay	Represent the initial day to be used when rendering the date picker.
initialMonth	Represent the initial month to be used when rendering the date picker.
initialYear	Represent the initial year to be used when rendering the date picker.
lblDateText	Represent the text identifier used when rendering the date picker.
Methods	
checkValidity	Method exposed for checking the validity of inserted date information.

In Figure 37 the overall class hierarchy of the date picker control is presented including the derived classes, as described in Table 19:

- icsDatePickerDropDownAutoEnforcement
- icsDatePickerDropDownManualCheck
- icsDatePickerGraphical
- icsDatePickerTextBoxAutoEnforcement
- icsDatePickerTextBoxManualCheck

The 'icsDatePicker' web control includes an instance of the 'icsDatePickerBase' class, that at runtime transforms itself to a derived class object and at design time to 'icsDatePickerDesigner' object (renders a generic date picker control).

Table 19: **Date picker class hierarchy**

Class	Description
icsDatePickerBase	This is the base class for each of the date pickers UI Components hierarchy.
icsDatePickerDropDownAutoEnforcement	This is the control representation of a date picker with dropdowns for month- day-year and automated checking of date validity
icsDatePickerDropDownManualCheck	This is the control representation of a date picker with dropdowns for month- day-year and manual checking of date validity
icsDatePickerGraphical	This is the control representation of a date picker with a graphical calendar for date selection
icsDatePickerTextBoxAutoEnforcement	This is the control representation of a date picker with text boxes for month- day-year and automated checking of date validity
icsDatePickerTextBoxManualCheck	This is the control representation of a date picker with text boxes for month- day-year and manual checking of date validity.

5.5 Designs Repository

5.5.1 Content-related alternative designs

Example: Images

Images appear usually as content of Web portals. Blind or low vision users are not interested in viewing images but only in reading the alternative text of them that describes the image. In order to facilitate blind and low vision users, two design alternatives were produced which are presented in Figure 38.

The text representation of the image simply does not present the image, but only a label with the prefix 'Image:' and followed by the alternative text of the image. The second representation, targeted to users with visual impairments, is same as the first with the difference that, instead of a label, a link is included that leads to the specific image giving the ability of saving the image. In particular, a blind user may not wish to view an image but may wish to save it to a disk and use it properly.

In addition to the above, another design was produced that can be selected as a preference by web portal users in which the images are represented as thumbnail bounding the size that holds on the web page. A user who wishes to view the image in normal size may click on it.

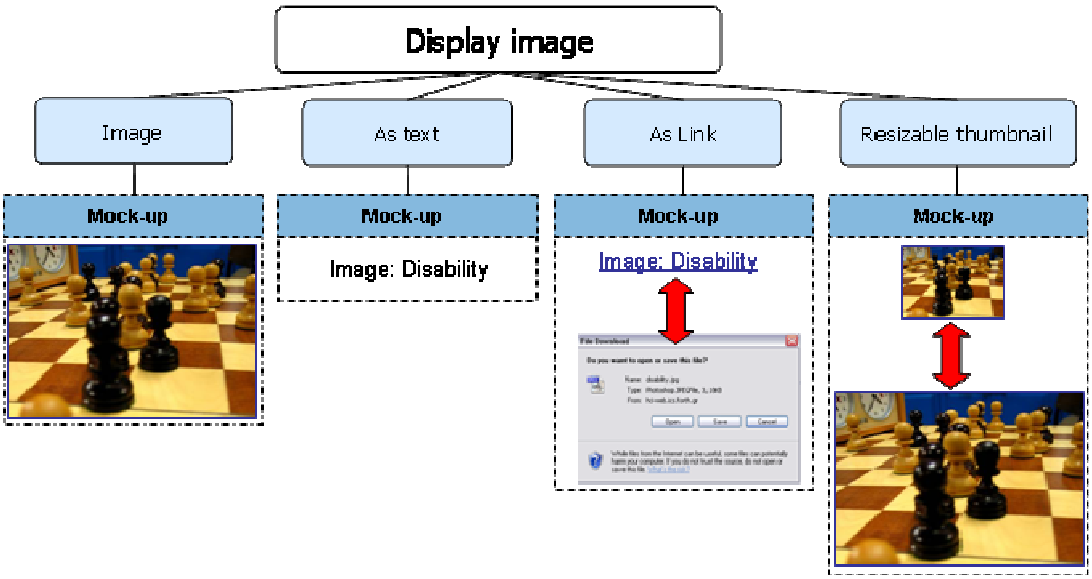


Figure 38: **Image alternative representations**

In Table 20, the design rationale of the alternative images design is presented.

Table 20: **Design rational of the images alternatives**

Task: Display image				
Style:	**Image**	**As text**	**As link**	**Resizable thumbnail**
Targets:	-	Facilitate screen reader and low vision users in order not to be in difficulties with image viewing	Facilitate screen reader and low vision users in order not to be in difficulties with image viewing but with the capability to save or view an image	Viewing images in small size with the capability to enlarge the image to normal size upon user request.

Parameters:	User(Default)	User (Blind or Low vision)	User (Blind or Low vision) and user preference)	User (preference)
Properties:	View image	Read image alternative text	Read image alternative text or and select linked named as the image alternative text to save or view the image	View image thumbnail and select it to view it in normal size
Relationships:	Exclusive	Exclusive	Exclusive	Exclusive

Example: Images displayers

Portals usually include lists of downloadable images. As presented in Figure 39, five alternatives artefacts were designed according to user web expertise (Table 21). For novice user images are presented as thumbnails along with a link that downloads images and a description of the estimated time to download the image. For moderate user the link is accompanied with the image size. Finally, images list for expert user consists of the link to download the image along with the image name, size and type.

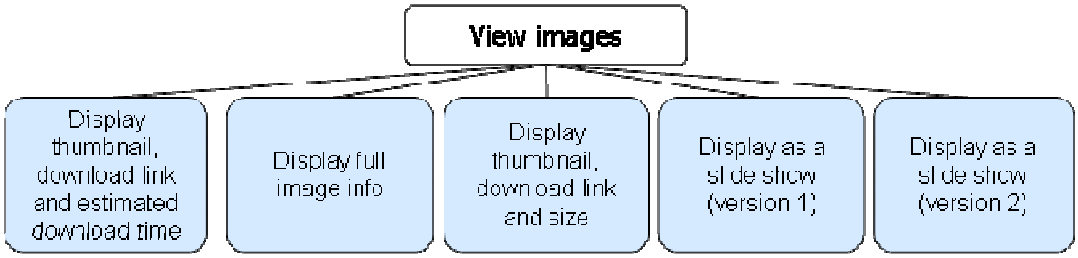


Figure 39: **Images representations**

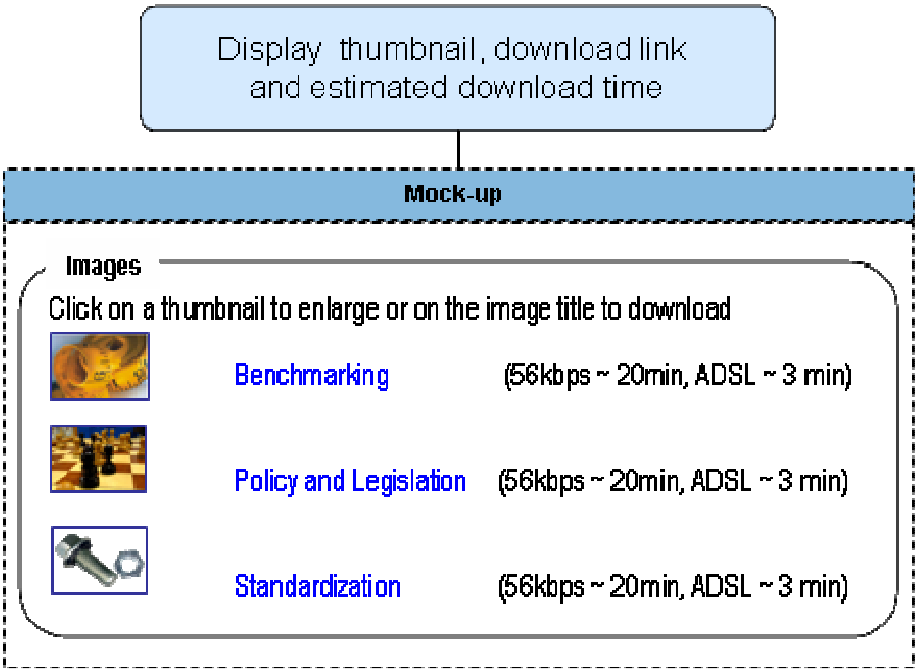


Figure 40: **Display thumbnail, download link and estimated download time**

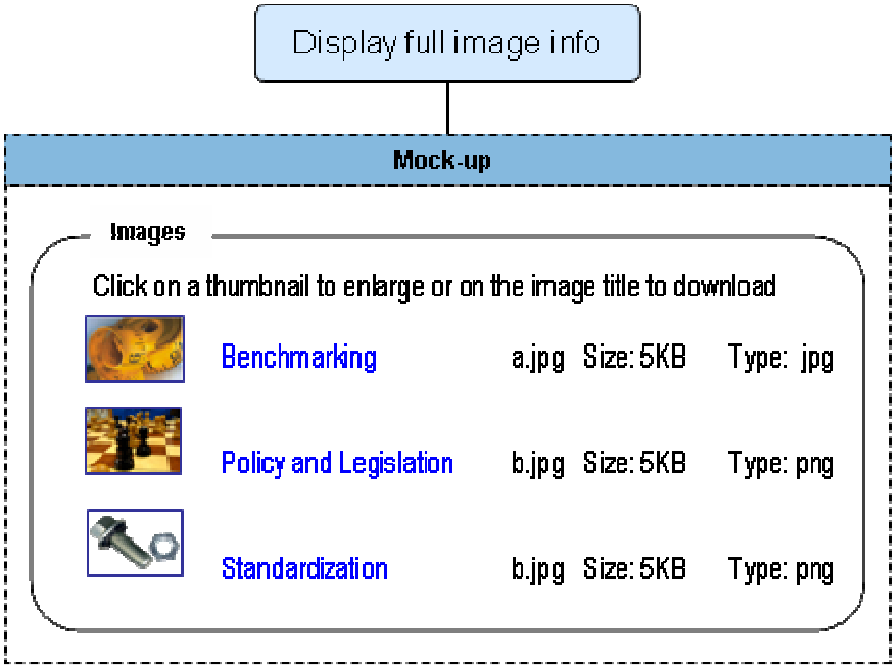


Figure 41: **Display full image info**

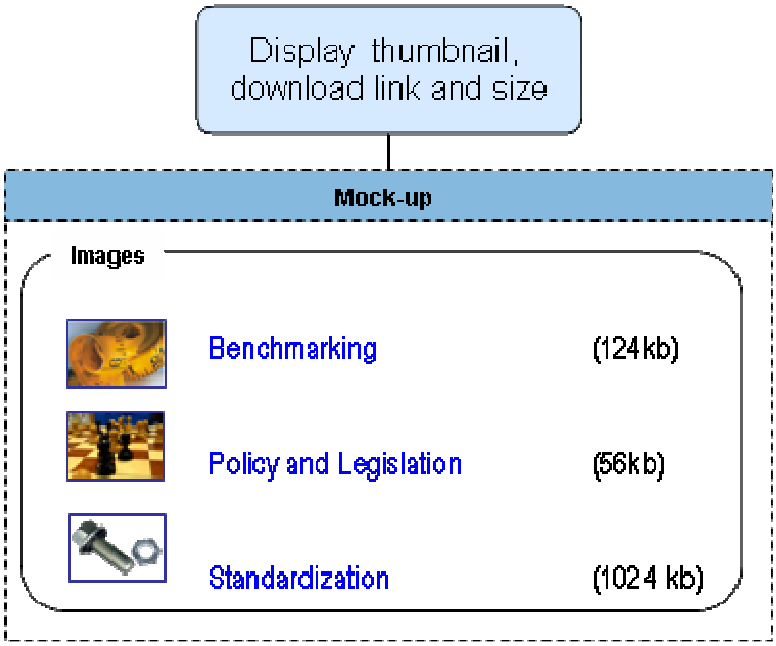


Figure 42: **Display thumbnail, download link and size**

In the three previous presentations the images are presented as thumbnails, along with information varying according to user web expertise. In the presentations that are shown in Figure 43 and Figure 44, a user that select these representations may view images in a greater size and navigate among them by clicking on image buttons or on links respectively.

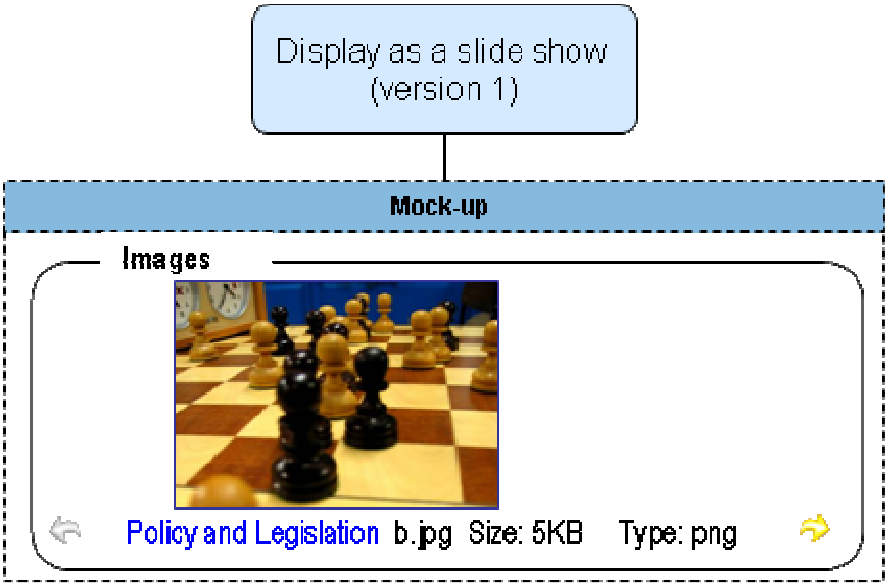


Figure 43: **Display as slide show (version 1)**

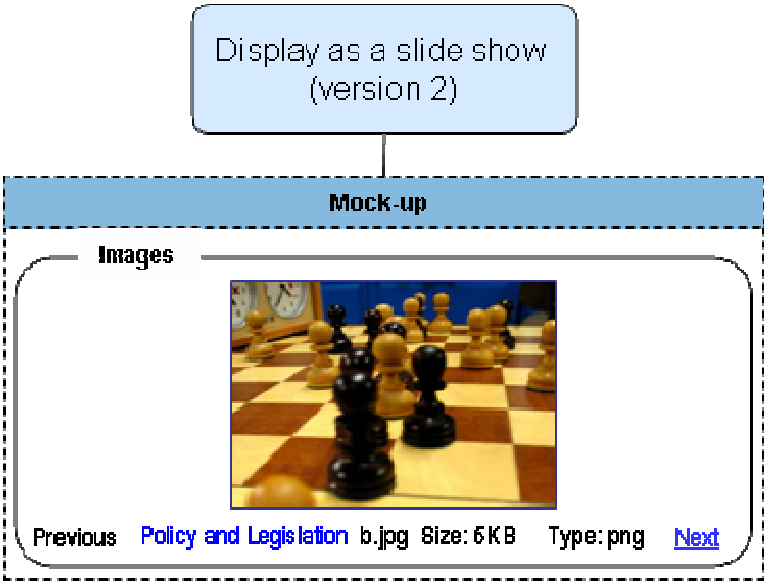


Figure 44: **Display as a slide show (version 2)**

Table 21: **Design rational of images representations**

Task: View images					
Style:	Display thumbnail, download link and estimated download time	Display full image info	Display thumbnail, download link and size	Display as a slide show (version 1)	Display as a slide show (version 2)
Targets:	Usability, flexibility	Usability, flexibility	Usability, flexibility	Usability, flexibility	Usability, flexibility
Parameters:	User (novice)	User (expert)	User (moderate)	User preferences	User preferences
Properties:	-	-	-	-	-
Relationships:	Exclusive	Exclusive	Exclusive	Exclusive	Exclusive

5.5.2 Layout-related alternative designs

Example: Template

A portal template generally maps to the generic scheme that incorporates the containers hosting contents. As presented in Figure 45, two generic template styles were designed. The linearized template style contains all the containers (top navigation, content, bottom navigation) in a linear form. On the other hand, the columns template style has three alternative styles where top and bottom navigation are placed on the top and bottom positions, and the middle container is split in two, three or four columns respectively for the two, three, four columns template.

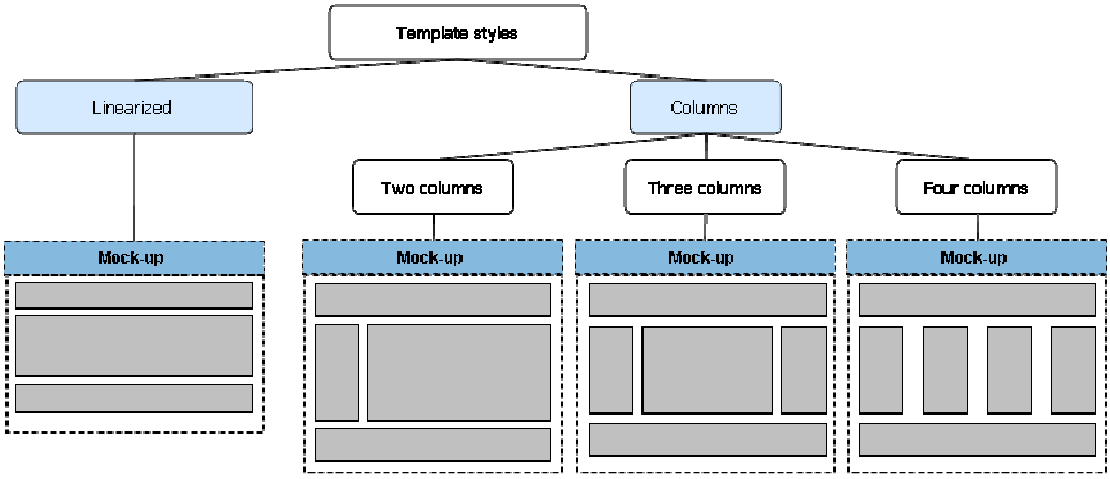


Figure 45: **Template representation styles**

According to the design rationale presented in Table 22, the linearized template supports speed, naturalness and flexibility for blind or low vision users, whereas the columns templates sustain speed, flexibility and optimum screen size for users with no visual impairments. The alternative columns templates are intended to be used in order to support content flexibility.

Table 22: **Design rationale of the template styles**

Task: Template styles		
Style:	Linearized	Columns
Targets:	Accessibility, speed, naturalness, flexibility	Speed, flexibility, cover optimum screen size
Parameters:	User (Blind, Low vision)	User (No visual impairments)
Properties:	-	-
Relationships:	Exclusive	Exclusive

Template size constitutes another significant aspect that is associated with the screen resolution in which the portal will be presented. According to [59], a web page has to be optimised for 1024x768 resolutions, but has to stretch well for any resolution, from 800x600 to 1280x1024 using a liquid layout. As presented in Figure Figure 46 and Table 23, the template size may be resized according the device screen resolution in order to cover the optimum screen size.

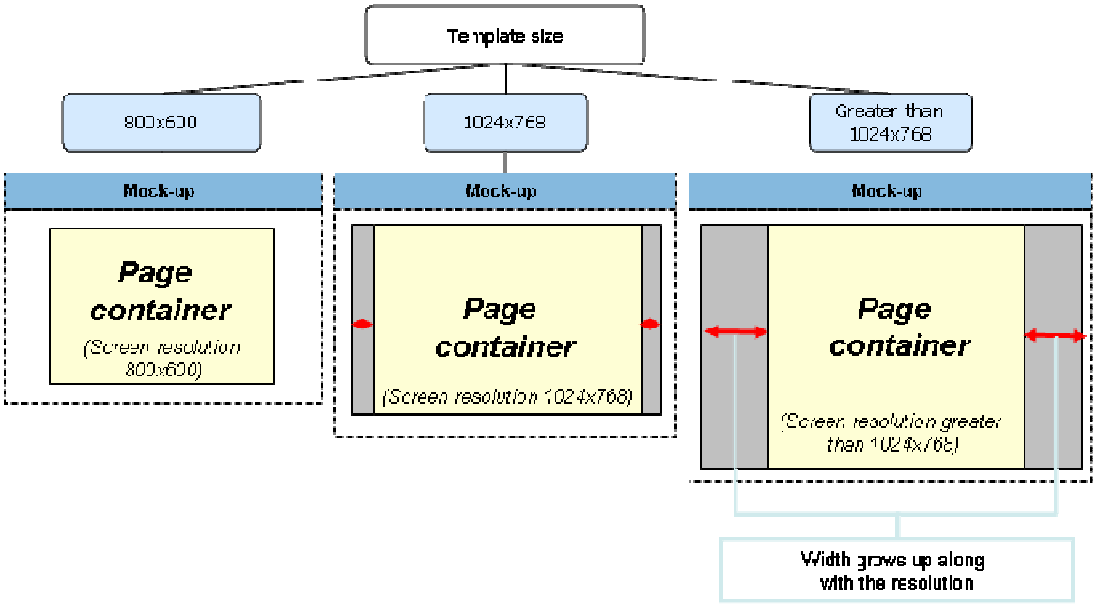


Figure 46: **Template alternatives according to device resolution**

When the resolution is 800x600, the template covers all the surface of the screen, whereas for 1024x768 resolutions the template lays has on its left and right sides a small unexploited area, in order to maximize readability of the contents. For resolutions greater than 1024x768, the width of the empty area left and right of the template is increased according to screen resolution.

Table 23: **Design rationale of the templates alternatives according to device resolution**

Task: Template styles			
Style:	800 x 600	1024 x 768	Greater than 1024 x 768
Targets:	Cover optimum screen size	Cover optimum screen size	Cover optimum screen size
Parameters:	Device resolution: 800 x 600	Device resolution: 1024 x 768	Device resolution: greater than 1024 x 768
Properties:	-	-	-
Relationships:	Exclusive	Exclusive	Exclusive

5.5.3 Navigation-related alternative designs

Example: Full navigation

Navigation constitutes one of the main mechanisms that a web portal user uses. Multiple alternatives of the navigation mechanism were designed in order to support individual user abilities and preferences. These are presented in Figure 47.

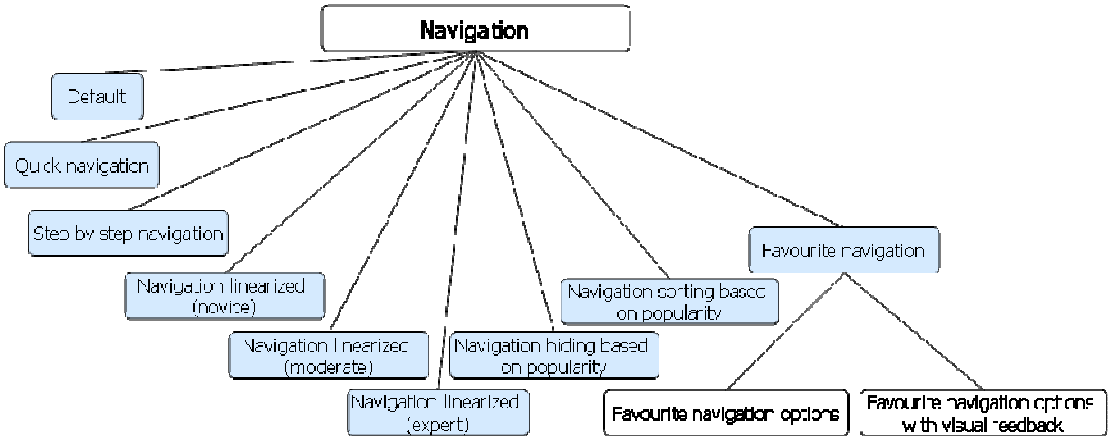


Figure 47: **Navigation alternatives**

The linearized navigation for novice users (see Figure 48) offers a linear form for all the navigation links of the portal, and in parallel a step by step navigation is supported. Initially, the user has to select among navigation hierarchies, next among entire navigation elements and finally among entire navigation sub-elements. In each step, the previous hierarchy is available in order to navigate back to another navigation hierarchy or navigation element. This step by step navigation mechanism offers a guided navigation to novice users with vision impairments, in order to enhance accessibility, flexibility and usability of the portal.

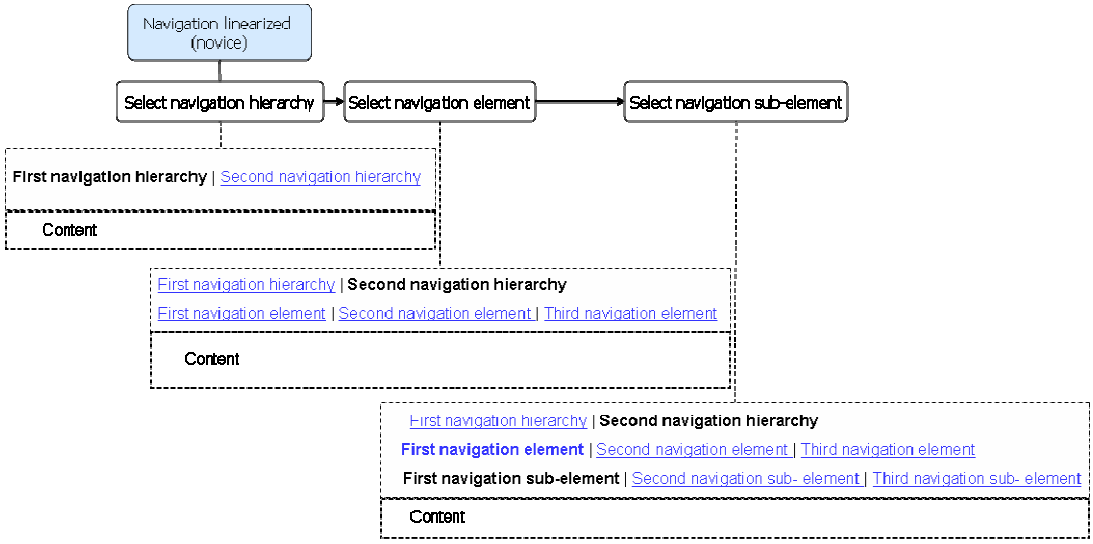


Figure 48: **Navigation linearized (novice)**

The linear navigation targeted for moderate with visual impairments, supports a linear form of the entire navigation of the portal. Initially, the user selects among navigation hierarchies and then the available navigation elements for the selected navigation hierarchy are presented, along with a navigation path through which the user may navigate back to the navigation hierarchy. Through this procedure the user has to scan limited navigation options using the screen reader, knows each time which page are browsed, and always has an efficient way to navigate back to the navigation hierarchies thanks to the path mechanism (see Figure 49).

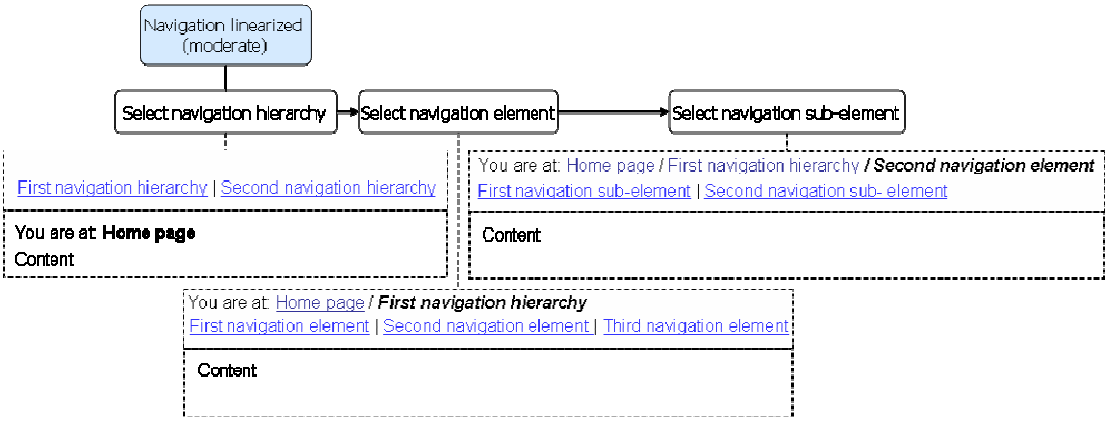


Figure 49: **Navigation linearized (moderate)**

The linear navigation for expert users with visual impairments resembles the linear navigation for moderate users, but without the path mechanism. In this way, the expert has the ability to navigate back to the navigation hierarchy but is not notified about the web page browsed each time (see Figure 50). In this way, the expert can browse through the navigation mechanism quickly, without having the screen reader always reading the path of the entire page.

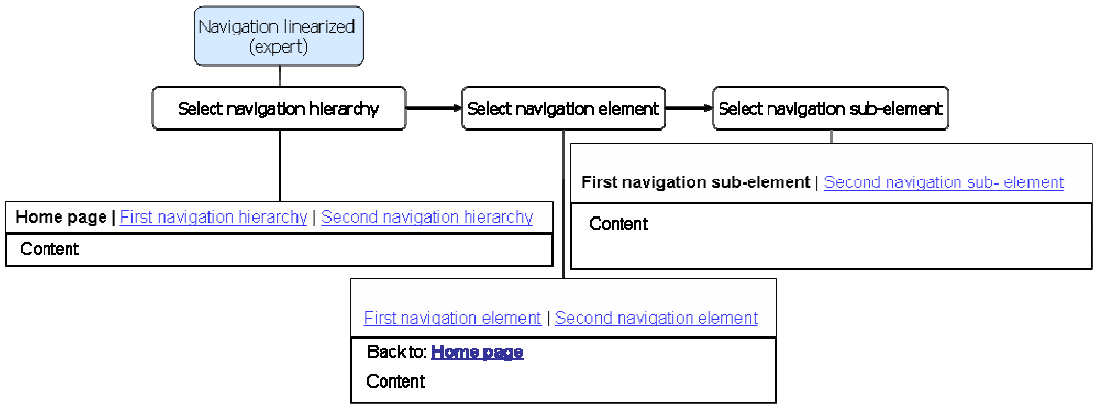


Figure 50: **Navigation linearized (expert)**

Table 24: **Design rationale of the step by step navigation (novice, moderate, expert)**

Task: Navigation			
Style:	Navigation linearized (novice)	Navigation linearized (moderate)	Navigation linearized (expert)
Targets:	Accessibility, flexibility, usability	Accessibility, flexibility, usability, limited reading by the screen reader	Accessibility, speed, flexibility, usability, limited reading by the screen reader
Parameters:	User (Blind or Low vision and novice web expertise)	User (Blind or Low vision and moderate web expertise)	User (Blind or Low vision and expert web expertise)
Properties:	Navigation hierarchy first, navigation element next (for desired navigation hierarchy), navigation sub-element next (for	Navigation hierarchy first, navigation element next (for desired navigation hierarchy), navigation sub-element next (for	Navigation hierarchy first, navigation element next (for desired navigation hierarchy), navigation sub-element next (for desired navigation

	desired navigation element)	desired navigation element)	element)
Relationships:	Exclusive	Exclusive	Exclusive

5.5.4 Interaction-related alternative designs

Example: File up-loader

Uploading files constitutes a frequently used function for web users. As it is shown in Figure 51 to Figure 54, three alternative designs were produced targeted to expert, moderate and novice users in order to upload and delete files.

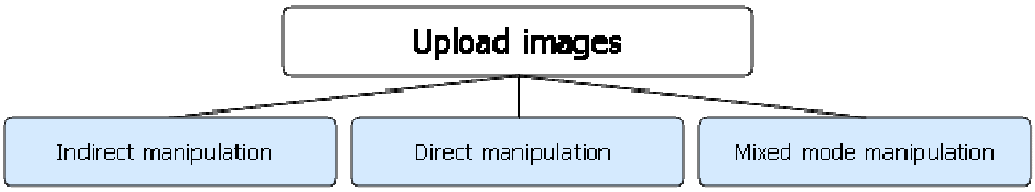


Figure 51: **Upload files alternatives**

As it is shown in Figure 52, a novice user in order to upload a file has to complete several simple steps; firstly, the user has to press button 'add new', then the interface changes, and the user has to complete three simple steps, browse a file, type a title and push the button 'upload'. A progress bar with the file upload time appears. To delete an uploaded file, the user has to press button 'delete' and then to check the files to delete and press again the 'delete' button. The described presentation is targeted to novice users because it contains simple and detailed steps.

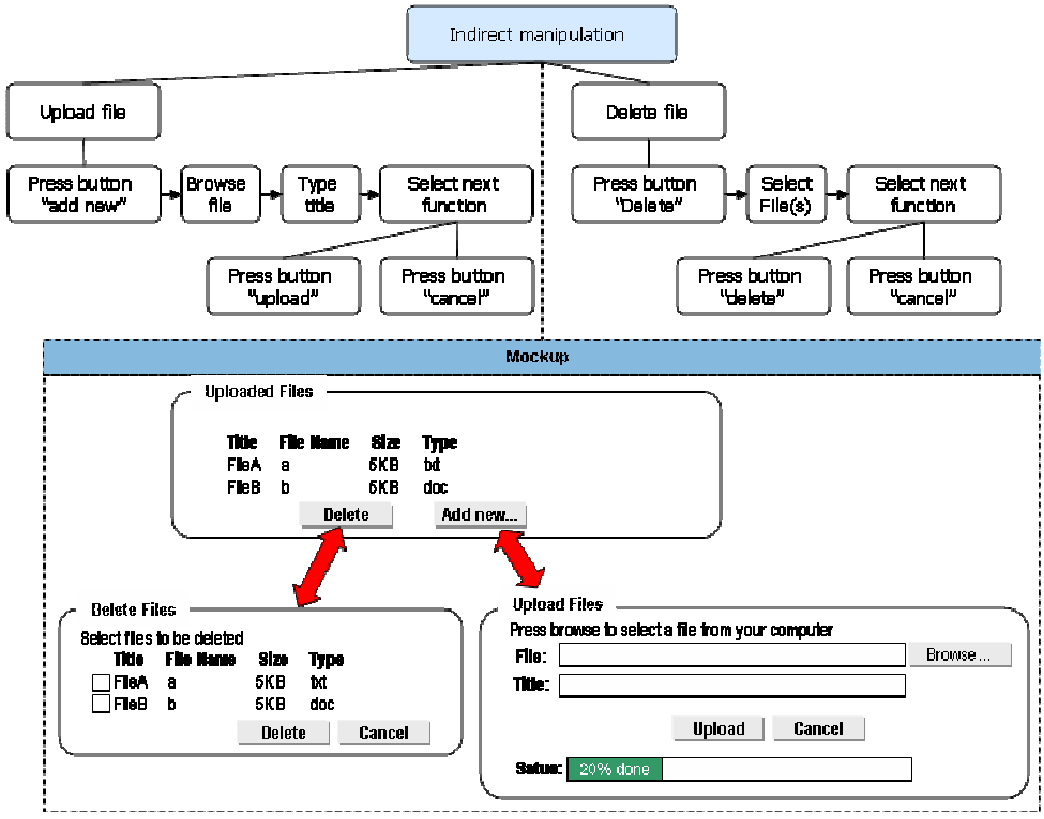


Figure 52: **File upload (indirect manipulation)**

The direct manipulation representation is designed for expert users. All the functions are lying in a single interface in order to be accessed by the user quickly and effectively. The user has only to browse a file, type a title and press the button 'add' in order to upload a file. On the other hand, in order to delete a file that has already been uploaded, the user has only to select the file or files and press button 'delete'.

For moderate users, an intermediate design (between novice and expert user design) was prepared that includes two interfaces: one to upload files and another to view uploaded files and delete files these that were uploaded by accident. The moderate user in order to upload a file has to press the button 'add', then automatically a second interface appears where the user browses a file, types a title and finally presses button 'upload'. A moderated user in order to delete a file uploaded by accident has only to check the file or files to be deleted and then to press the button 'Delete'.

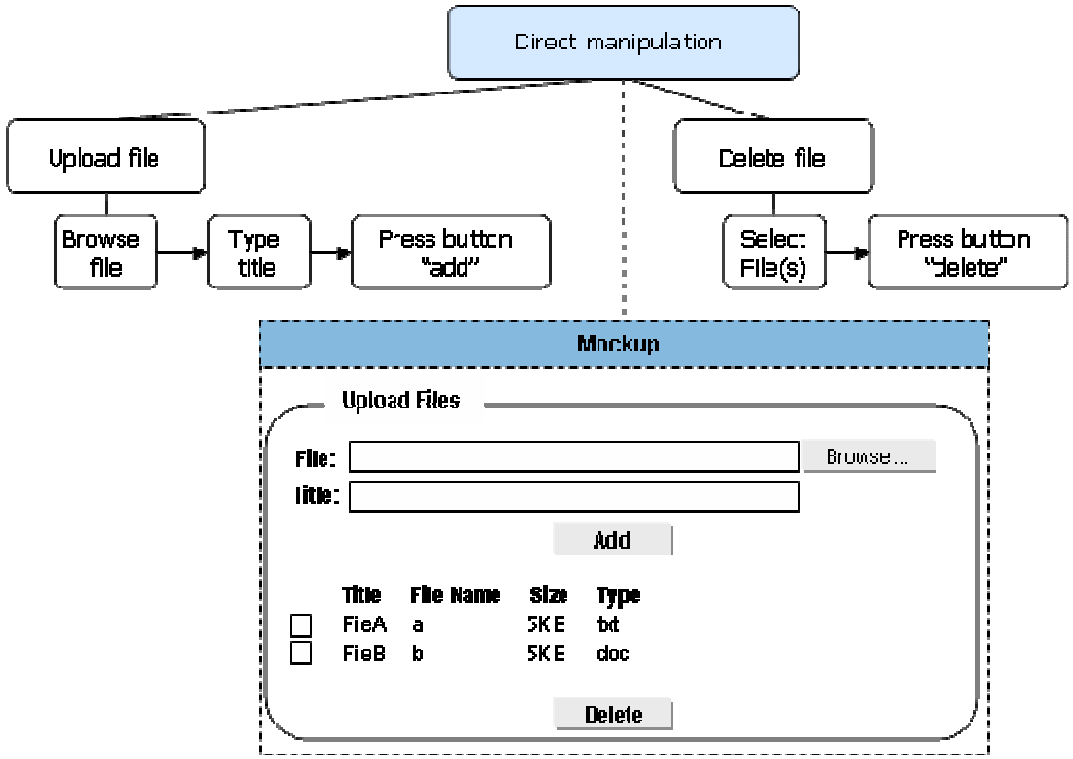


Figure 53: **File upload (direct manipulation)**

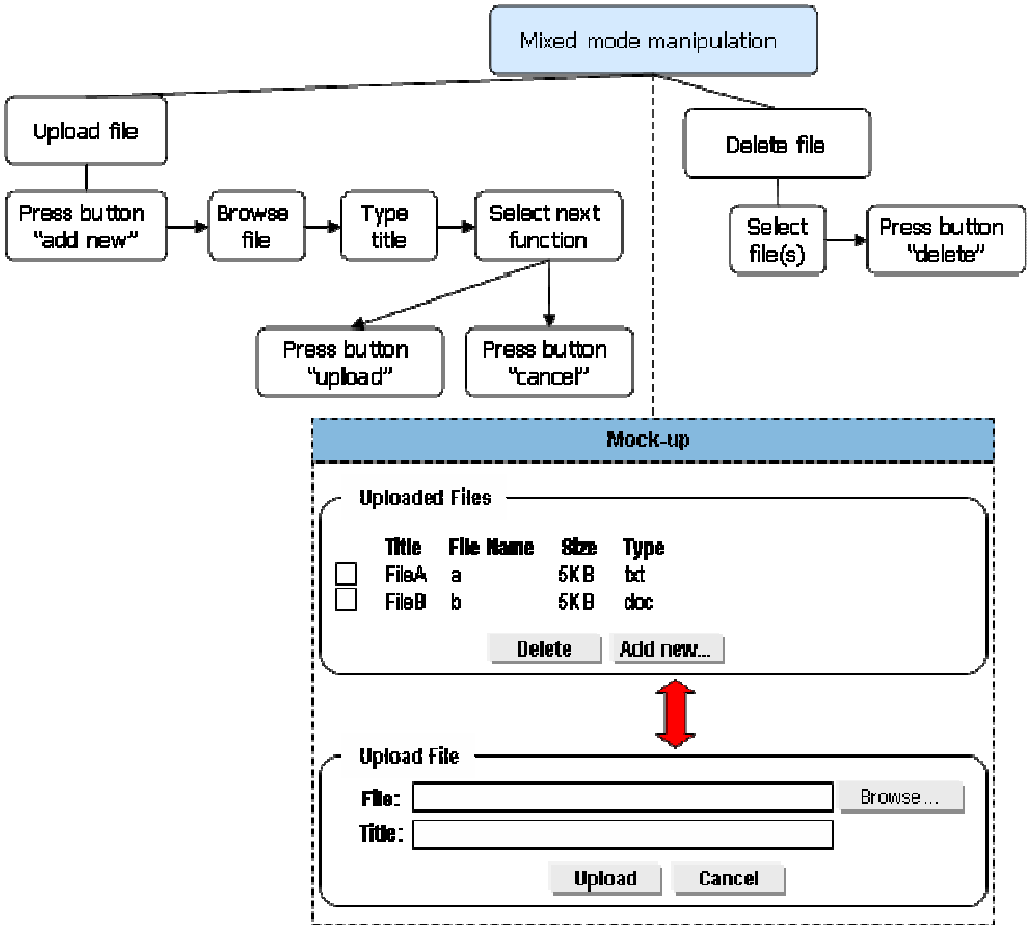


Figure 54**: File upload (mixed mode manipulation)**

Table 25: **Design rationale of the upload files alternatives**

Task: Upload files			
Style:	Indirect manipulation	Direct manipulation	Mixed mode manipulation
Targets:	Simplicity, guided steps	Speed, effectiveness	Guides steps, effectiveness, usability
Parameters:	User (novice)	User (expert)	User (moderate)
Properties:	Upload file: Press button 'add new' first, browse file next, type title next, press button 'upload' Delete file: Press button 'delete' first, select file(s) next, press button 'delete'	Upload file: Browse file first, type title next, press button 'add Delete file: Select file(s) first, press button 'delete' next	Upload file: Press button 'add new' first, browse file next, type title next, press button 'upload' Delete file: Select file(s) first, press button 'delete' next
Relationships:	Exclusive	Exclusive	Exclusive

Example: Date entry

Date entry constitutes a frequent task for web users in case of registering an event to the web portal or in case of searching for data based on dates. Five alternatives were designed for date entry, and are presented in Figure 55. As recorded in Table 26, the first design includes three drop downs where user has to select three items (year, month, day) using dropdowns. These dropdowns are constructed in such a way that

when a user selects a specific year and month, the appropriate calculations are made based on leap years and number of days that each month includes, in order that only the valid days are placed in the days' dropdown. This option is targeted to novice users because of the simplicity and error prevention that offers.

The second design looks like the first design, with the difference that the validation of the user input is made manually. The user after the selection of the appropriate values has to push the button 'ok' in order to validate the selected date. In case of a mistake, an error in red appears guiding the user to correct the mistake. This design artefact is targeted to users with visual impairments, since it does not necessitate extended screen reading and is simple in use.

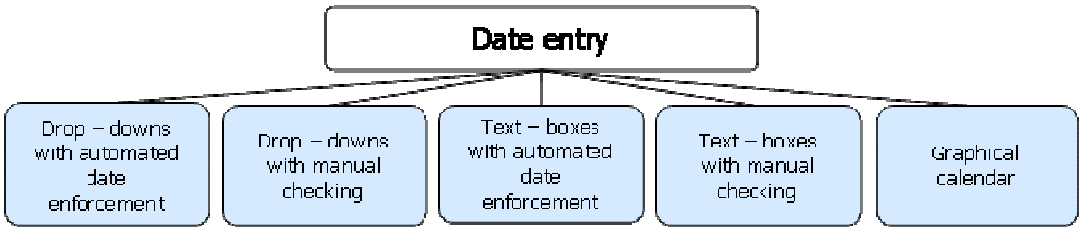


Figure 55: **Date entry alternatives**

As shown in Figure 58 to Figure 57, in textboxes design with automated date enforcement design the user has to type year, month and day in the textboxes. The validation of the date would take place when the user will try to complete the particular action. This date input alternative is designed for expert web users to improve the speed and the effectiveness of the use. There is another design intended to be used by users who have visual impairments and are web experts too. Simply, in this design the user has to push the button 'ok' to validate the date inserted. For moderate and motor impaired users, another design was produced that offers a virtual calendar from where the user has monthly previews, may navigate through years and months using two dropdowns and can select a date by clinking on it when the appropriate date is met. This design is characterized as suitable for motor impaired users because it does not necessitate much user input, and is suitable for moderate users too because it offers a calendar-like graphical representation of dates that is more familiar to the users.

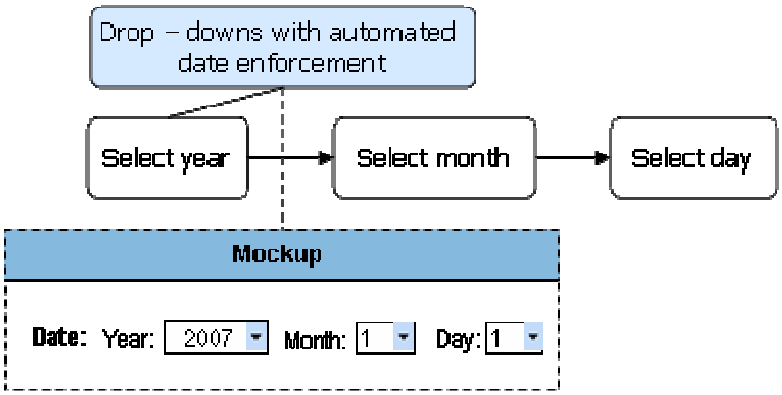


Figure 56: **Date entry (drop-downs, automated date enforcement)**

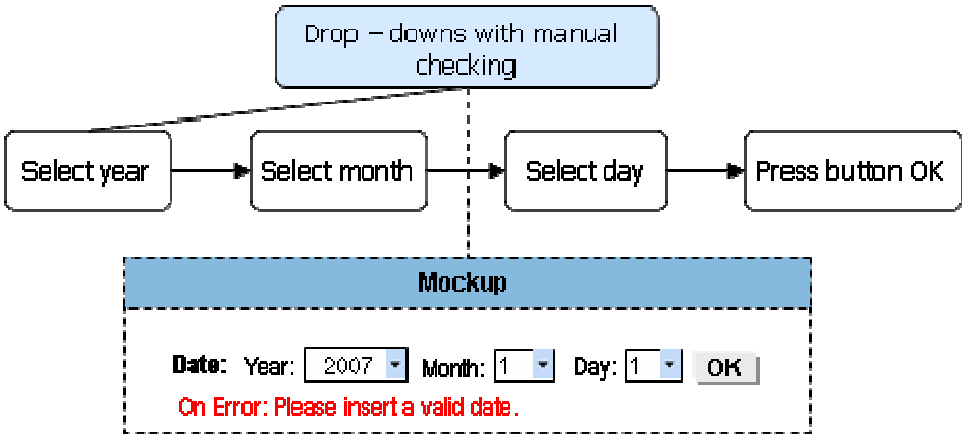


Figure 57: **Date entry (drop-downs)**

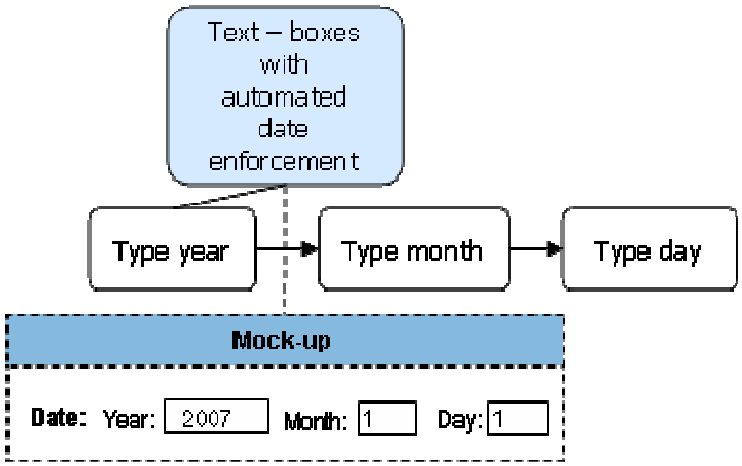


Figure 58: **Date entry (text – boxes, automated date enforcement)**

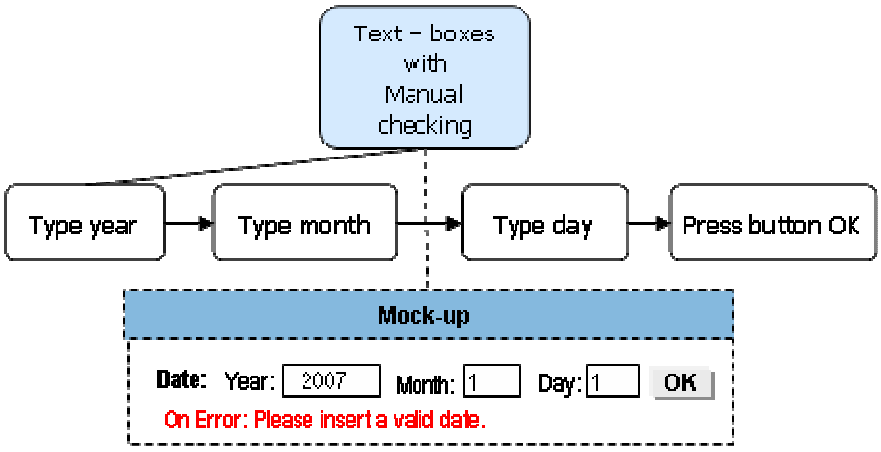


Figure 59: **Date entry (text – boxes, manual checking)**

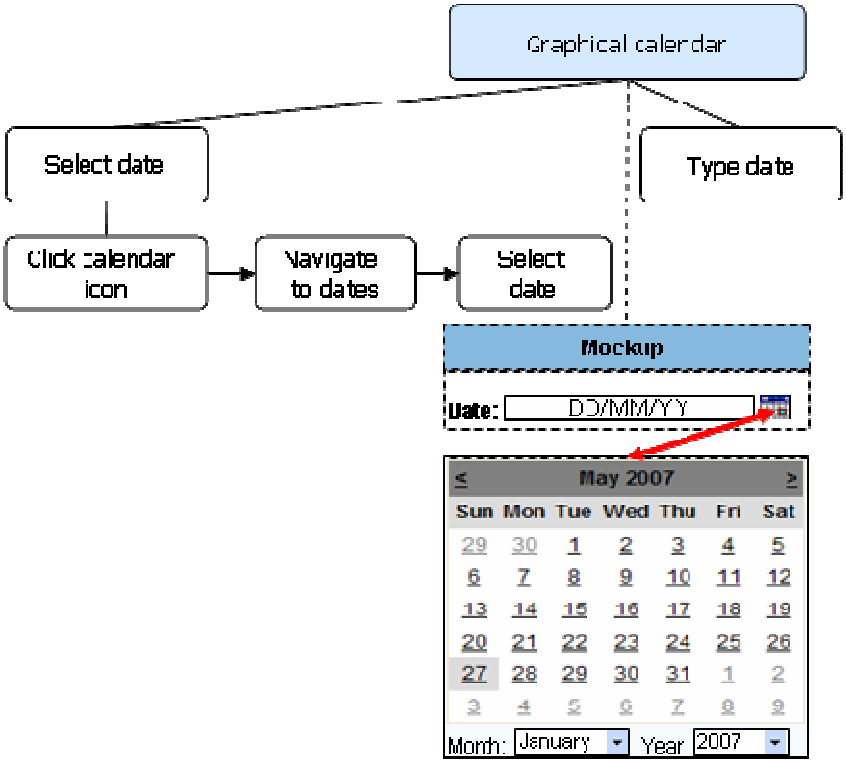


Figure 60: **Date entry (text box & graphical calendar)**

Table 26: **Design rationale of date input alternatives**

Task: Date Entry					
Style:	Drop downs with automated date enforcement	Drop down with manual checking	Text boxes with automated date enforcement	Text boxes with manual checking	Graphical calendar
Targets:	Simplicity, error prevention easiness	Facilitate screen reader, effectiveness	Speed, effectiveness	Facilitate screen reader, effectiveness	Limited necessity of user input, simplicity
Parameters:	User (novice)	User (Blind or Low vision)	User (expert)	User ((Blind or Low vision) and expert)	User (Moderate, motor impaired)
Properties:	Select year first, month next and day at last	Select year first, month next, day next and press button 'ok' at last	Type year first, month next and day at last	Type year first, month next, day next and press button 'ok' at last	Type day first, '/' next, month next, '/' next, year at last or click on 'calendar' icon and select date on the virtual calendar
Relationships:	Exclusive	Exclusive	Exclusive	Exclusive	Exclusive

Chapter 6
EAGER VS. VISUAL STUDIO® ALONE

The aim of this Chapter is to elucidate the benefits of employing the EAGER toolkit instead of the Microsoft's Visual Studio alone, in terms of the developer's performance and efficiency. In the following subsection, an example of how to develop a webpage for posting messages using the standard .Net framework is compared to the process followed when using the EAGER toolkit. The initial steps that are necessary for both approaches are presented in section 6.1.

Subsequently, the development process for developing the aforementioned webpage using the standard .Net framework are presented in section 6.2. Finally, the process for developing the same webpage using the EAGER toolkit is presented in order to outline the expected empowerment of the developer, together with the support provided for performing user interface abstraction via user task oriented development.

6.1 Initial development stages

In order to develop a Web application using the ASP.NET, the following are considered as minimum prerequisites:

- An operating system supporting ASP .NET: Windows 2000 (Professional, Server, and Advanced Server) or Windows XP Professional or Windows Server 2003
- Internet Information Services
- .NET Framework version 1.1
- Microsoft Visual studio 2003
- Microsoft SQL Server

Visual Studio is the standard Development Environment provided by Microsoft for building applications using .Net technologies. Visual Studio provides a developer friendly environment together with a vast number of facilities such as syntax highlighting, advanced debugging facilities, etc. For providing an overview of the tasks involved in developing a web application using the aforementioned facilities, this sections walks-trough on how to create, build and run a simple web page, for example a page that supports posting a topic on a message board. The development language used to build the aforementioned example is C# along with the standard ASP .NET Web UI controls Library.

The first step introduced in the development process is the creation of a new web application. This is achieved by selecting File → New → Project in the Visual Studio 2003 IDE. This brings up the 'New project' dialog that is presented in Figure 61, where clicks on the 'Visual C#' node in the tree-view on the left hand side of the dialog box and choose the 'ASP.NET Web Application' icon. Next, the developer types the name of the project and hits the button OK.

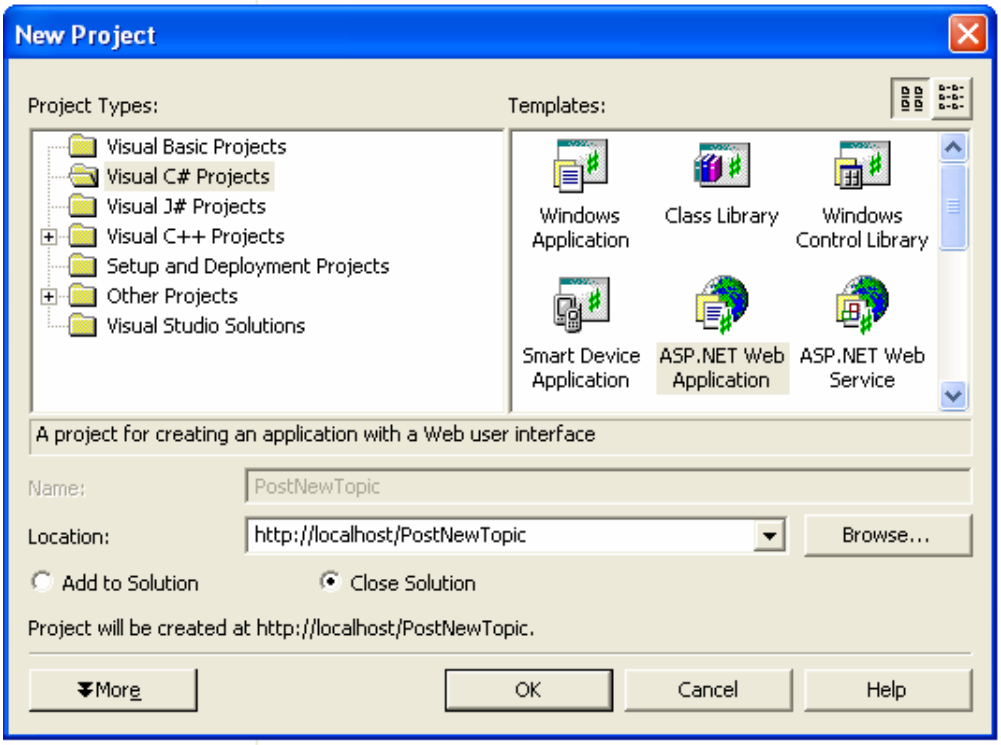


Figure 61: **Visual studio: The New Project dialog**

Visual Studio will then create and open a new web project within the solution explorer. By default, it will have a single page (WebForm1.aspx), an AssemblyInfo.cs file, a Global.asax file, as well as a Web.config file (see Figure 62). All project file-meta-data is stored within an MSBuild based project file.

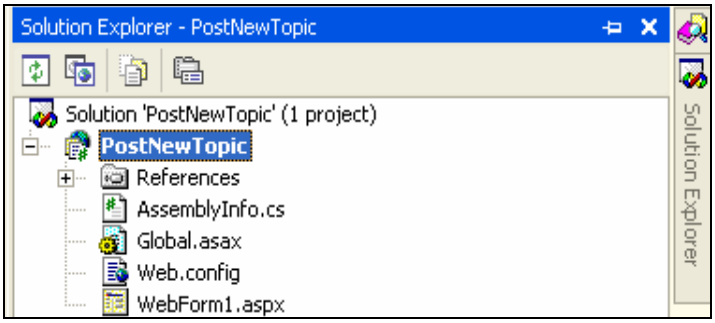


Figure 62: **Visual studio: New project solution explorer**

6.2 Using Microsoft's .Net technologies alone

Step 1 - Design the User Interface

Firstly a mock-up of the page to be developed is designed. Figure 63 presents a mock-up that includes a title and a description field that are required along with the date entry module and a module for attaching files to topic. Files are attached using the browse button in order to locate the file, and the attach button to upload the located file. The field-set topic files are used to present the uploaded files. If a file was uploaded by mistake, the user can delete it by checking it and then by pressing the delete button.

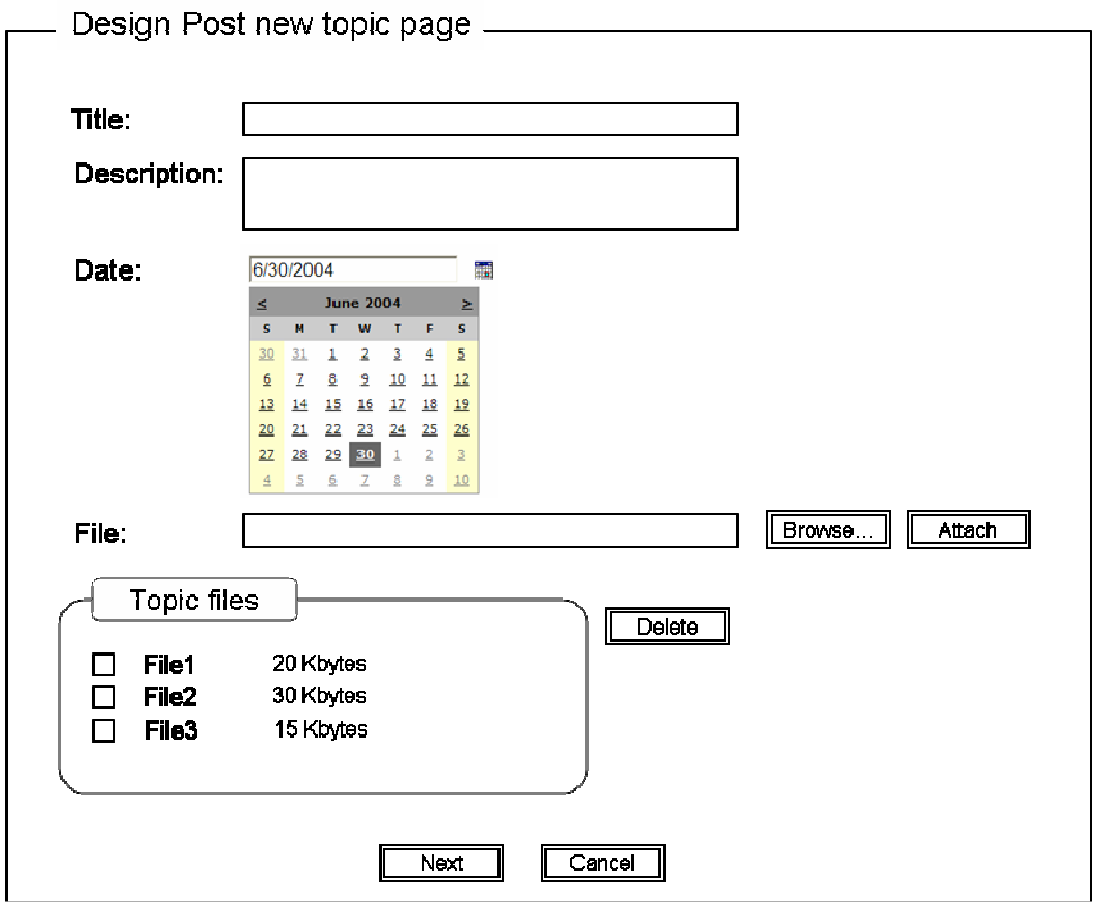


Figure 63: **Mock-up of the page**

Step 2 - Writing ASP. Net mark-up

When the web page has been designed, the next step is the process of writing ASP .Net mark-up that renders the controls that have been designed. Figure 64 includes the mock-up fields for title, description, file uploader and date along with their ASP .Net representation. Figure 65 presents the ASP .Net mark-up that is required in order to render the field 'Topic files', the list of uploaded files and the buttons 'Next' and 'Cancel'.

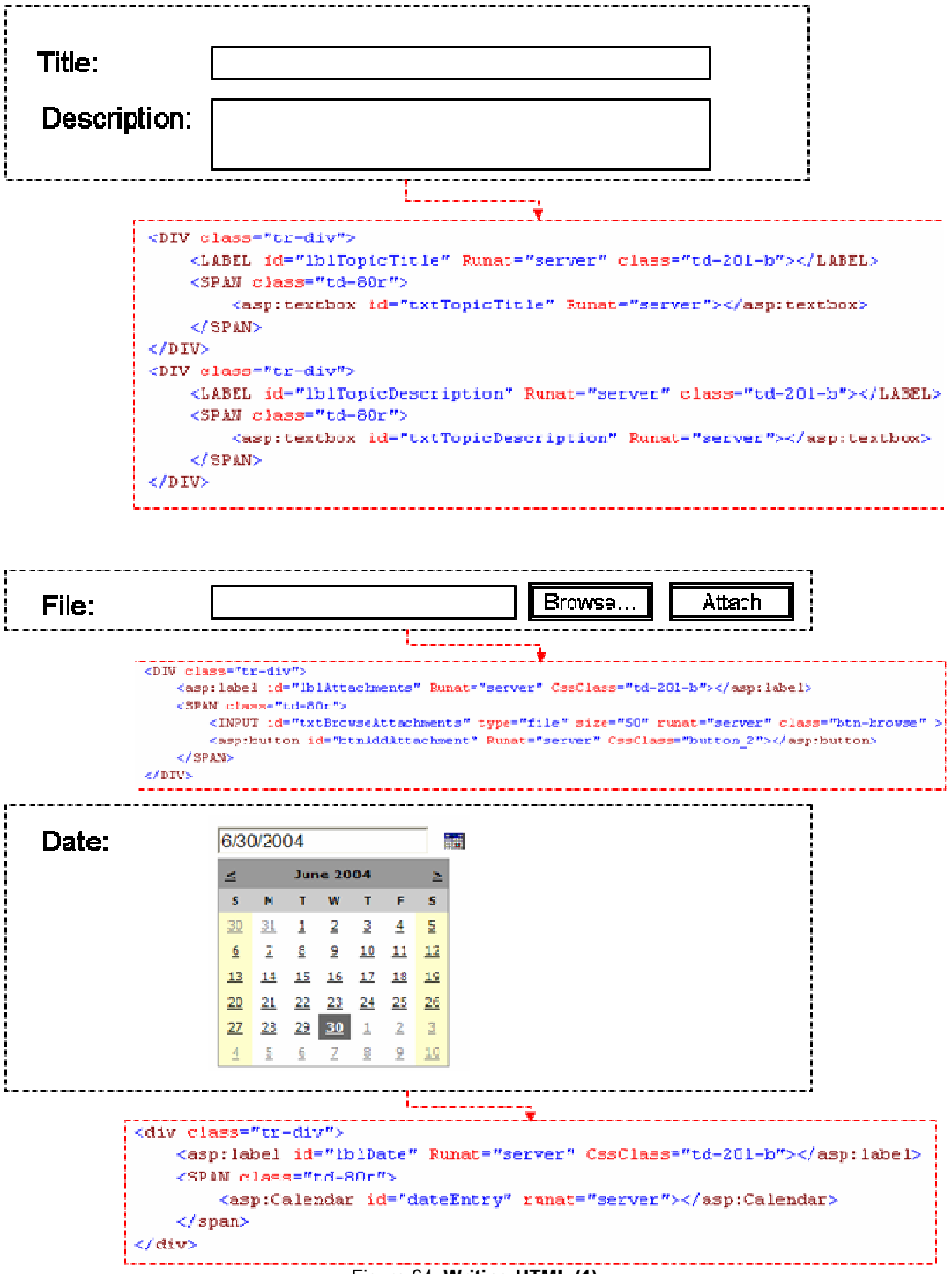
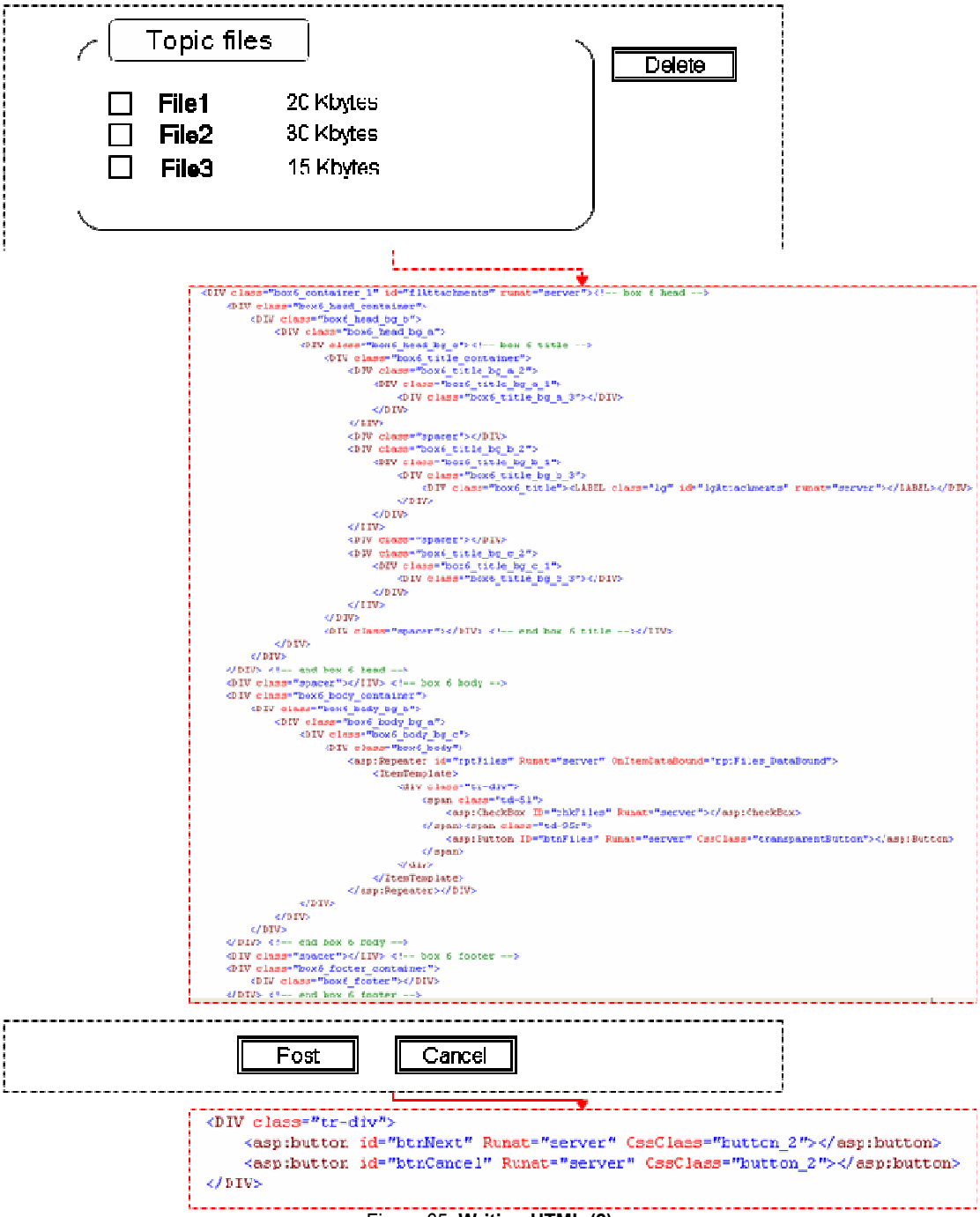


Figure 64: **Writing HTML (1)**




```
<DIV class="box6_container_1" id="filtAttachments" runat="server"><!-- box 6 head -->
    <DIV class="box6_head_container">
        <DIV class="box6_head_bg_b">
            <DIV class="box6_head_bg_a">
                <DIV class="box6_head_bg_c"><!-- box 6 title -->
                    <DIV class="box6_title_container">
                        <DIV class="box6_title_bg_a_2">
                            <DIV class="box6_title_bg_a_1">
                                <DIV class="box6_title_bg_a_3"></DIV>
                            </DIV>
                        </DIV>
                        <DIV class="spacer"></DIV>
                        <DIV class="box6_title_bg_b_2">
                            <DIV class="box6_title_bg_b_1">
                                <DIV class="box6_title_bg_b_3">
                                    <DIV class="box6_title"><LABEL class="lg" id="lgAttachments" runat="server"></LABEL></DIV>
                                </DIV>
                            </DIV>
                        </DIV>
                        <DIV class="spacer"></DIV>
                        <DIV class="box6_title_bg_c_2">
                            <DIV class="box6_title_bg_c_1">
                                <DIV class="box6_title_bg_c_3"></DIV>
                            </DIV>
                        </DIV>
                    </DIV>
                    <DIV class="spacer"></DIV> <!-- end box 6 title --></DIV>
                </DIV>
            </DIV>
        </DIV>
    </DIV> <!-- end box 6 head -->
    <DIV class="spacer"></DIV> <!-- box 6 body -->
    <DIV class="box6_body_container">
        <DIV class="box6_body_bg_b">
            <DIV class="box6_body_bg_a">
                <DIV class="box6_body_bg_c">
                    <DIV class="box6_body">
                        <asp:Repeater id="rptFiles" Runat="server" OnItemDataBound="rp:Files_DataBound">
                            <ItemTemplate>
                                <div class="tr-div">
                                    <span class="td-S1">
                                        <asp:CheckBox ID="chkFiles" Runat="server"></asp:CheckBox>
                                    </span><span class="td-S2r">
                                        <asp:Button ID="btnFiles" Runat="server" CssClass="transparentButton"></asp:Button>
                                    </span>
                                </div>
                            </ItemTemplate>
                        </asp:Repeater></DIV>
                </DIV>
            </DIV>
        </DIV>
    </DIV> <!-- end box 6 body -->
    <DIV class="spacer"></DIV> <!-- box 6 footer -->
    <DIV class="box6_footer_container">
        <DIV class="box6_footer"></DIV>
    </DIV> <!-- end box 6 footer -->
```

```
<DIV class="tr-div">
    <asp:button id="btnNext" Runat="server" CssClass="button_2"></asp:button>
    <asp:button id="btnCancel" Runat="server" CssClass="button_2"></asp:button>
</DIV>
```

Figure 65: **Writing HTML (2)**

Step 3 - Adding functionality to the controls

Subsequently, functionality has to be added to the controls in order to function properly. The functionality that is added to controls varies and includes assigning text to labels, setting visibility to controls, filling repeater of files, deleting files, storing data to database, etc. In Figure 66 an example of the functionality that is required in order to upload files (topic attachments) is presented. Figure 67 includes the necessary code in order to fill in the repeater representing the uploaded files, and Figure 68 the code to delete undesirable files.

```
private void btnAddAttachment_Click(object sender, System.EventArgs e)
{
    string tmpFileName= txtBrowseAttachments.PostedFile.FileName.ToString();
    if(!tmpFileName.Equals(""))
    {
        string fileName = tmpFileName.Substring( tmpFileName.LastIndexOf( "\\" ) + 1 );
        string path = "./MessageBoard/Files/";
        string filePath = Server.MapPath( path + fileName );

        System.IO.DirectoryInfo newFolder = new System.IO.DirectoryInfo( Server.MapPath(path));
        if ( !newFolder.Exists )
        {
            newFolder.Create();
        }

        if(!txtBrowseAttachments.Value.Equals(""))
        {
            txtBrowseAttachments.PostedFile.SaveAs( filePath );
            string[] files = attached_files.Value.Split(',');
            string[] sizes = string_size.Value.Split(',');

            for (int i=0;i<files.Length;i++)
            {
                repeaterItems.Add(files[i] + "(" + sizes[i] + ")");
                ItemsSizes.Add(sizes[i]);
                ItemsNames.Add(files[i]);
            }
        }
        else
        {
            lblErrorTopic.Text = interfaceText[13].ToString();
            pErrorTopic.Visible = true;
        }
    }
    else
    {
        divAtError.Visible = true;
        lblAtError.Text = interfaceText[22].ToString();
    }
}
```

Figure 66: **Code for uploading file**

```
protected void rptAttachments_DataBound(object sender, System.Web.UI.WebControls.RepeaterItemEventArgs e)
{
    string attachments = (string)e.Item.DataItem;
    string[] sizes = string_size.Value.Split(',');
    int i;

    Button btnAttachments = (Button)e.Item.FindControl( "btnAttachments" );
    HyperLink lnkAttachment = (HyperLink)e.Item.FindControl("lnkAttachment");

    ListItemType itemType = e.Item.ItemType;
    if ( itemType == ListItemType.Item ||
        itemType == ListItemType.AlternatingItem)
    {
        btnAttachments.CommandName = "File";
        btnAttachments.CommandArgument = attachments;
        btnAttachments.Text = attachments + "(" + sizes[Convert.ToInt32(hdSizeValue.Value)] + ")";
        i = Convert.ToInt32(hdSizeValue.Value) + 1;
        hdSizeValue.Value = i.ToString();
    }
}

private void rptAttachments_ItemCommand(object source, System.Web.UI.WebControls.RepeaterCommandEventArgs e)
{
    string path = "MessageBoard/Files/" + e.CommandArgument.ToString();
    UploadDownload.DownloadFile(Server.MapPath(path),Response);
}
```

Figure 67: **Code for presenting uploaded files**

```
private void btnRemoveFiles_Click(object sender, System.EventArgs e)
{
    ArrayList filenames = new ArrayList();
    ArrayList sizes = new ArrayList();
    ArrayList names = new ArrayList();

    foreach( RepeaterItem ri in rptFiles.Items)
    {
        CheckBox chkFiles = (CheckBox)ri.FindControl("chkFiles");
        Button btnFiles = (Button)ri.FindControl( "btnFiles" );
        if (chkFiles.Checked)
        {
            filenames.Remove(btnFiles.Text);
            string path = "./MessageBoard/Files/";
            string filePath = Server.MapPath( path + temp1 );
            System.IO.DirectoryInfo folder = new System.IO.DirectoryInfo( Server.MapPath(path));
            if (folder.Exists)
            {
                System.IO.FileInfo[] files = folder.GetFiles();

                for (int k=0; k<files.Length;k++)
                {
                    if(files[k].Name.Equals(temp1))
                    {
                        files[k].Delete();
                    }
                }
            }
            names.Remove(temp1);
            sizes.Remove(temp2);
        }
    }
    rptFiles.DataSource = filenames;
    rptFiles.DataBind();
}
```

Figure 68: **Code to remove uploaded files**

Step 4 - Building and running the project

When the developer has incorporated all the appropriate functionality, the Web application can be run by hitting button ‘F5’, and the results are viewed in the browser.

6.3 Using EAGER [in combination with .Net]

As discussed in the previous chapters, the EAGER framework supports the development of web applications that adapt their UI elements to meet the requirements set by user and context specific attributes. In this section, the methods and techniques used for building a post topic page using the EAGER framework are presented.

Step 1- Design the User Interface

The design of a web page for posting topic demands different logic when it is intended to be developed with the EAGER toolkit. The design of such a web page is task oriented as long as the DCC discussed in section 5.4 includes several alternative User Interface components which encapsulate functionality for several tasks. Figure 69 includes two text entry components, a date entry component, multiple files entry component and a component for functions to be applied.

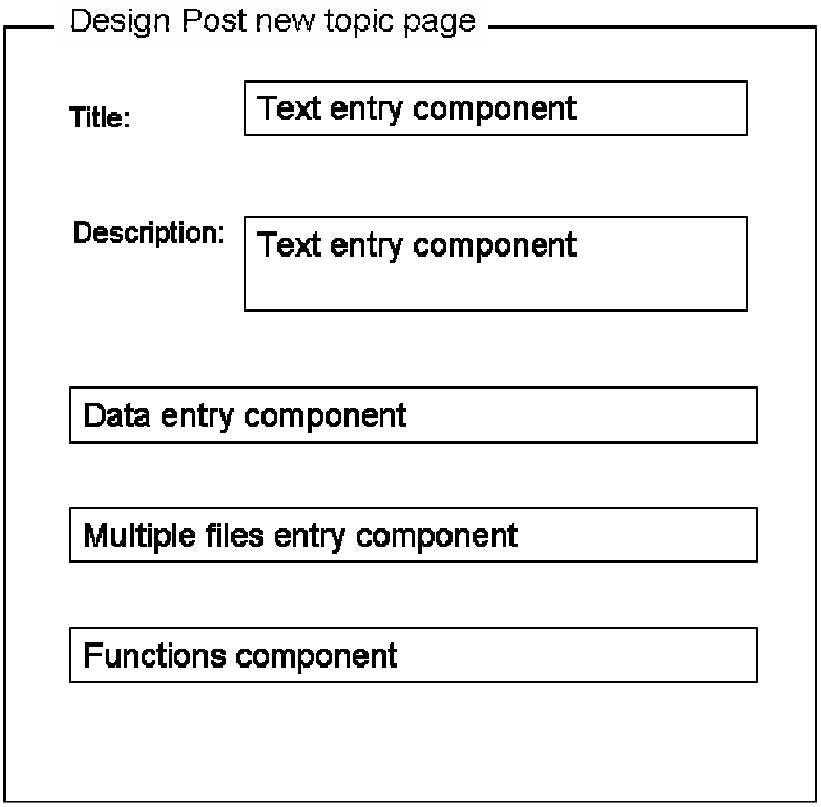


Figure 69: **Task oriented mock-up of the page**

Step 2 - Writing EAGER mark-up

Following the design phase of the Web page, the subsequent step includes the process of writing EAGER mark-up that renders the components that have been designed. Figure 70 includes all the components included in the mock-up, along with their EAGER mark-up code. As an example, 'ics:icsDatePicker' defines a date entry control which may be transformed to alternative user interfaces such as dropdowns which contain year, month, day or textboxes for filling in year, month and day or a graphical calendar for clicking on date.

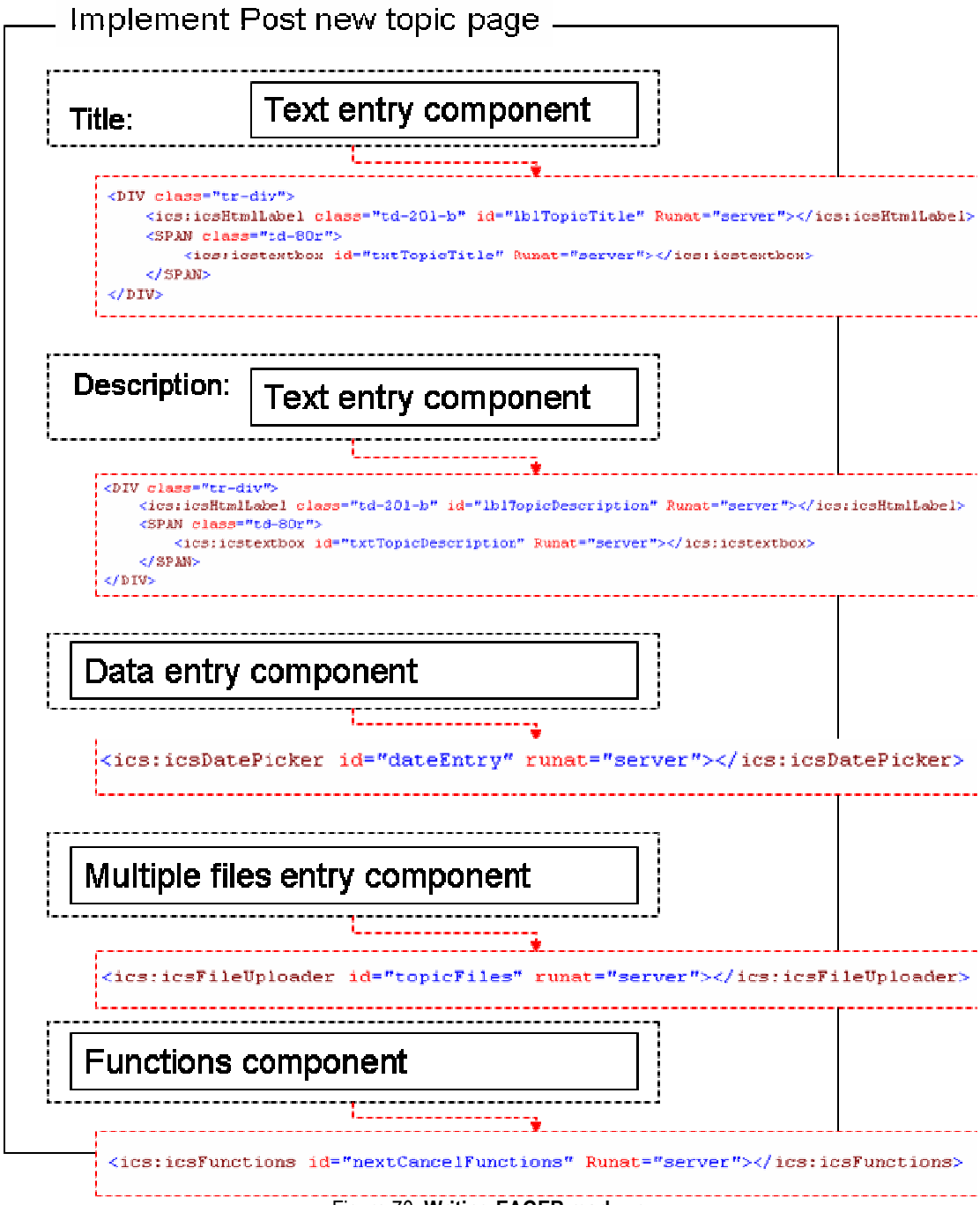


Figure 70: **Writing EAGER mark-up**

Step 3 - Adding functionality to the controls

At this stage, functionality has to be added to the controls in order for them to function properly. The functionality that is needed to be added to EAGER controls is radically reduced compared to the code that is required for the ASP .Net controls. Figure 71 represents all the code that is required in order for the module to support uploading - presenting files and deleting files. To configure the component for file uploading four variables are used, 'sourcePath' is used along with the 'sourceID' to combine the temporary location where the files will be saved and the 'fPath' along with the 'destID' to combine the final location where the files have to be saved. Source path and final path can be the same.

```
topicFiles.sourcePath = "~/MessageBoard/Files/temp";
topicFiles.fpath = "~/MessageBoard/Files/Message";
topicFiles.sourceID = userId;
topicFiles.destID = topicId;
```

Figure 71: **Code for uploading file**

Step 4 - Building and running the project

When the developer has incorporated all the appropriate functionality, the Web application can be run by hitting button 'F5' and the results are viewed in the browser. In Figure 72, some of the alternative representations that may appear when the Web application runs, depending on end user characteristics and the context of use, are presented.

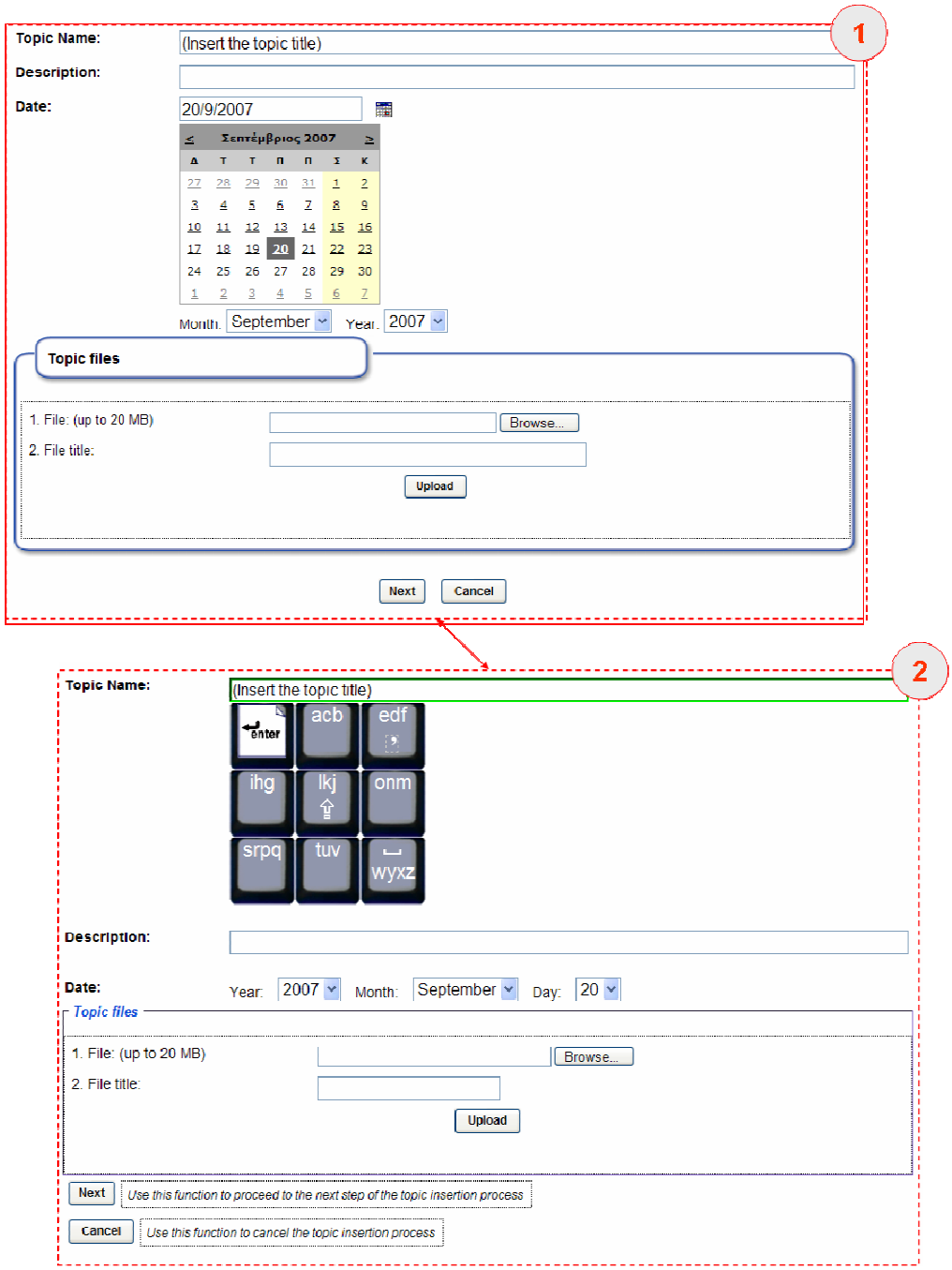


Figure 72: **Post topic (two examples of alternative representations)**

In the first alternative representation, simple textboxes appear along with a graphical calendar and field-set and a simple file up loader. In the second option, the textbox changes colour on focus and is offered along with a virtual keyboard. Date entry uses three dropdowns for year, month and day respectively. Finally, the functions 'Next' and 'Cancel' are provided with a small description next to each of them.

6.4 Porting EAGER into existing Web applications

The EAGER toolkit is not only a versatile tool for developing UWI from scratch. In case a portal developed by means of Microsoft's Visual Studio, EAGER can be ported in order to incorporate adaptation and improve the user-experience of end-users. This transformation can bring great benefits in a number of directions including accessibility, usability and the ability to serve diverse user requirements.

It is worthy to notice that, when the EAGER toolkit was ported into an existing web-based module named 'Ariadne' (see section 7.2.2.3, pp.157) that was originally developed using Visual Studio, the resulting total **number of code lines was significantly reduced by 50%** (see Figure 73).

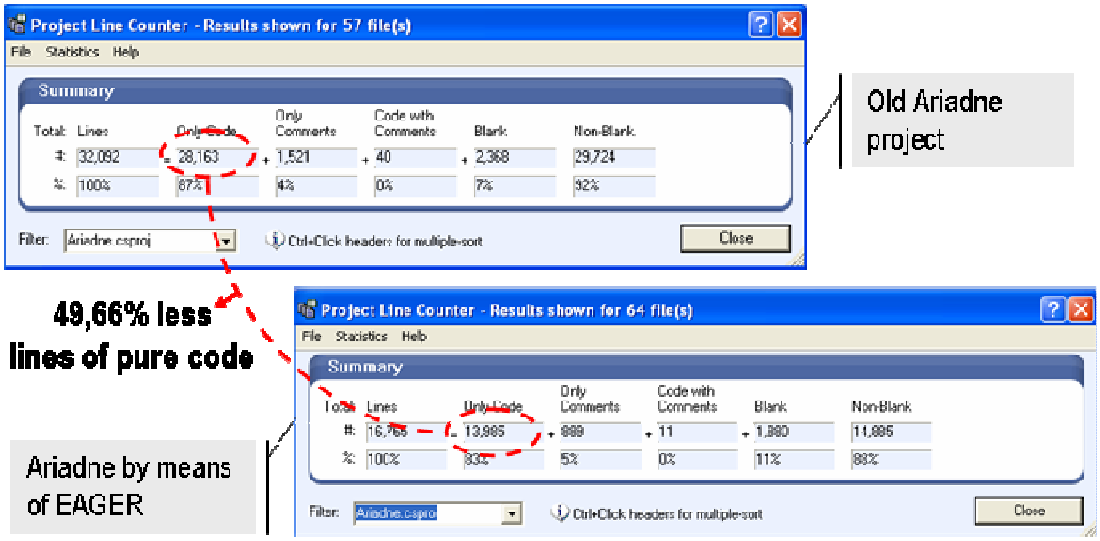


Figure 73: **Developing Ariadne with Visual Studio alone vs. with the EAGER toolkit**

The process of porting EAGER into existing Web portals typically introduces the following steps:

- EAGER setup: Involves the process of setting up the database schemes – Web services used by the EAGER toolkit for storing and retrieving user and context specific parameters.
- Import EAGER UI toolkit: Add a new reference to the EAGER toolkit to each project contained in the old portal.
- Import EAGER administrative facilities: Add the EAGER Profile Selection UI Module, Profiles Administration UI module and Statistics UI Module projects to the solution.
- Analyse existing application interfaces and identify functionality that can be abstracted and therefore replaced by the EAGER equivalents (such as file uploading code snippets, paging facilities, etc).
- Use the EAGER toolkit primitive controls instead of the build in ASP.NET controls (for example use the EAGER label instead of the ASP.NET label).

6.5 Conclusions

The benefits gained by using the EAGER toolkit lie on a number of dimensions, including:

- The time required for designing a web application and the detailed of design information needed.
- The time required for designing the front end of the application to be used by end users.
- The developer effort for setting up the application.

As discussed above, in the process of designing a simple form for posting topics, the complexity of the UI design effort is radically reduced due to the flexibility provided by the EAGER toolkit for designing interfaces at an abstract task-oriented level. Using EAGER, designers are not required to be aware of the low level details introduced in representing interaction elements, but only of the high level structural representation of a task and its appropriate decomposition into sub tasks, each of with represents a basic UI and system function.

On the other hand, the process of designing the actual front end of the application using a mark-up language is radically decreased in terms of time, due to the fact that developers initially have to select a radically increased number of interface components each of witch representing a far more complex facility. Additionally, developers do not have to spend time for editing the presentation characteristics of the high level interaction element, due to the internal styling behaviour.

The actual process of transforming the initial design into the final Web application using traditional UI controls introduces a lot of coding in conjunction the higher level elements offered by the EAGER toolkit. Here it is worthy to notice that the EAGER toolkit currently consists of a total number of 55K pure code lines at the availability of Web portal developers (see Figure 74). Furthermore the incorporation of EAGER's higher level elements make a portal's code more usable, more readable and especially safe, due to the fact that each interaction component introduced by this framework is designed separately, developed and tested introducing a high level of code reuse, efficiency and safety.

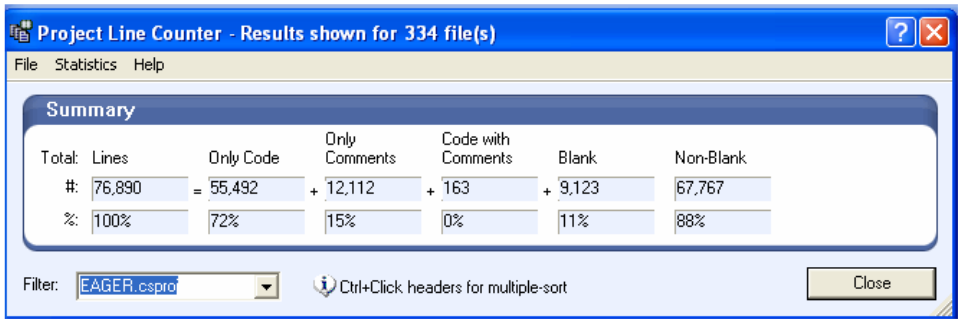


Figure 74: **Total number of lines of pure code constituting the EAGER toolkit**

In conclusion, the EAGER toolkit offers, not only the aforementioned benefits, but also opens a more promising direction. UIs developed with the EAGER toolkit can adapt according to specific user and context parameters, and therefore are rendered in a number of variations. It is therefore clear that using a standard UI toolkit a monolithic interface is created, whereas using the EAGER toolkit dynamically adaptable interfaces are generated.

Chapter 7
THE EDEAN CASE STUDY

This chapter presents a prototype portal developed, as proof-of-concept, following the UWI methodology by means of the proposed EAGER toolkit. In order to elucidate the benefits of EAGER, an already existing portal was selected and redeveloped from scratch. In this way, it was possible to identify and compare the advantages of using EAGER, both at the developer's site, in terms of developer's performance, as well as at the end-user site, in terms of the user-experience improvement. In particular, the original portal of the *European Design for All e-Accessibility Network* (EDeAN), namely Hermes[29], was redesigned and implemented using the EAGER development framework. The following sections provide an overview of the features and development lifecycle of the new EDeAN portal[30].

7.1 HERMES

HERMES (Figure 75) was originally implemented in the context of the European Project D4ALLnet[31]in order to facilitate and support the virtual networking activities of the EDeAN *Special Interest Groups* (SIGs). The EDeAN SIGs are in effect online communities that share common interests and goals in the field of *Design for All* (DfA). The EDeAN SIGs have gathered experts and actors in the various fields related to DfA, and have created discussion fora for the exchange of information within each of the groups. The ARIADNE DfA virtual Resource Centre developed by the project and acts as a catalyst in making DfA knowledge widely available, and aims at fostering shared best practices, knowledge and experience on DfA.

Figure 75: **The old HERMES portal**

[29] Old version of the EDeAN portal: http://www.edean.org

[30] Current, temporary address of the new EDeAN portal: http://hci-web.ics.forth.gr/edean/

[31] Http://www.d4allnet.gr/

7.2 The new EDeAN Portal

Aiming to continue the success of its predecessor (HERMES), the new Portal was designed and developed in the context of the Dfa@eInclusion[32] Project. The new version of the Hermes portal introduces a number of major improvements in terms of functionality, and is the first portal developed using a versatile fully reusable framework for the development of Unified Web Interfaces. All the modules already existing in the previous version and the newly added modules were altered to use the interaction artefacts offered by the EAGER development framework.

In Figure 76, we see that the total number of lines of the programming code constituting the EDeAN portal.

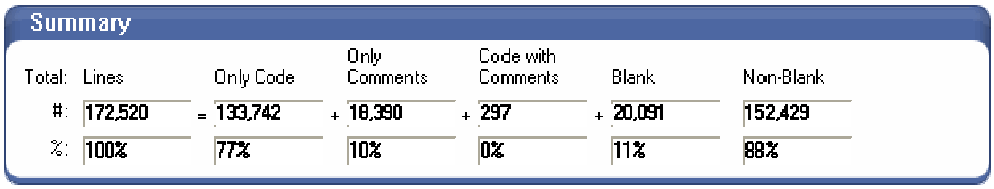


Figure 76: **Total number of pure code constituting the new EDeAN portal**

7.2.1 Public area

The purpose of the public area is to disseminate information about the scope, objectives and outcomes of the networking activities. From the portal public area (Figure 77) a number of facilities can be accessed such as:

- Information about EDeAN
- Resources from the Ariadne Resource Centre
- News and announcements
- Frequently Asked Questions
- Statistics regarding the networking activities
- Surveys for collecting user feedback

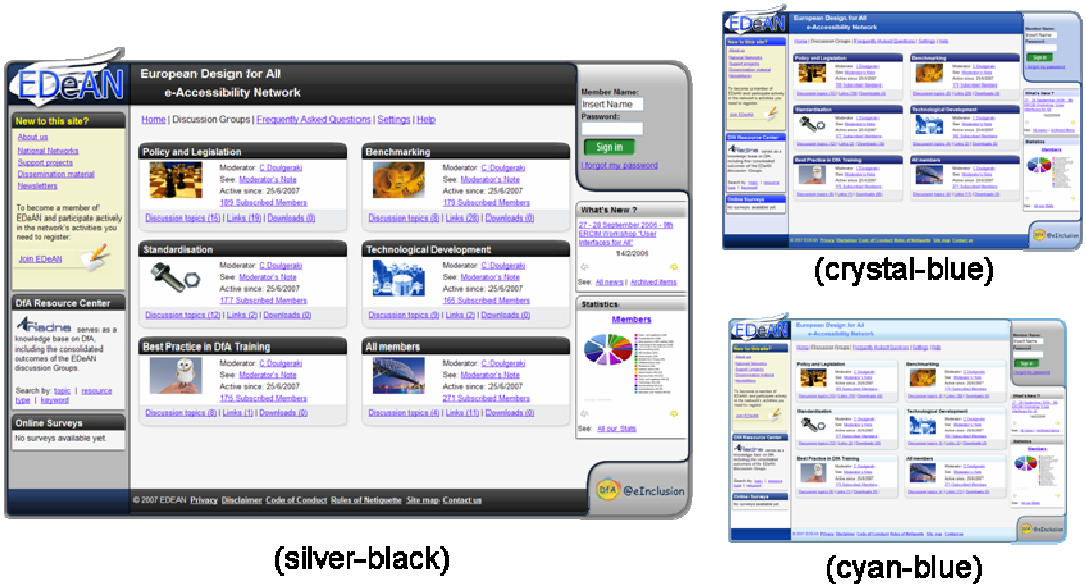


Figure 77: **The public area (various skins)**

[32] http://www.dfaei.org/

7.2.1.1 General settings

The ability to adapt the portal interface to the user interaction and accessibility requirements is not intended to support only the portal subscribed users. On the contrary, visitors of the Hermes portal have the option to access their Settings and alter them accordingly in order to map their personal characteristics and the characteristics of the context of use. This can be achieved through the interface presented in Figure 78.

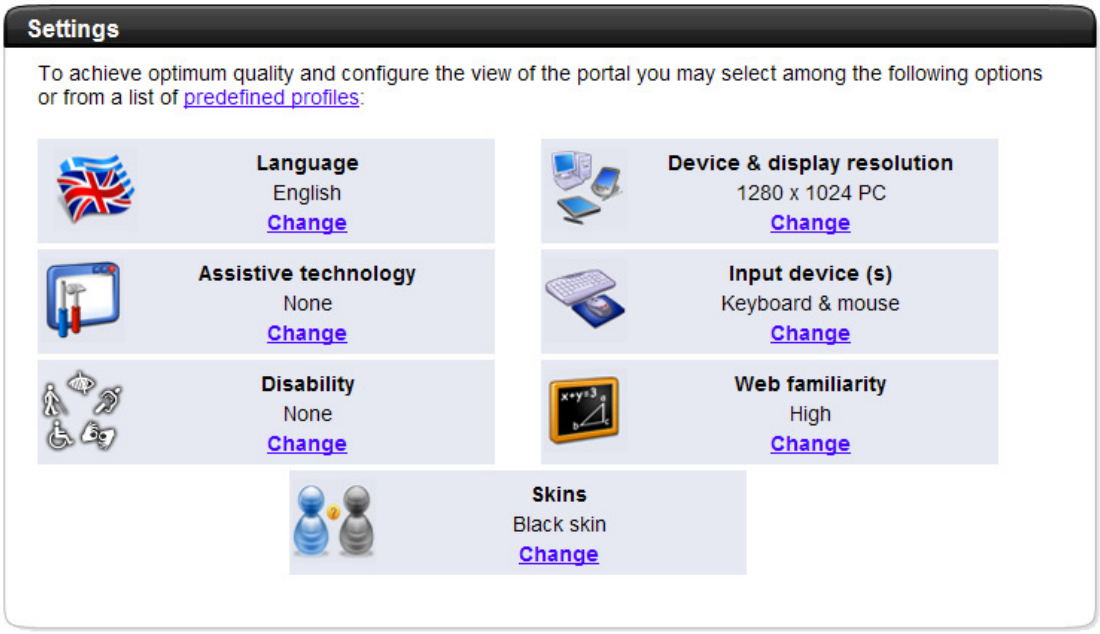


Figure 78: **Accessibility and Interaction options for portal visitors**

Using this interface a number of parameters can be set, such as:

- **Language:** Altering this setting affects the language used for rendering the portal content
- **Device & display resolution:** Altering this setting affects the presentation of the portal to meet the device presentation requirements
- **Assistive technology:** Altering this setting affects the facilities offered by the portal in order to facilitate the existence or not of a specific assistive technology
- **Input Device:** Altering this setting affects the facilities offered by the portal in order to facilitate the use of specific input devices
- **Disability:** Altering this setting affects the overall presentation characteristics of the portal to meet the requirements set by the selected disability
- **Web familiarity:** Altering this setting affects the interaction elements and help facilities offered to portal users to address the need of the selected expertise

After altering these settings, the user can bookmark the resulting page and in that way obtain a personalized experience without the need for subscription. Additionally, in order to allow users to quickly alter their settings, the quick settings option is offered.

Using this option, a number of predefined profiles is presented, as shown in Figure 79.

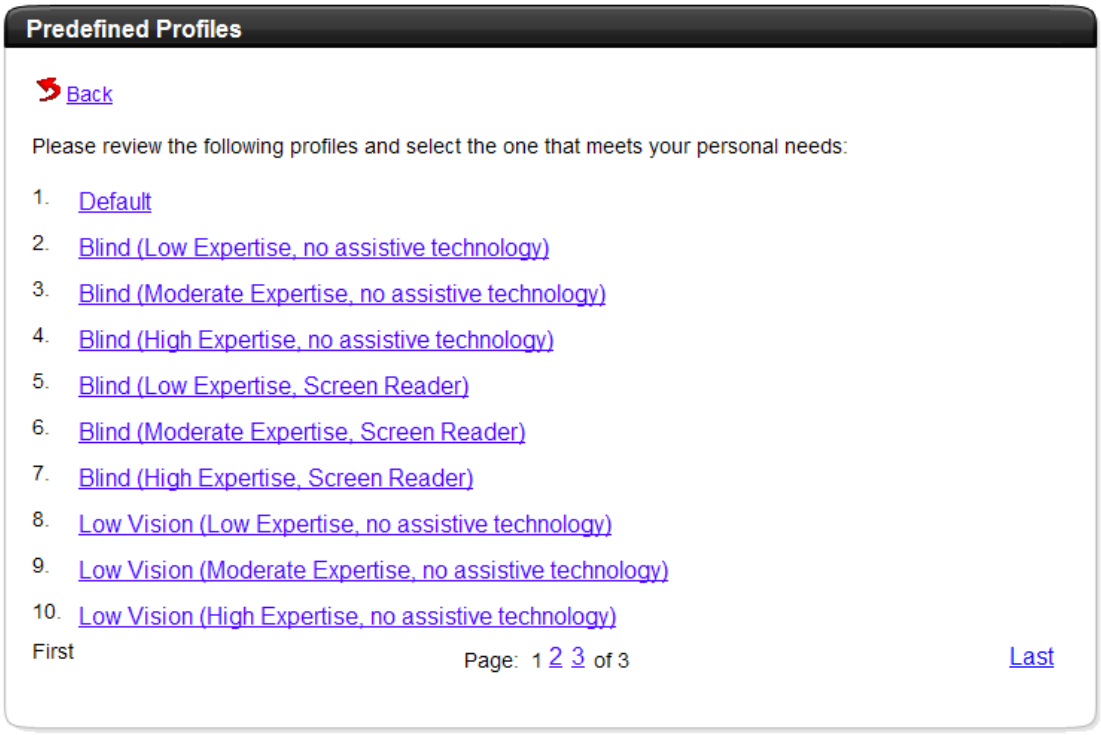


Figure 79: **Selection from a number of predefined profiles**

7.2.1.2 Applying different Accessibility & Interaction settings

In this section an overview of the resulting portal using a number of alternative predefined profiles is presented in order to provide a quick overview of the possible transformation at the not subscribed user's end.

More specifically, a typical set of different settings may include:

- Blind with no Assistive Technology and High Expertise
- Low vision with no Assistive Technology and Moderate Expertise
- Motor Impaired with two Switches and Low Expertise
- Colour Blind (Protanope) with Low Expertise
- Colour Blind (Deuteranope) with Moderate Expertise
- Colour Blind (Tritanope) with High Expertise

Activating the "Blind with no Assistive Technology and High Expertise" profile results in the interface presented in Figure 80. In this figure, the following adaptations are highlighted:

1. Quick access links are presented on the top right and bottom right section of the page
2. Section breaks are displayed on each page region
3. Images are displayed as text
4. Tables are linearized
5. Image Buttons are transformed to links

General adaptations that affect the overall look and feel of the page include the linearization of templates, the absence of graphics, and the colour scheme introduced (white for background black for foreground).

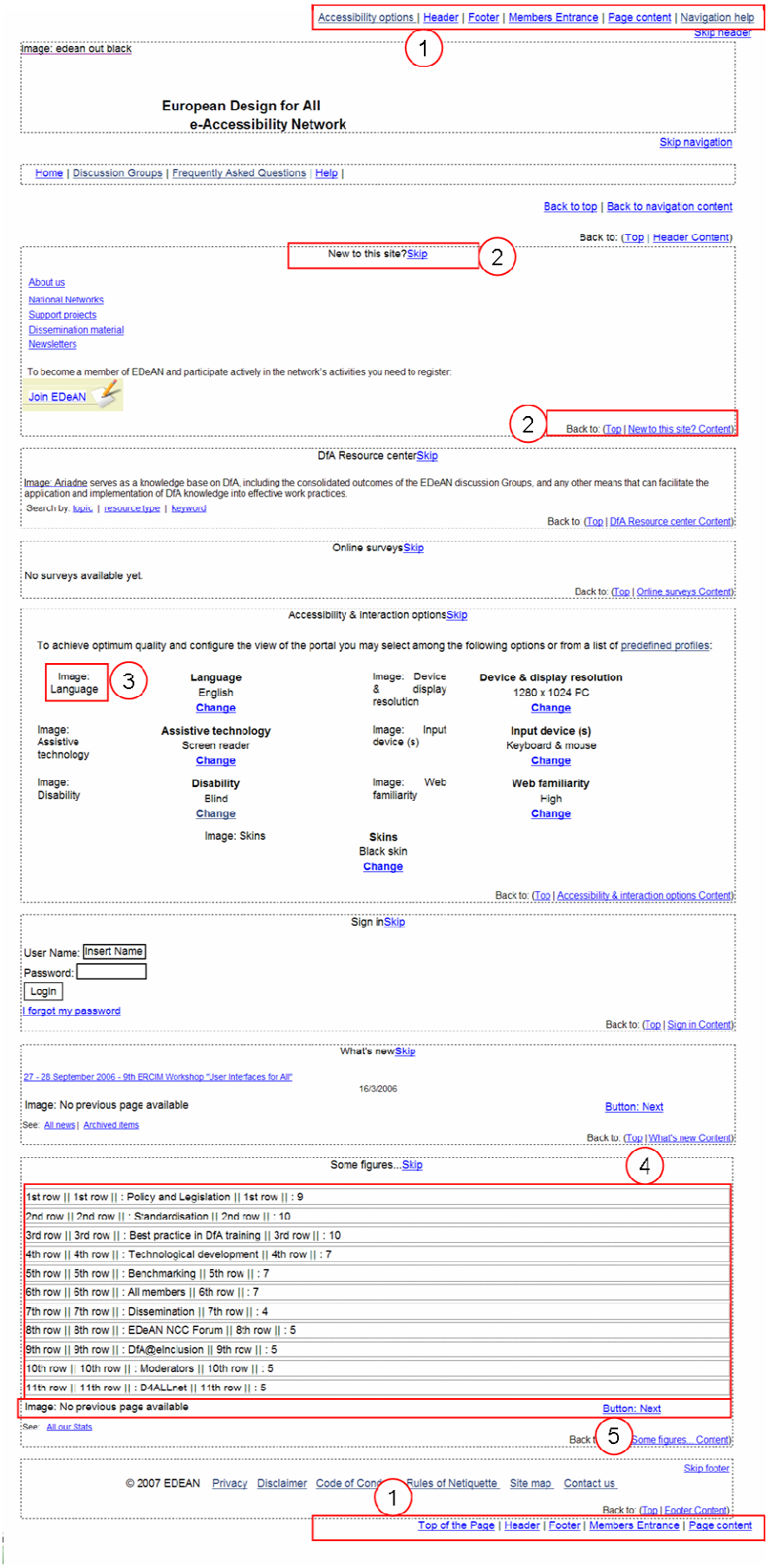


Figure 80: **The "Blind, no Assistive Technology, High Expertise" profile**

Activating the "Low vision with no Assistive Technology and Moderate Expertise" profile results in the interface presented in Figure 81.

As shown in this figure, the adaptations are similar with the ones presented for the previous profile activations, with a number of significant variations. This profile results in a different colour setting with yellow colour for background and black for foreground in order to maximize the contrast. Buttons are presented with larger font size with blue colour for background and white for foreground.

Finally, font sizes have been increased and images are displayed normally only on page content.

Figure 81: **The "Low vision, no Assistive Technology, Moderate Expertise" profile**

Activating the "*Motor Impaired, two Switches, Low Expertise*" profile results in the layout presented Figure 82. In this figure, the following adaptations are highlighted:

1. Various quick access links are presented at the top and bottom of the page
2. Links are displayed as buttons
3. Section breaks are displayed on each page region
4. Text boxes provide feedback on focus
5. A software keyboard is provided for text entry
6. A window with the favourite navigation options is displayed (for novice user)

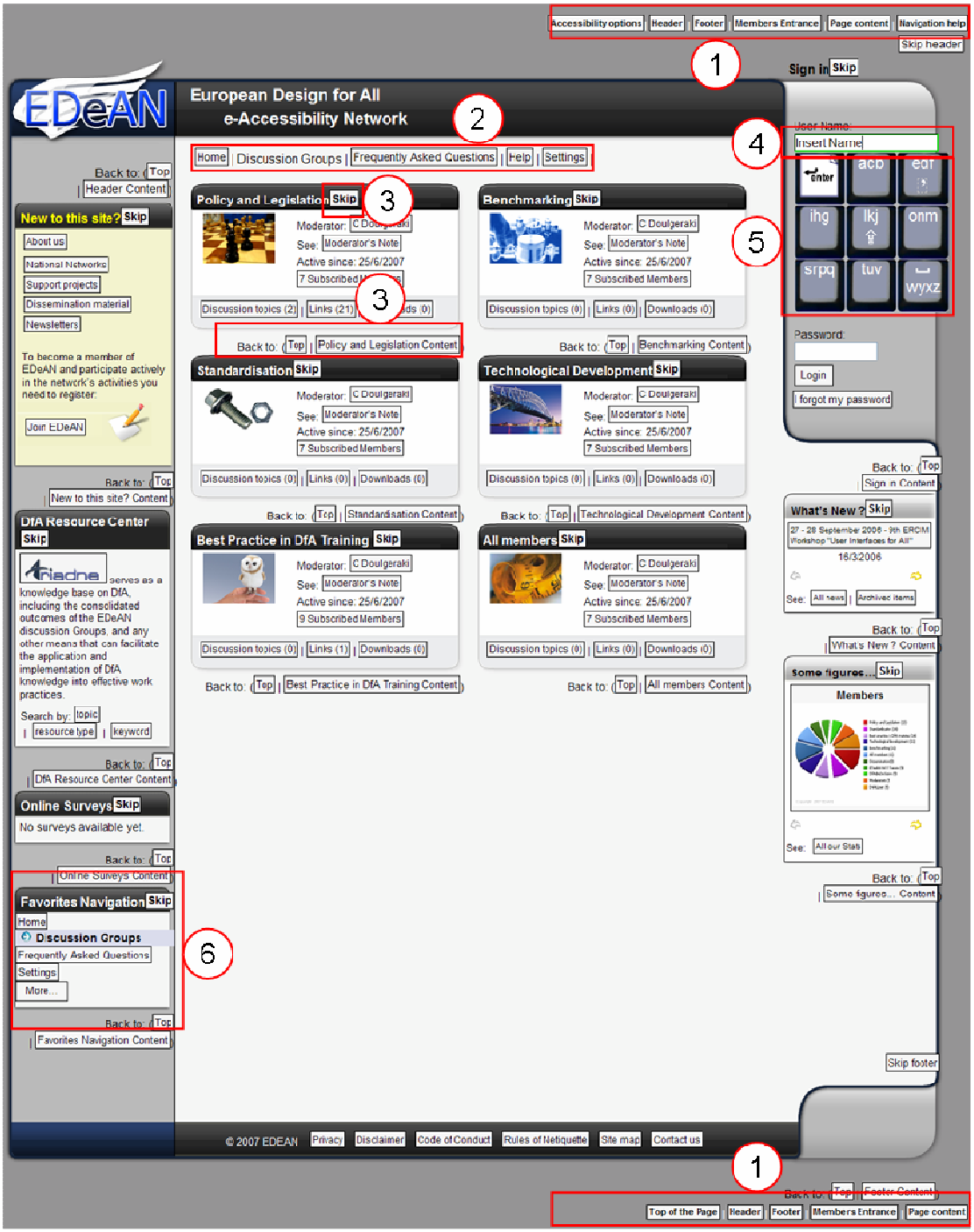


Figure 82: **The "Motor Impaired, two Switches, Low Expertise" profile**

Activating the "Colour Blind (Protanope) with Low Expertise" profile results in the interface presented Figure 83. In the figure above, the following adaptations are highlighted:

1. Links are displayed with pink colour
2. The background colour is set to black
3. Buttons use yellow colour for background, red for border and black for text
4. Charts are rendered using an appropriate colour palette

Figure 83: **The "Colour Blind (Protanope), Low Expertise" profile**

Activating the "Colour Blind (Deuteranope) with Moderate Expertise" profile results in the interface presented Figure 84. In this figure, the following adaptations are highlighted:

1. Links are displayed with blue colour
2. The background colour is set to yellow
3. Buttons use blue colour for background, black for border and white for text
4. Charts are rendered using an appropriate colour palette

Figure 84: **The "Colour Blind (Deuteranope), Moderate Expertise" profile**

Activating the "Colour Blind (Tritanope) with High Expertise" profile results in the interface presented Figure 85. In this figure, the following adaptations are highlighted:

1. Links are displayed with pink colour
2. The background colour is set to blue
3. Buttons use yellow colour for background, red for border and black for text
4. Charts are rendered using an appropriate colour palette

Figure 85: **The "Colour Blind (Tritanope), High Expertise" profile**

7.2.2 Subscribed Area

The portal subscribed area is intended to support the actual networking activities, and therefore provides a number of communication and collaboration facilities. The initial home page provided to portal subscribed users (Figure 86 and Figure 87) is divided in the following sections:

- **The user personal area:** Through this area, the user can access facilities that personally affect the way the portal is adapted to their personal requirements and additional personal facilities such as Registration Data.
- **The SIGs area:** This is the entrance point for the user subscribed Special Interest Groups.
- **The DfA Knowledge tools area:** Through this area users can access a repository of resources on Design for All and additional access training material that can be personalized to form their personal courses.

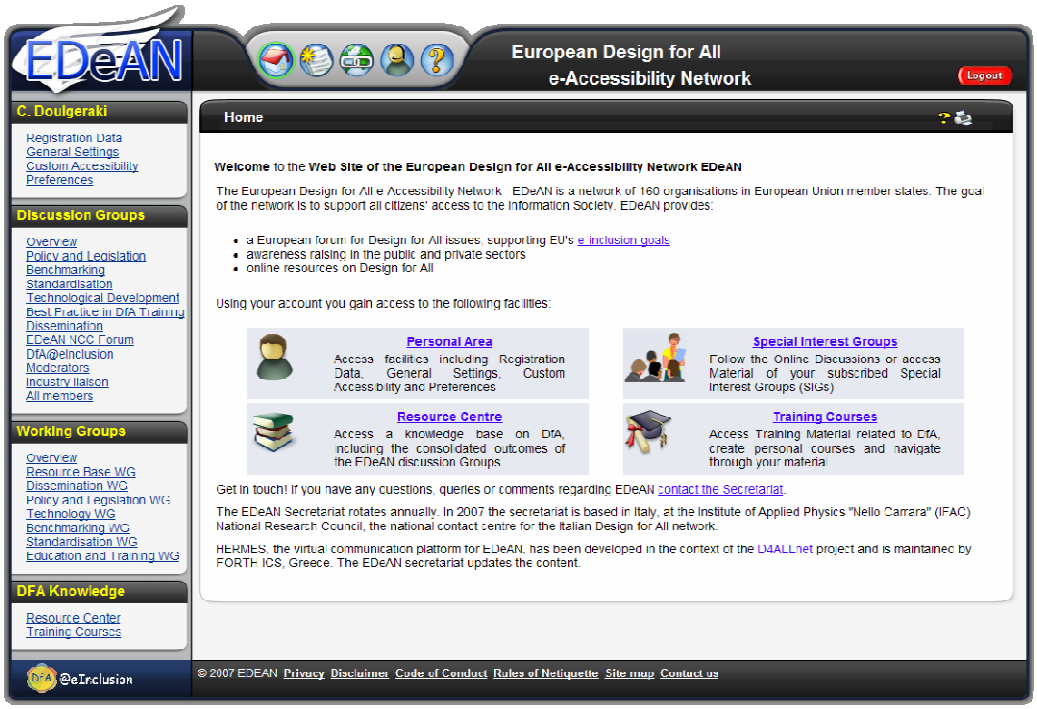


Figure 86: **The Home Page for high display resolution (1024X768 or higher)**

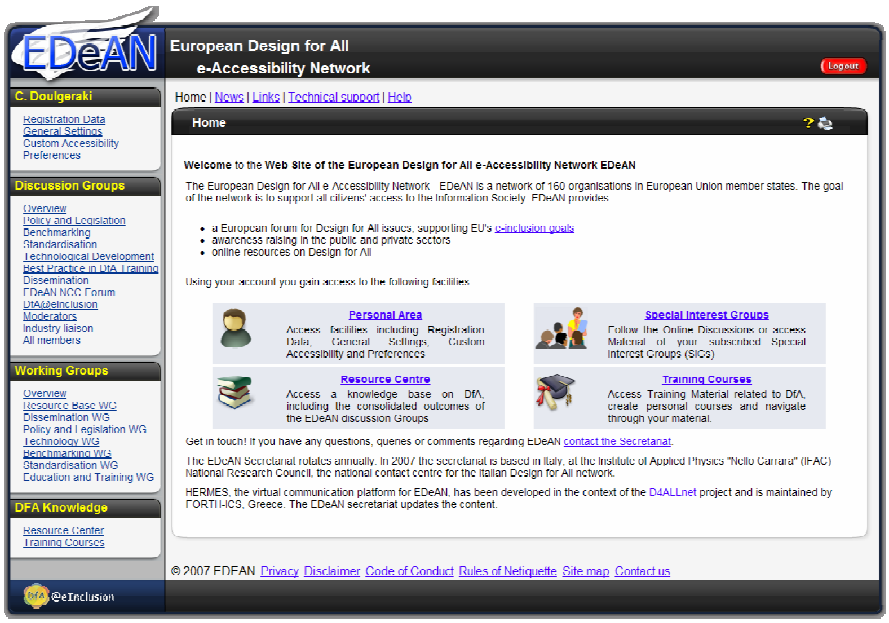


Figure 87: **The Home Page for low display resolution (800X600 or lower)**

For all the sections presented above, the underling functionality and purpose shall be presented in detail in the following sections followed by examples of how each of these facilities can be adapted to specific interaction and accessibility user requirements.

7.2.2.1 User Profiling (Usability – Accessibility settings)

From the Subscribed area users are provided with the ability to alter the portal representation based on a number of different criteria. On a first layer of abstraction the provided User Interface can be altered based on Settings related with the user and context specific parameters. Based on these parameters the default decision making takes place. On a second layer the user can manually override the default adaptation logic by altering specific interaction and accessibility preferences. Towards this direction facilities were developed for enabling users to administer the way that adaptations occur. These facilities are in depth analysed in the following sections.

Settings

Settings represent the user and context specific attributes for performing the default adaptation logic. These attributes can be thought as the minimum level of information required. More specifically the basic user attributes are:

- Language
- Disability
- Web familiarity

On the other hand the basic context specific attributes are:

- Device and display
- Input device
- Assistive technology

For facilitating the manual specifications of these settings by portal users a specific administration interface was developed as presented in Figure 88. In this interface each of the available settings is graphically presented together with the currently selected user setting.

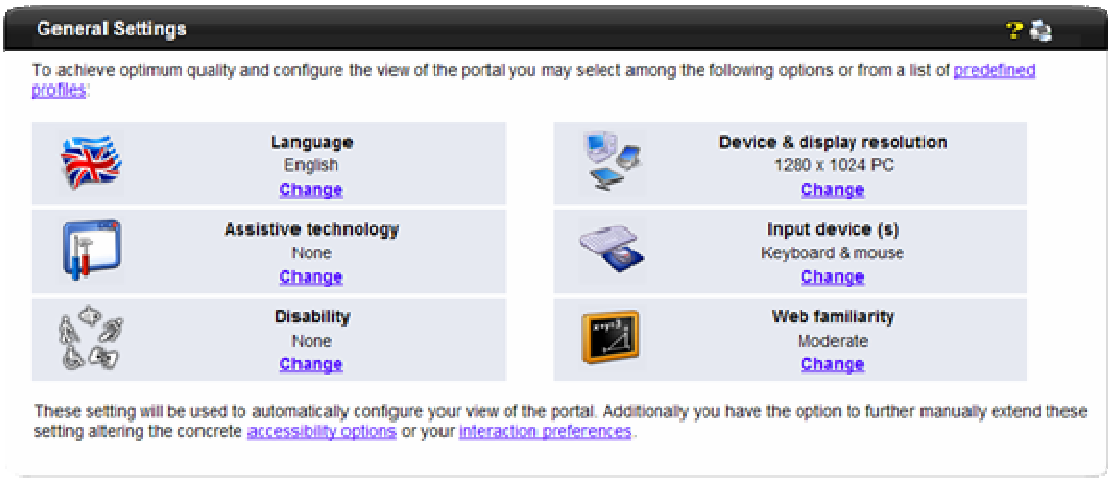


Figure 88: **The basic settings interface**

Language

Selecting the preferred language affects both the interface and content representation. Through this setting the interface and module content is presented using the selected language. The selection of the preferred language is accomplished using the interface presented in Figure 89.

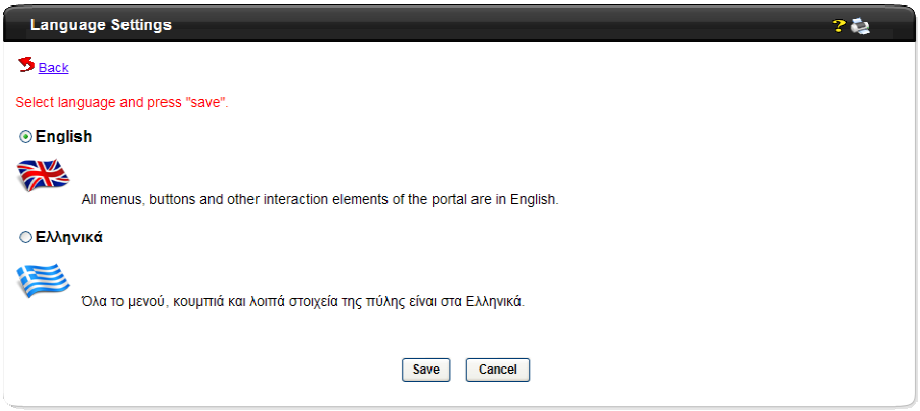


Figure 89: **The language settings interface**

Disability

Defining user disability affects a wide number of adaptations. For example a typical adaptation scenario in the case of a blind user would be to linearize the portal template and tables, to stop displaying images on page template and content, to alter font size and colour etc. Therefore the selection of disability can radically increase the accessibility and usability of the interface presented to each user. Towards this direction Figure 90 provides an overview of the disability selection interface.

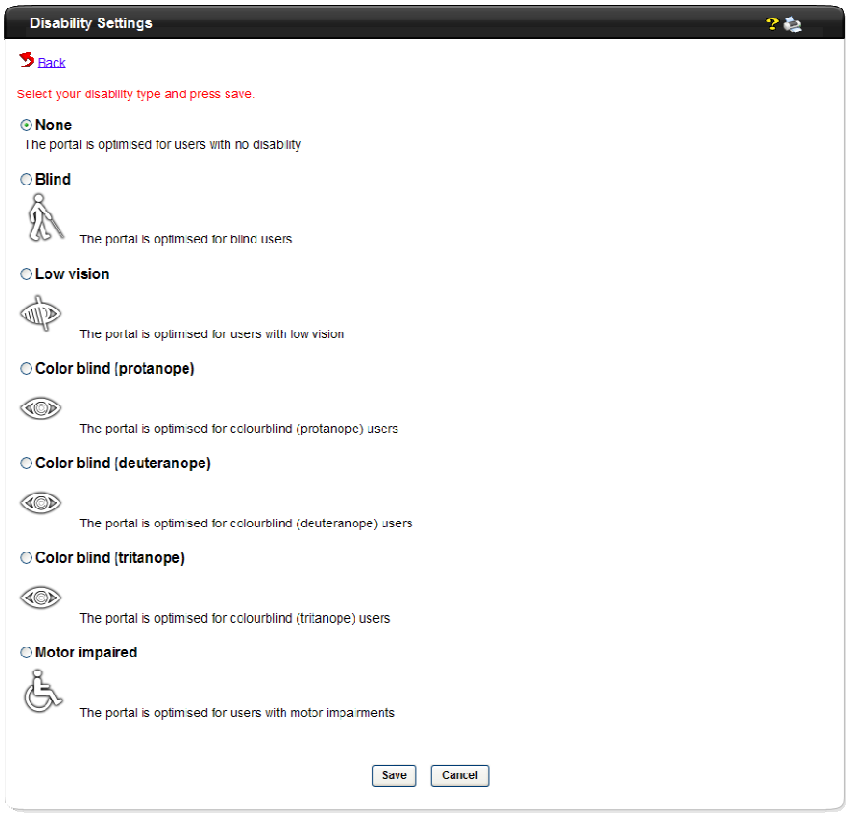


Figure 90: **The disability settings interface**

Web familiarity

The Web familiarity setting was incorporated for allowing each user to define his expertise. This setting is used for performing a number of adaptations that affect the usability of the resulting interface. For example for a user with low web familiarity interaction elements that have more explanatory information are presented. Additionally in the case of high Web familiarity interactions elements that enable faster completion of tasks are provided. Through the portal users can specify their expertise using the interface presented in Figure 91.

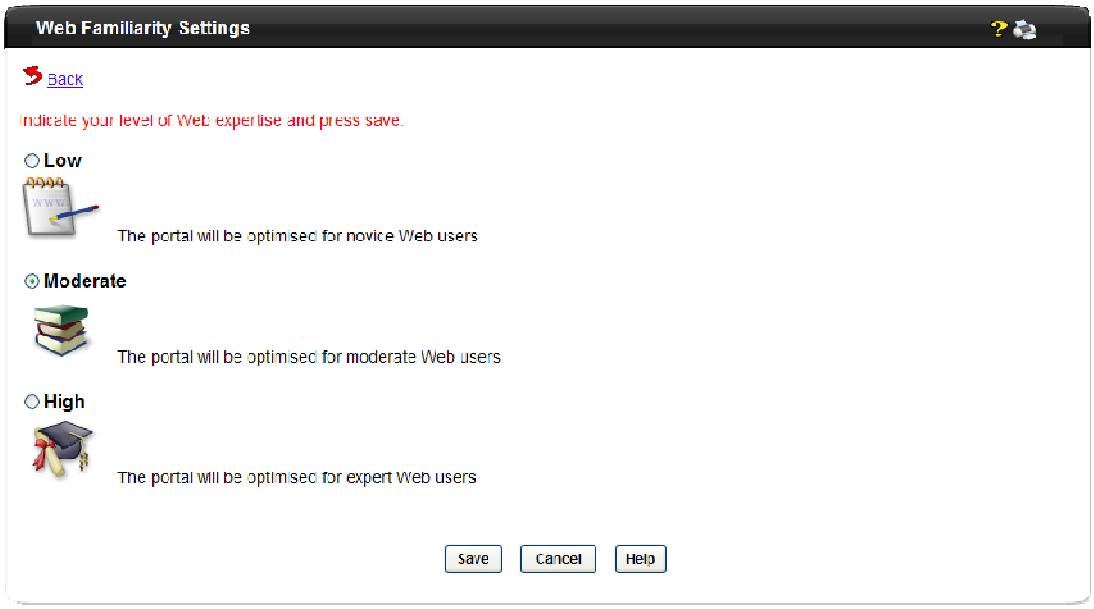


Figure 91: **The Web Familiarity settings interface**

Device and Display

The Device and Display selection can affect the way that the portal is presented based on device used for accessing the portal and the presentation characteristics of the device. This setting reflects the increasing trend of user mobility and the resulting need for providing seamless access to web environments thought a number of different devices such as PDAs tablet PCs etc. The interface enabling the selection of the currently used device is presented in Figure 92.

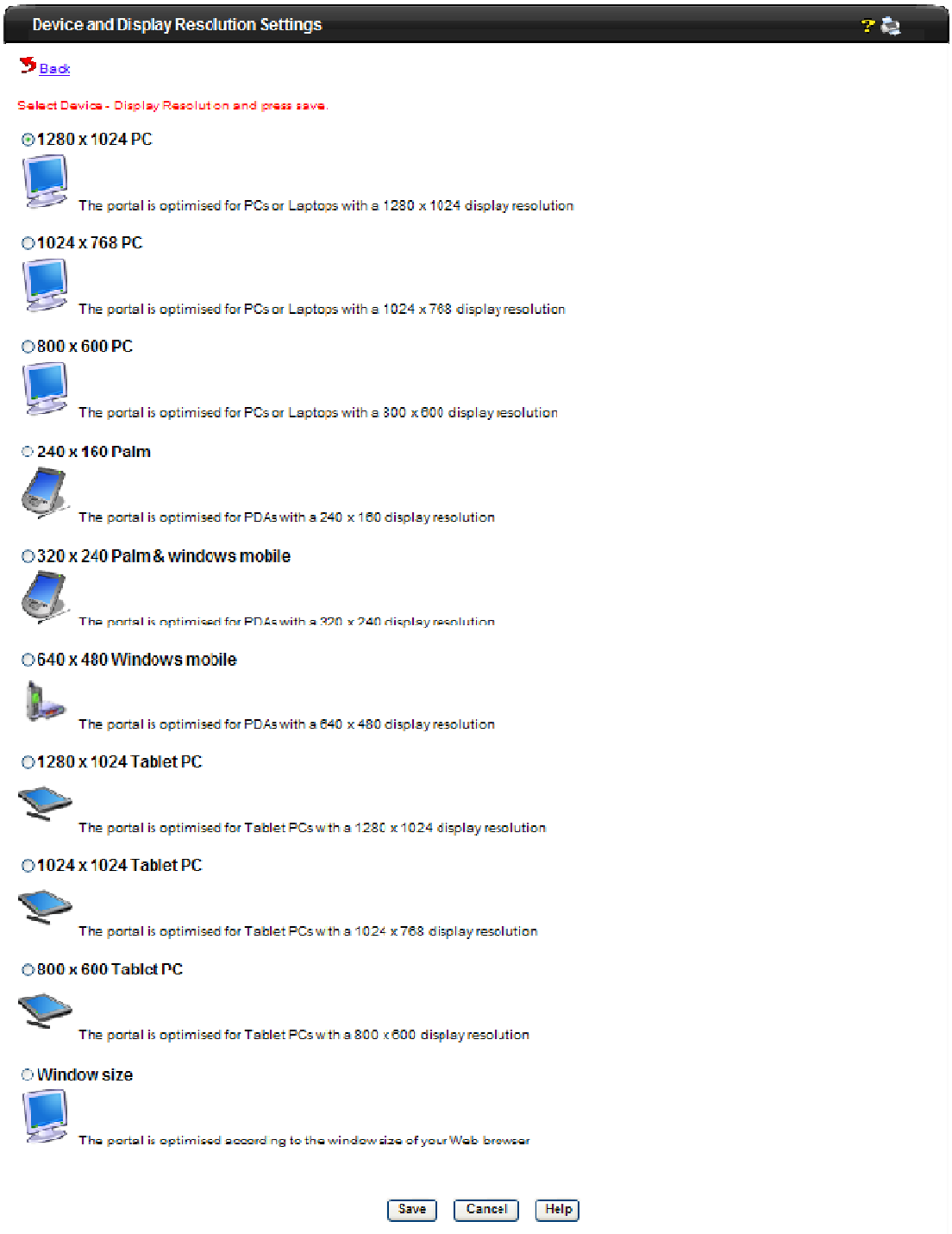


Figure 92: **The Device and Display resolution settings interface**

Input device

The input device selection is basically used for adapting the portal on the device used for browsing and inserting content. For example if only mouse is used for both functions then mouse operated virtual keyboard must be rendered when text input is required. On the same manner when only keyboard is used the portal must be adapted for keyboard only navigation. The interface provided for facilitating the selection of the currently used input scheme is presented in Figure 93.

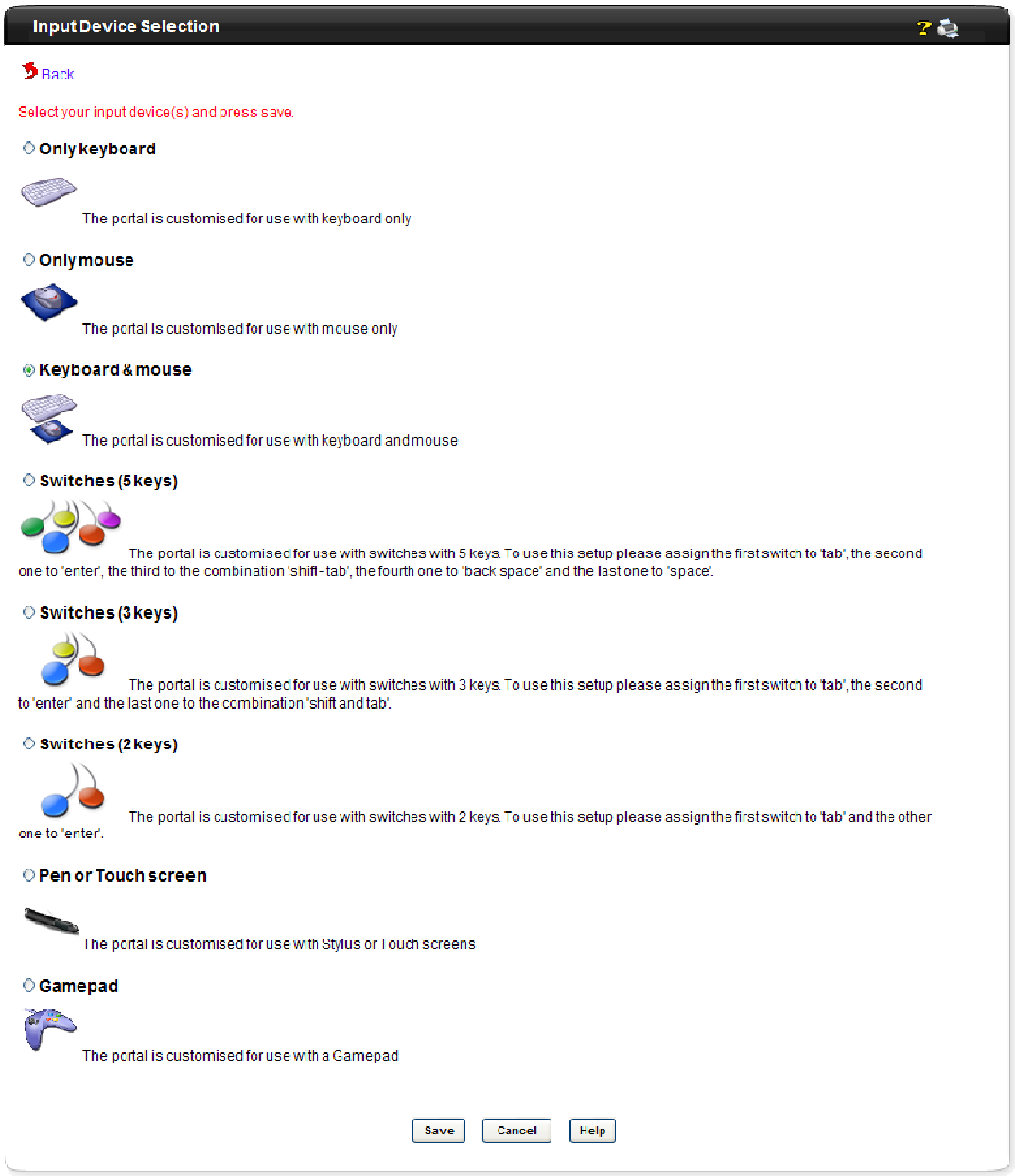


Figure 93: **The Input device selection interface**

Assistive technology

The process of selecting the assistive technology used by a user for accessing the portal is important for performing a number of adaptations. For example in the case of a motor impaired user this setting can be used to clarify whether an external software keyboard is used or the system should provide an alternative text entry mechanism. In the same manner in the case of a blind user that does not use a screen reader text to speech output must be provided. Towards this direction the interface presented in Figure 94 is used for facilitating the selection of the assistive technology used.

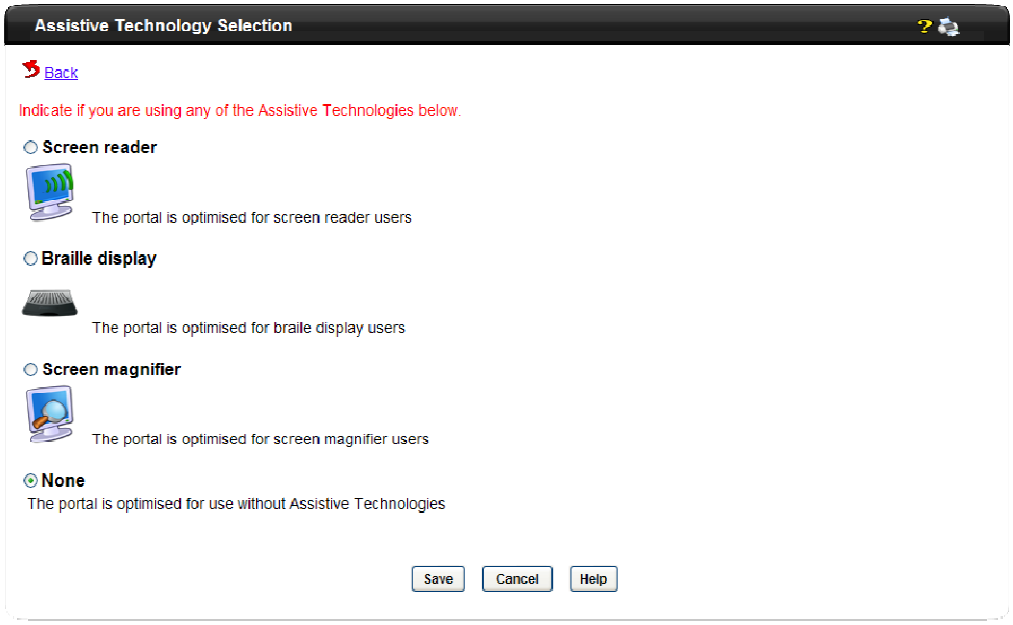


Figure 94: **The Assistive Technology selection interface**

Preferences

Preferences represent settings that affect the way that user interacts with the portal. More specifically these settings can alter the interaction elements used for performing fundamental operations such as browsing content and images or uploading files. The changes made to these settings are propagated to all portal modules. By manually altering these setting the default adaptation logic that occurs based on the user basic setting is enriched. The administration interface provided to portal user for altering these settings is presented in Figure 95.

Figure 95: **The interaction preferences administration interface**

File uploading

This setting affects the way that files are uploaded. More specifically several different styles are available such as the direct or indirect manipulation dialogues. The selection of the preferred file uploading style can completed through the interface presented in Figure 96.

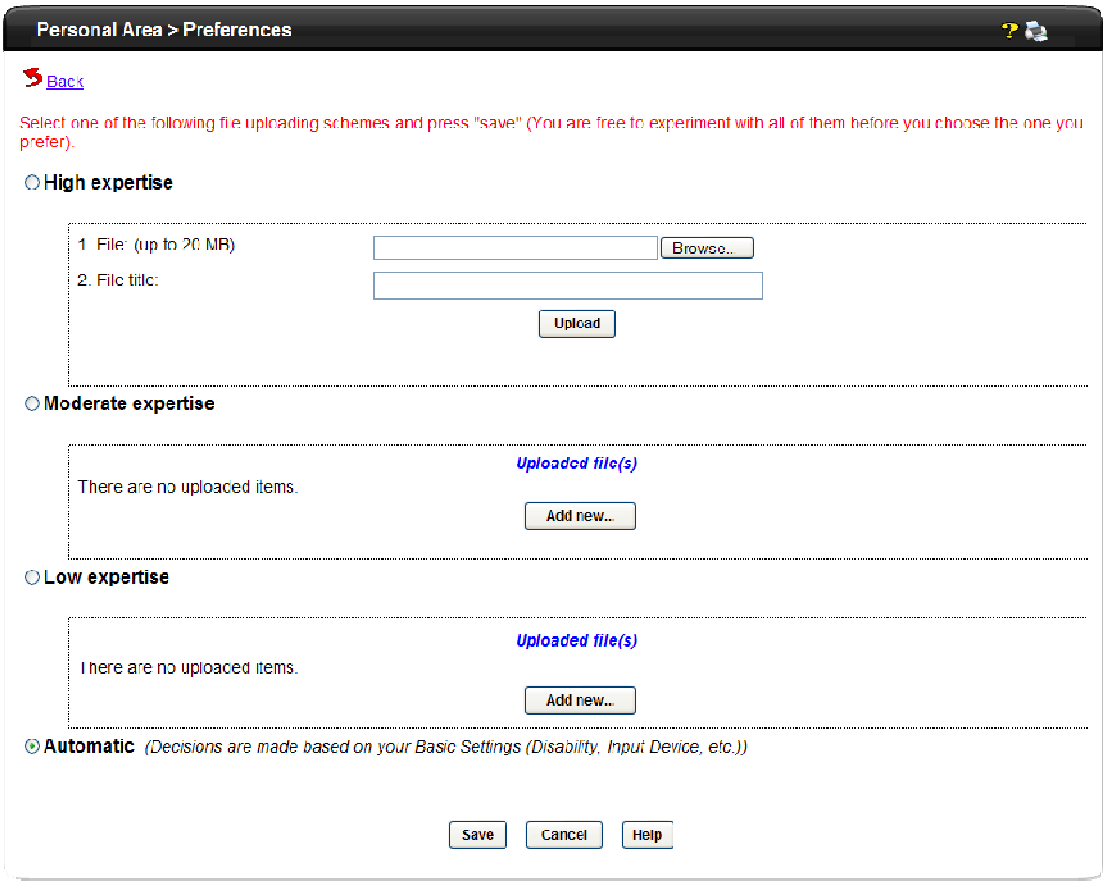


Figure 96: **The File uploading style selection interface**

File displaying

The File displaying setting affects the way that a collection of files is displayed. More specifically several different styles are available differing in the amount of information presented for each file. The selection of the preferred file displaying style can be done using the interface presented in Figure 97.

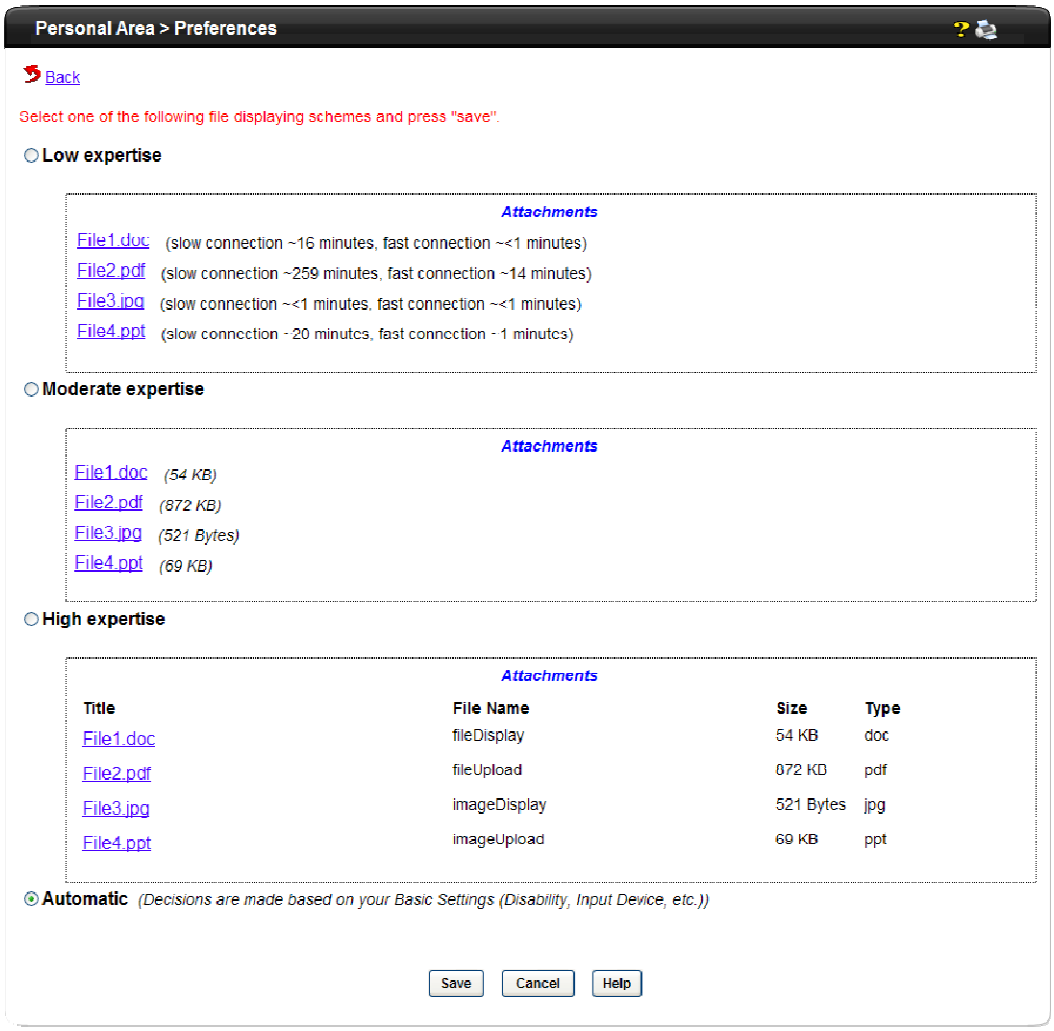


Figure 97: **The File displaying style selection interface**

Paging

This setting affects the way that navigation through a large number of paged data is accomplished. More specifically several different styles of navigation are available such as the page number selection or the next previous page scheme. Different styles differ not only on the available functionality but also on their accessibility characteristics. The selection of the preferred paging style can be done using the interface presented in Figure 98.

Figure 98: **The Paging style selection interface**

Date entry

Date entry affects the interactive elements presented to end users for date input. Towards this directions several interactions styles are incorporated such as the selection through a graphical calendar, the selection from drop down menus etc. The interface presented for choosing the preferred date selection style is presented in Figure 99.

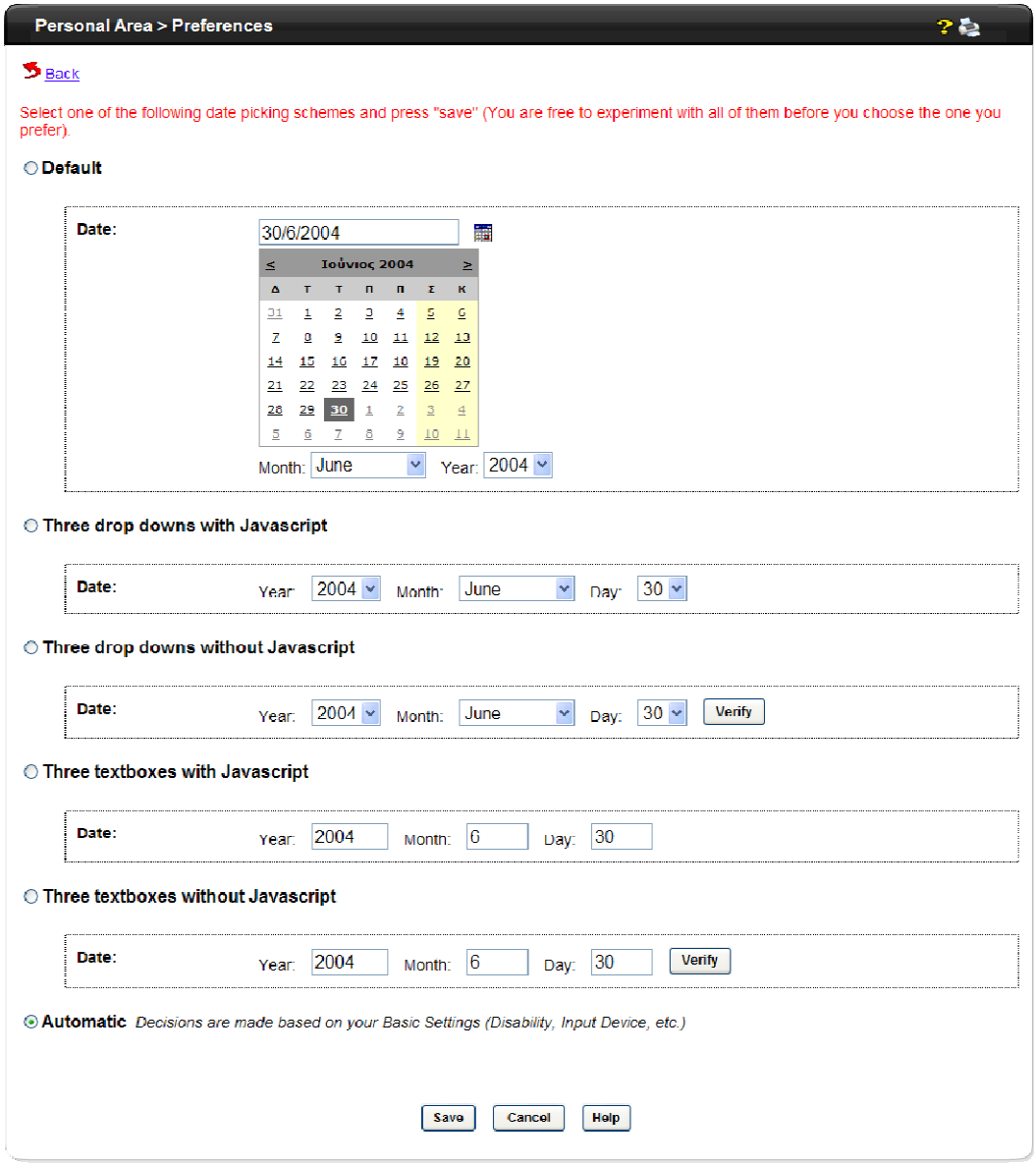


Figure 99: **The Date selection style interface**

Image uploading

Image uploading affects the task structure provided to end user for uploading images. Different uploading schemes can vary on the functionality concurrently appearing on the screen or the interaction steps required for completing a task. The process of selecting the desired task structure is carried out through the interface presented in Figure 100. Users can test the alternative styles and in the end select the one that best suits their personal requirements.

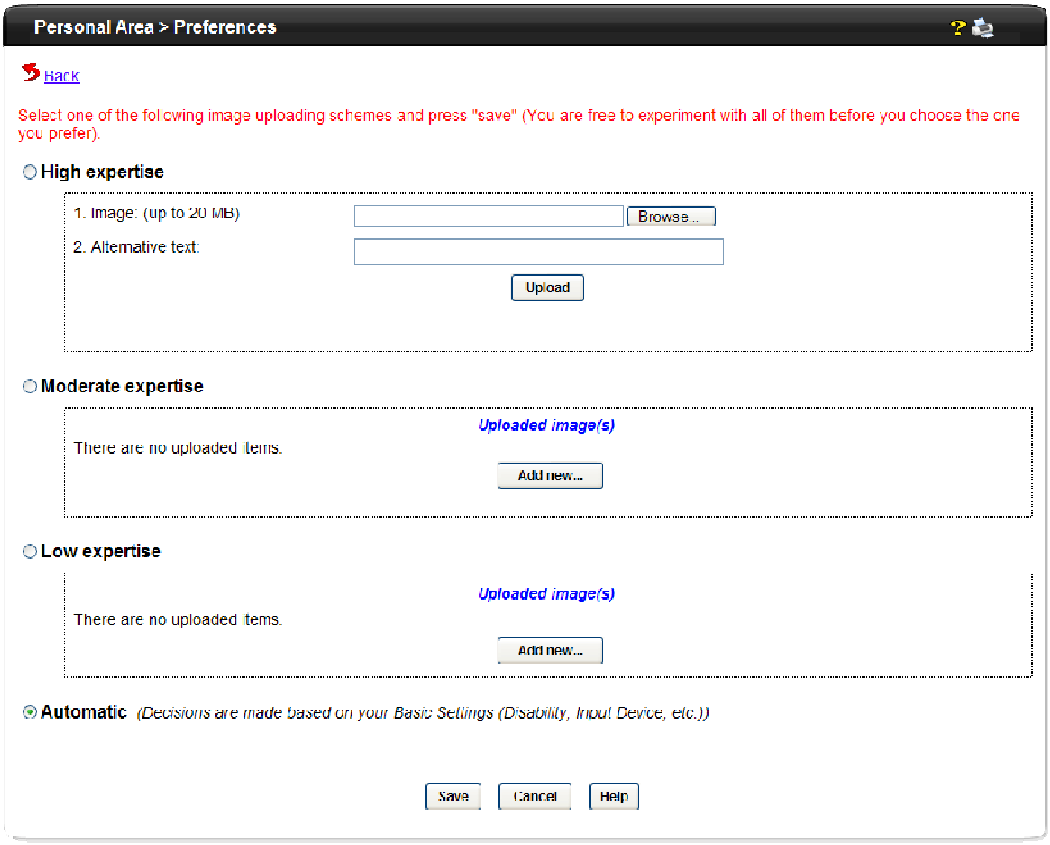


Figure 100: **The Image uploading style selection interface**

Image displaying

Displaying a collection of images is a function that can have a great number of variations and no prominent selection. Towards this direction images can be displayed on a list with variations on the amount of information present or in a slide show version with navigation. The interface where users can view the available variations and select the one that meets their requirements is presented in Figure 101.

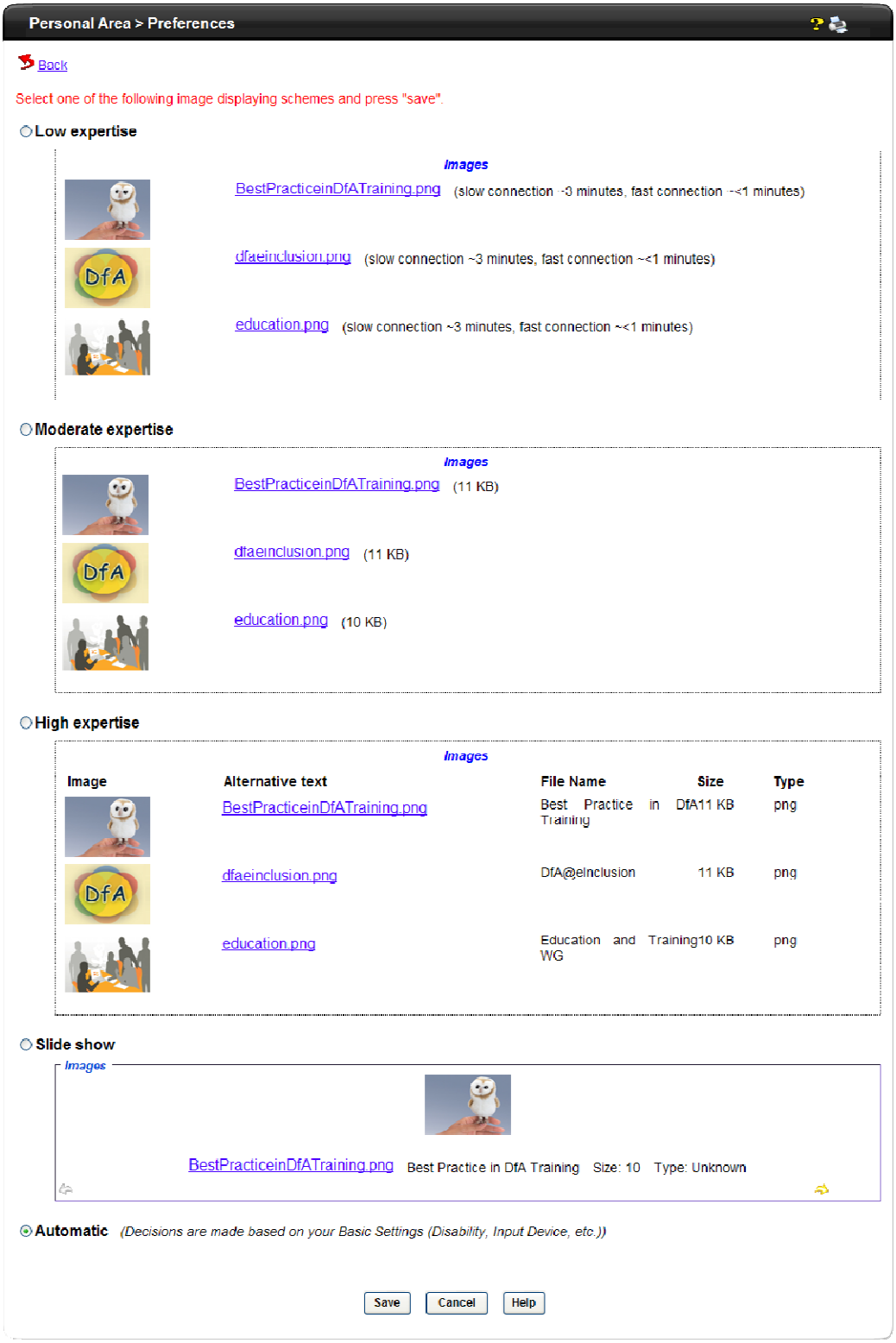


Figure 101: **The Image displaying style selection interface**

Module functions

The functionality offered by a module usually appears as a collection of buttons that perform different operations. These functions can vary in terms of the provided user guidance. A collection of functions can be accompanied by help for easy identification of the function scope or with ability to easily access help regarding functions when needed. These schemes are supported by the Module functions interface (Figure 102) where a user can visualise the different function representations and in turn select the one that best meets his preferences.

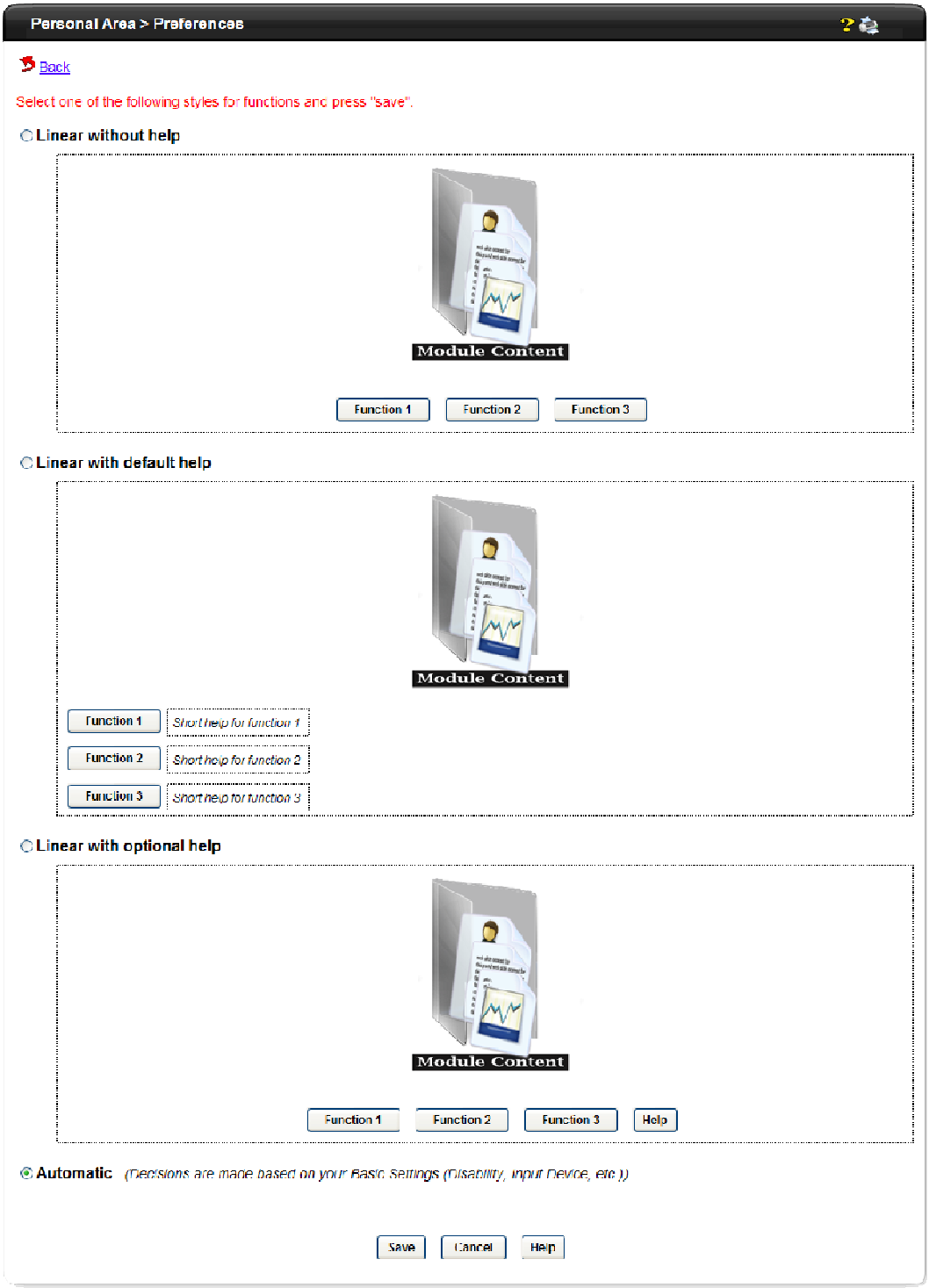


Figure 102: **The Module function style selection interface**

Module options

Module options usually represent the internal navigation structure of a module leading to the different module facilities. Towards these directions different module representations can vary according to the positioning scheme of the internal navigation scheme or the preferred navigation metaphor (widget with options, tabs, links etc). The interface where the different module option representations are visualised for selecting the one best suited for a user in presented in Figure 103.

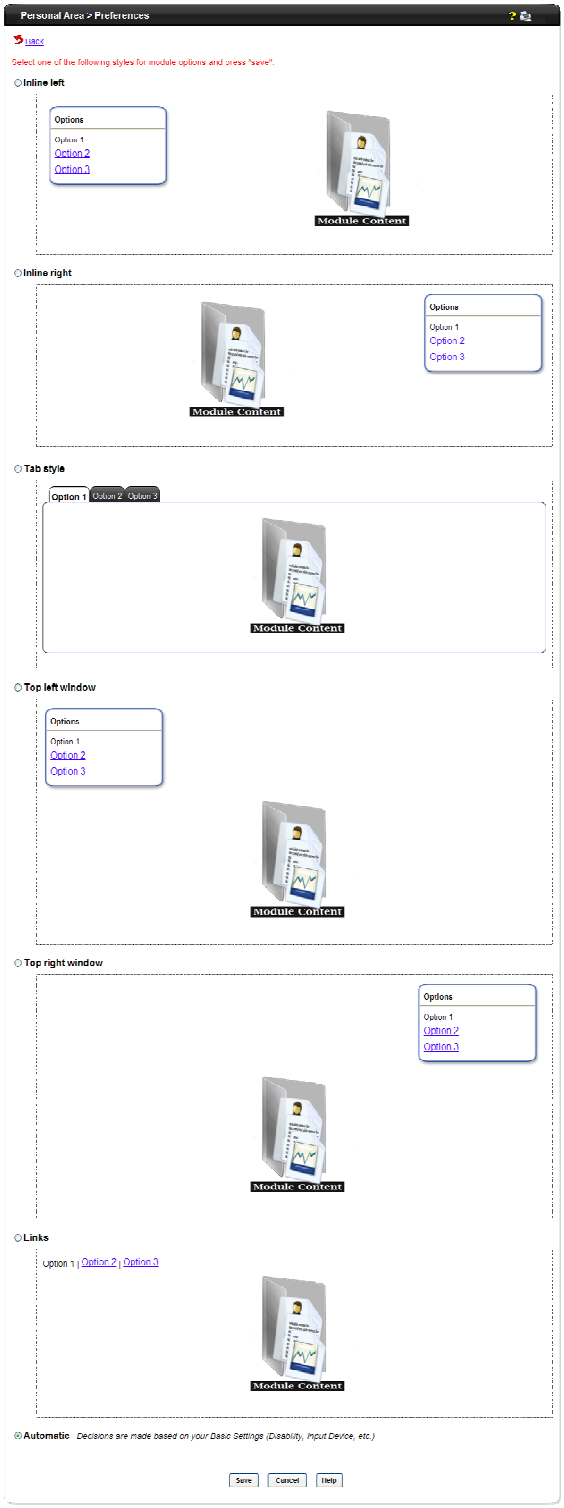


Figure 103: **The Module options style selection interface**

Tabs

Tab based navigation is usually incorporated when presenting different views of related information and therefore has become very popular in interactive applications. The presentation scheme used for presenting tabs may vary from simple links to graphical representations. Through the Hermes portal user can access and select their preferred tab presentation using the interface presented in Figure 104.

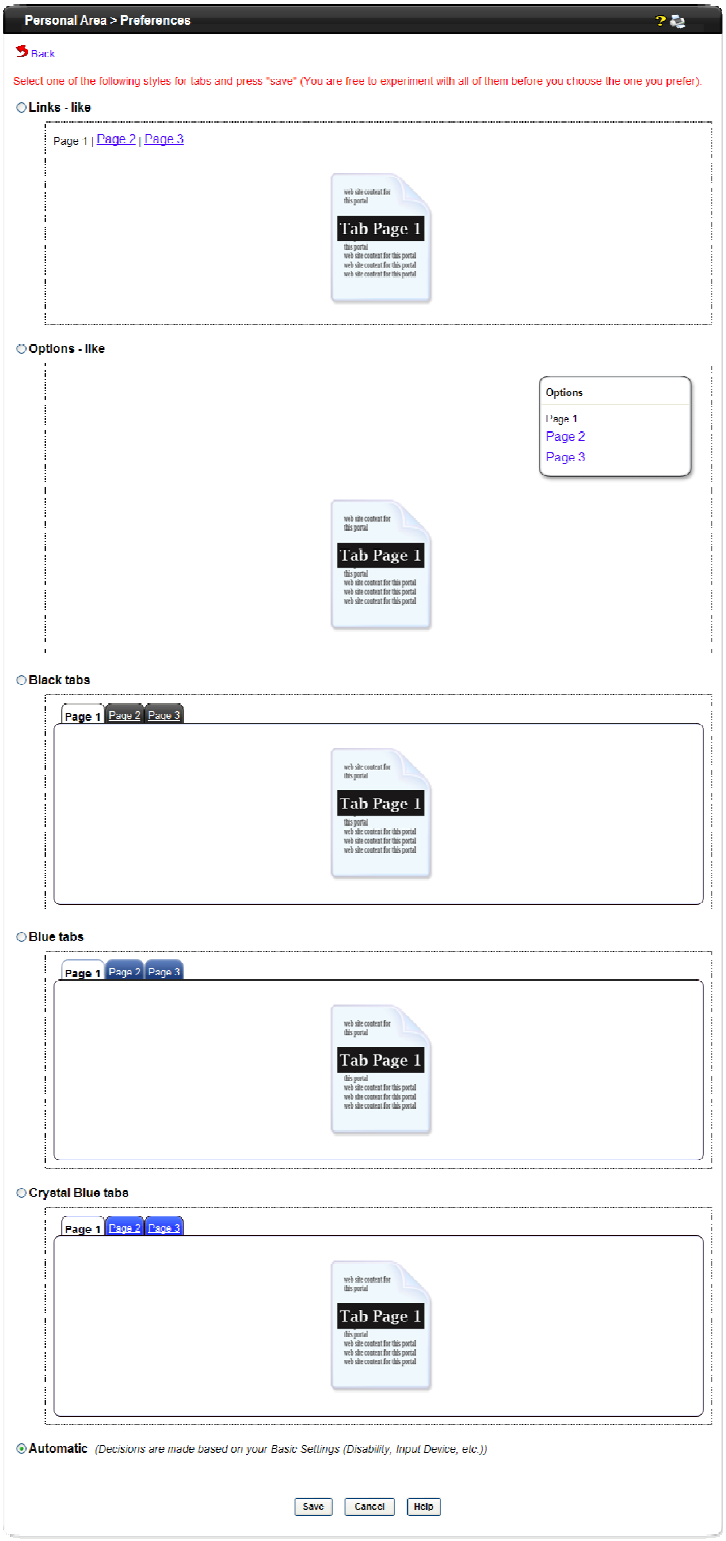


Figure 104: **The Tabs style selection interface**

Text editing

Editing text is a common function in most content management systems. The facilities offered for performing such operations vary from the incorporation of graphical editors to simple text boxes. Additional these variations can be further altered based on the expertise of the user accessing a graphical editor or his disability. In this context the Hermes portal provides an interface (Figure 105) for viewing the alternative text editing schemes and selecting the preferred one.

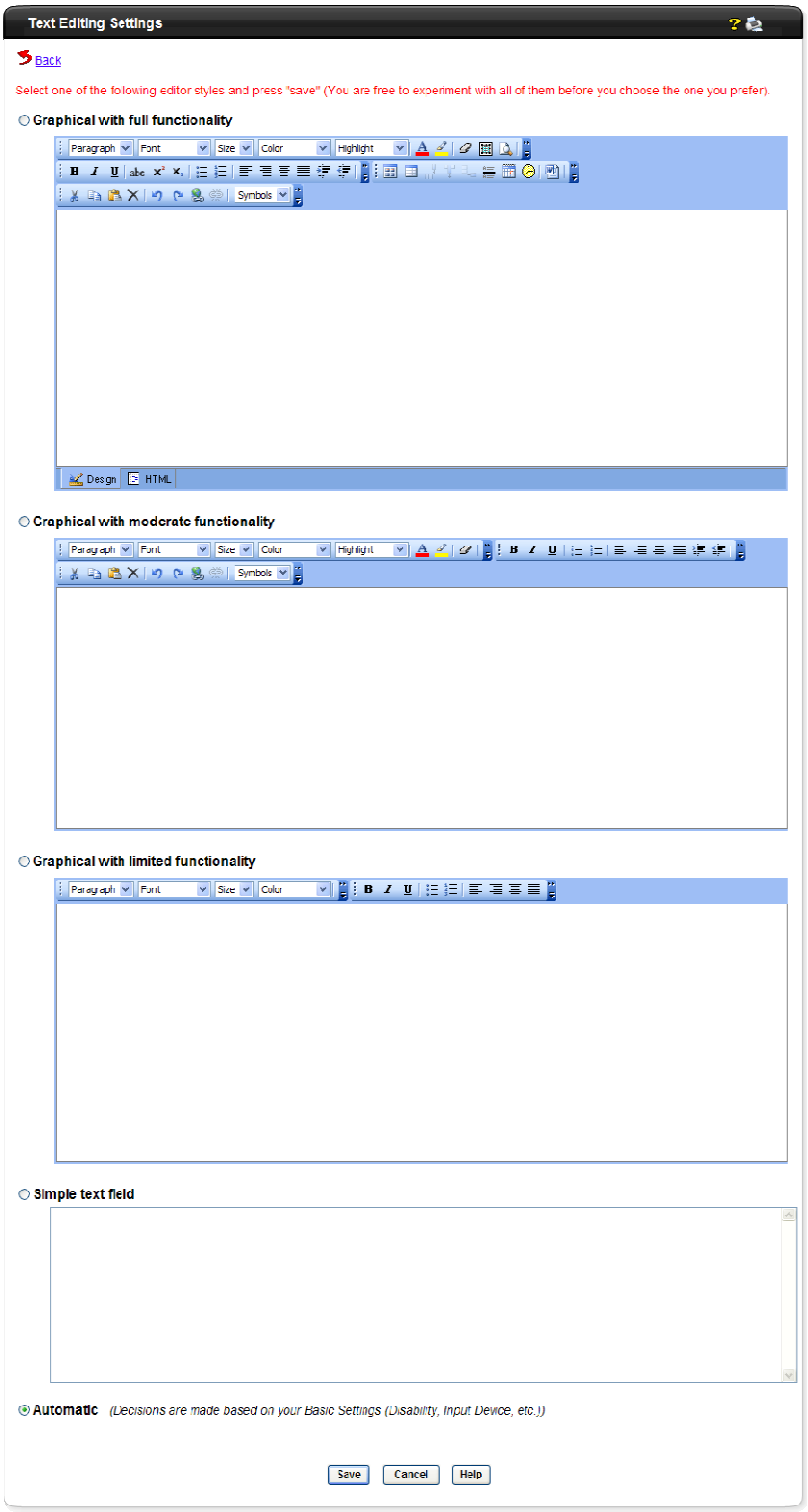


Figure 105: **The Text editing selection interface**

Search variations

This setting represents the variations that can be perceived based on the standard ways that search facilities are represented in web applications. More specifically search facilities usually come with a simple version as the initial interface providing the additional option to move to a more complex one. This on the one hand can frustrate inexperienced users and on the other can make experienced users that require full functionality to perform another unnecessary step for accessing the advanced search facilities. Towards this direction different search variations may include a simple only version, an advanced only version and the standard simple to advanced representation. The interface where these variations are visualised together with the ability to select the preferred style is presented in Figure 106.

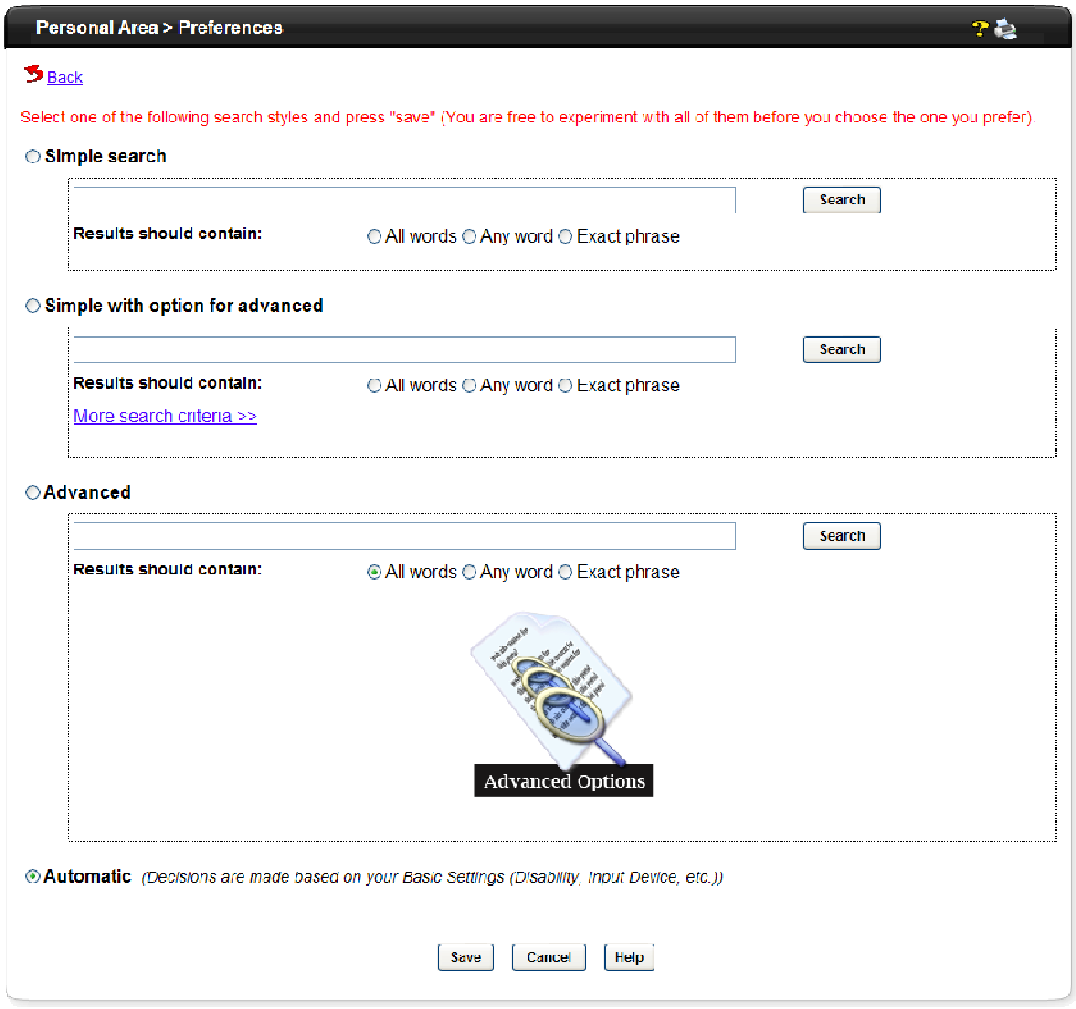


Figure 106: **The Search variations interface**

Favourite navigation options

Although interactive applications usually offer a wide collection of facilities users most of the times use a small collection of the available functionality. Towards this direction the provision of a way to quickly access the favourite navigation options can enhance the usability of an interactive system by narrowing down the potential selection space. Regarding the representation of these favourite options several schemes can be followed such as the incorporation of a new interface collecting all favourite options, the provision of special marking for these options etc. Through the Hermes portal users can select their personal preference using the interface presented in Figure 107.

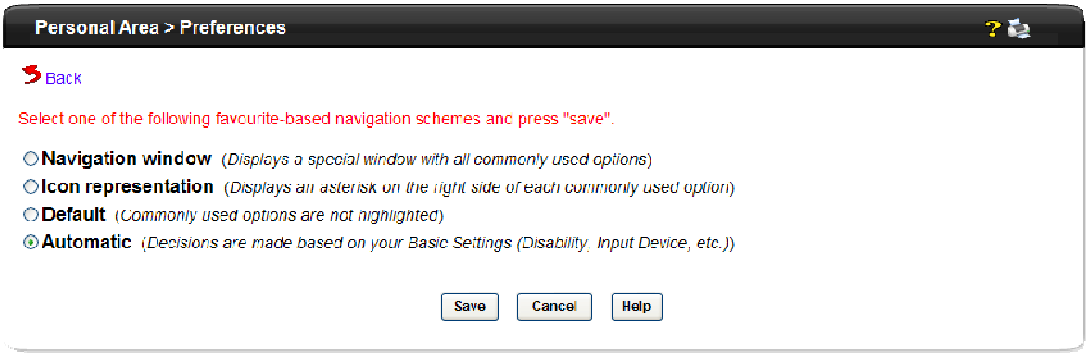


Figure 107: **The Favourite navigation options selection interface**

Navigation

Navigation affects the way that the available navigation options appear to end users. The potential variations in their presentation include the sorting of navigation options based on their popularity with the option to hide or not unused options. The interface provided for selecting the desired scheme is presented in Figure 108.

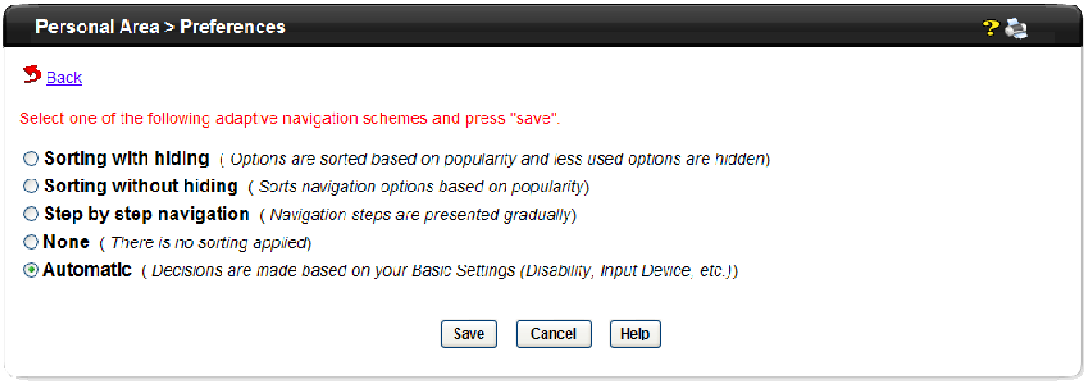


Figure 108: **The Navigation style selection interface**

Skin selection

Skinning has become one of the major facilities offered for personalizing an interactive application to the presentation preferences of a target user. The new Hermes platform is developed with an internal fully customizable skinning mechanism available to both subscribed and visitors of the portal. The interface provided to portal subscribed users for previewing the available skins and selecting the one that meets their personal preferences is presented in Figure 109.

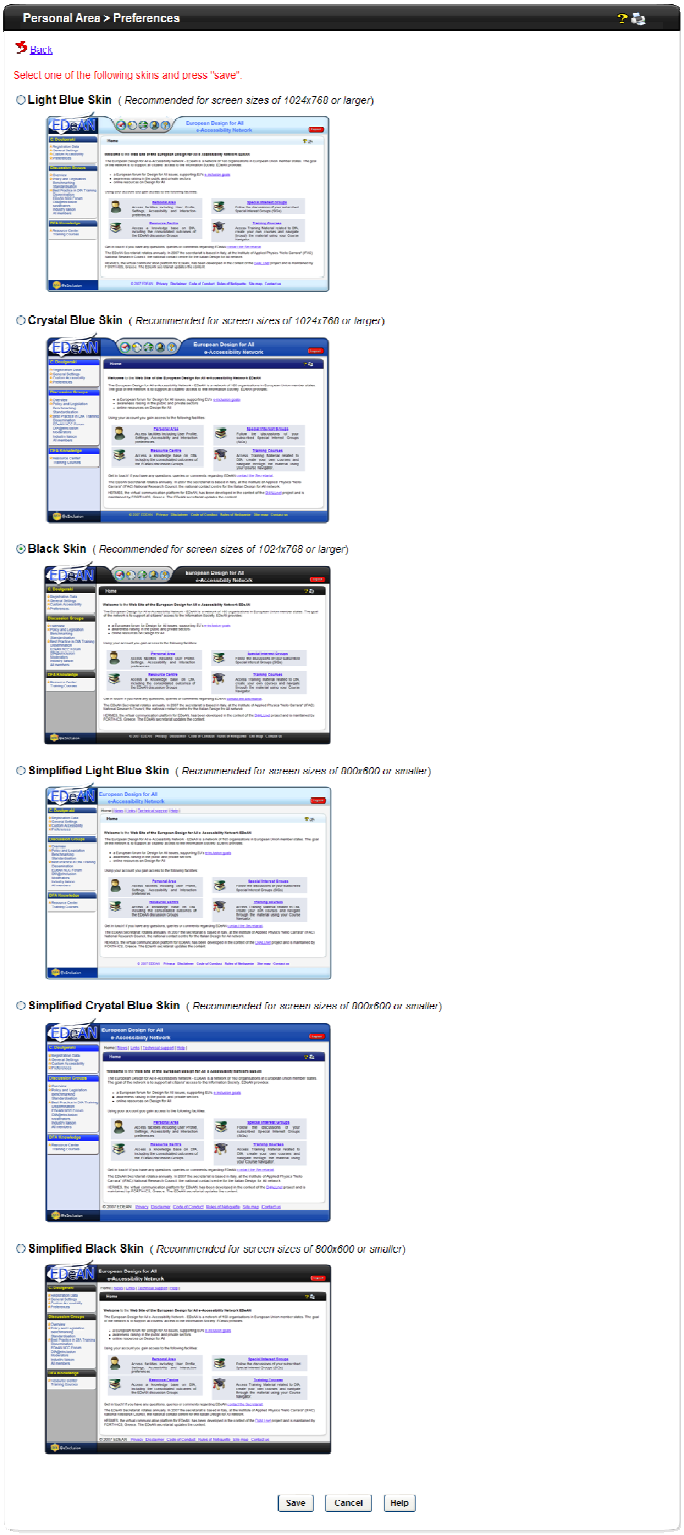


Figure 109: **The Skin selection interface**

Font Family

The ability to alter the font family used for presenting portal content can be very helpful for providing a personalised experience with respect to accessibility issues that may require the use of specific font families. Towards this direction the new Hermes platform allows the selection of font family through the interface presented in Figure 110.

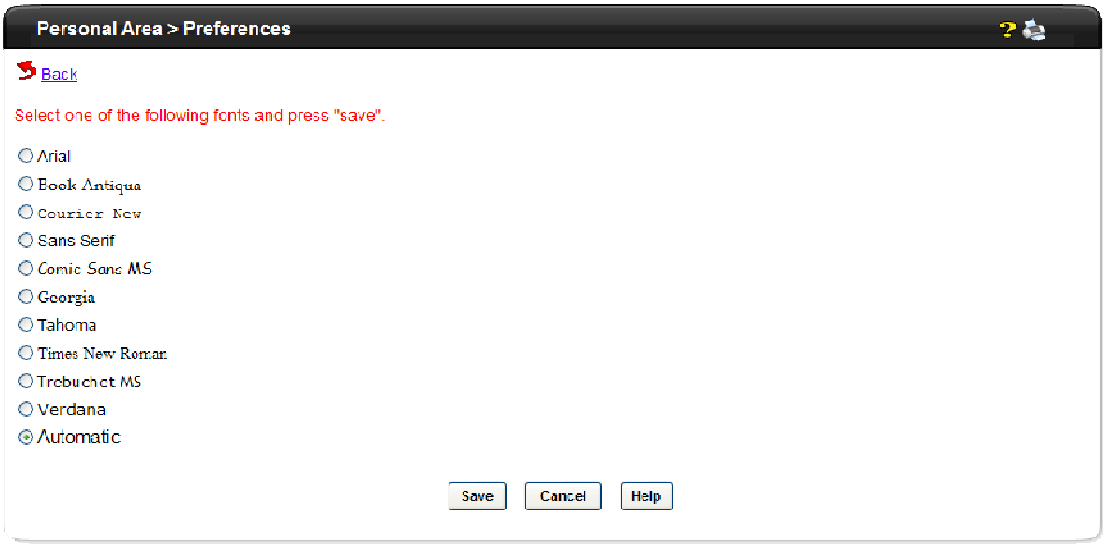


Figure 110: **The Font Family selection interface**

Custom Accessibility

Custom Accessibility includes all the settings that can be altered to enhance the accessibility characteristics of the final user interface. This is very important for offering personalised experience from an accessibility perspective. So although each user interface is already compliant with the W3C accessibility guidelines theses settings can further enhance the actual system accessibility and the perceived quality of interaction. The main administration interface provided to end users for editing these setting is presented in Figure 111. As presented in this figure each setting is presented graphically together with the currently selected value. Additionally the option to alter the selected values is provided.

Figure 111: **The Accessibility preferences administration interface**

Template linearization

This setting defines whether the template must be liberalised in order to be more easily manipulated by screen readers. In general the spatial arrangement of interface elements serves the needs of sighted users and does not offer any usability gain for users with partial or total blindness. Towards this direction the preferred template linearization scheme can be selected through the interface presented in Figure 112.

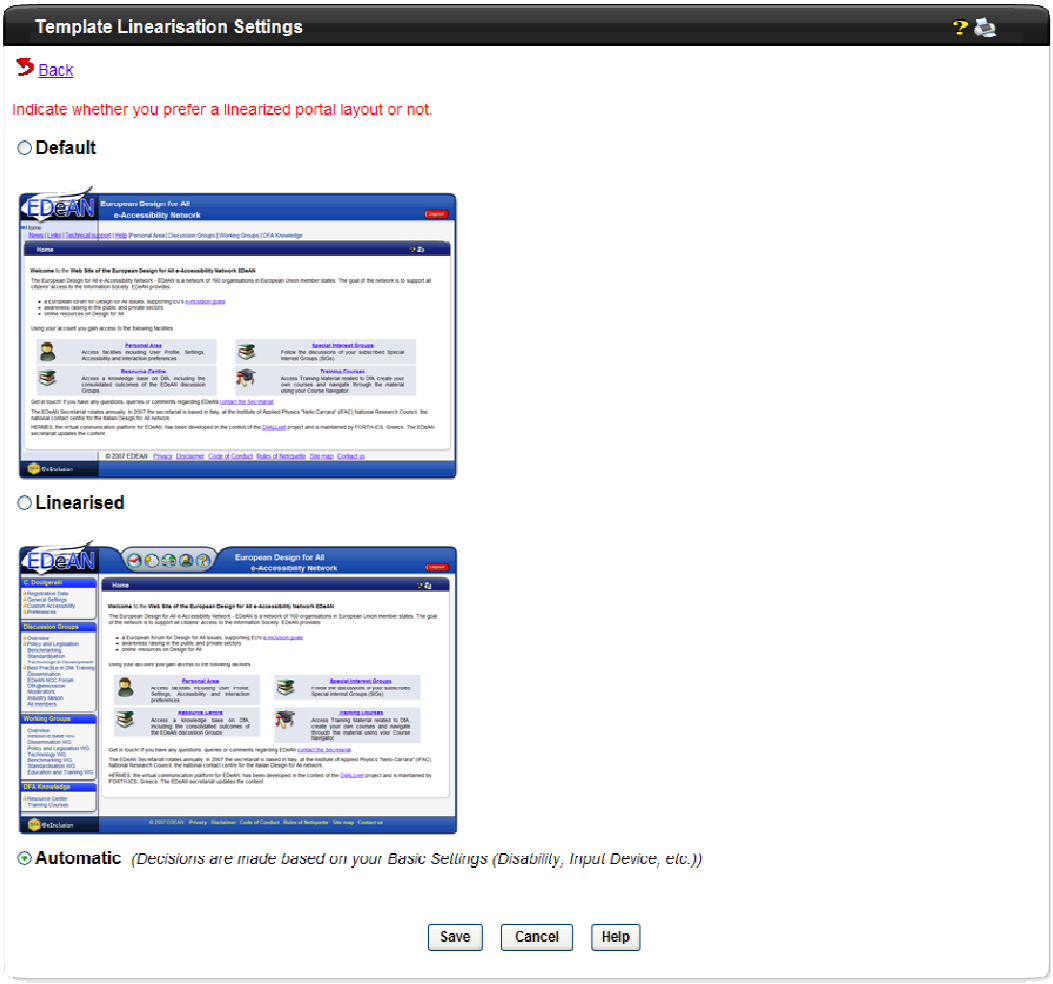


Figure 112: **Selecting the Template Linearization scheme**

Quick access links

Quick access links enable the fast transition to specific areas of the portal such as the sign – in area, page content, header etc. This facility is very important when user interaction is based on screen scanning (blind and motor impaired user). The selection of whether to display Quick Access Links is made using the interface presented in Figure 113.

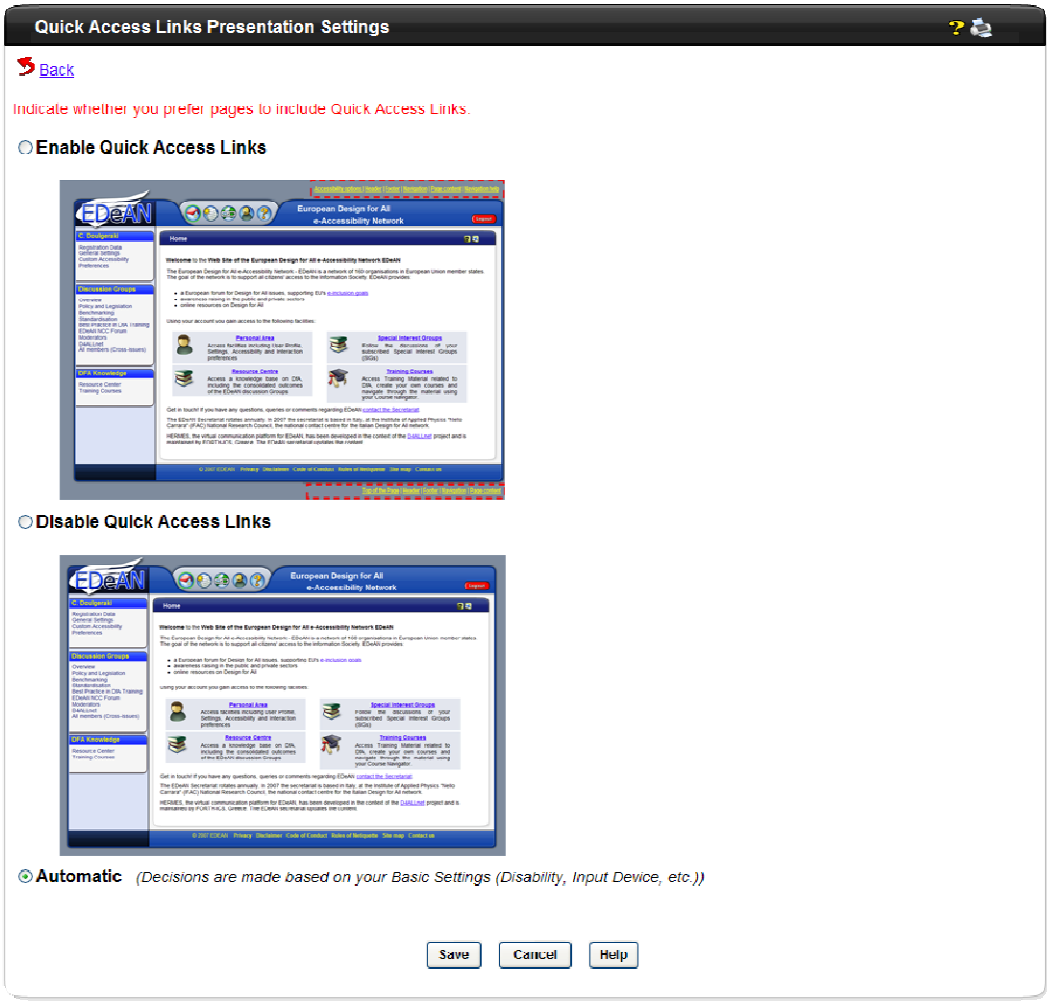


Figure 113: **Selecting whether to display Quick Access Links**

Dynamic adaptation

This setting affects all the Dynamic Adaptations that occur such as navigation hiding or navigation sorting based on popularity. The selection of the preferred scheme is carried out through the interface presented in Figure 114.

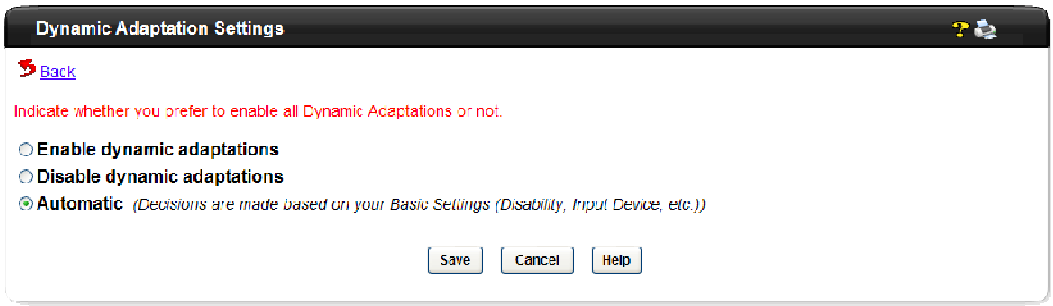


Figure 114: **Selecting whether to enable Dynamic Adaptation**

Section Breaks

This setting displays links that facilitate skipping to the end of each section, move to the start of the currently browsed section of move to the top of the page. Using these links users that interact via scanning can easily navigate back and forth to the main sections of the page without the need of rescanning a page. The selection of whether these Section Breaks should be displayed is carried out through the interface presented in Figure 115.

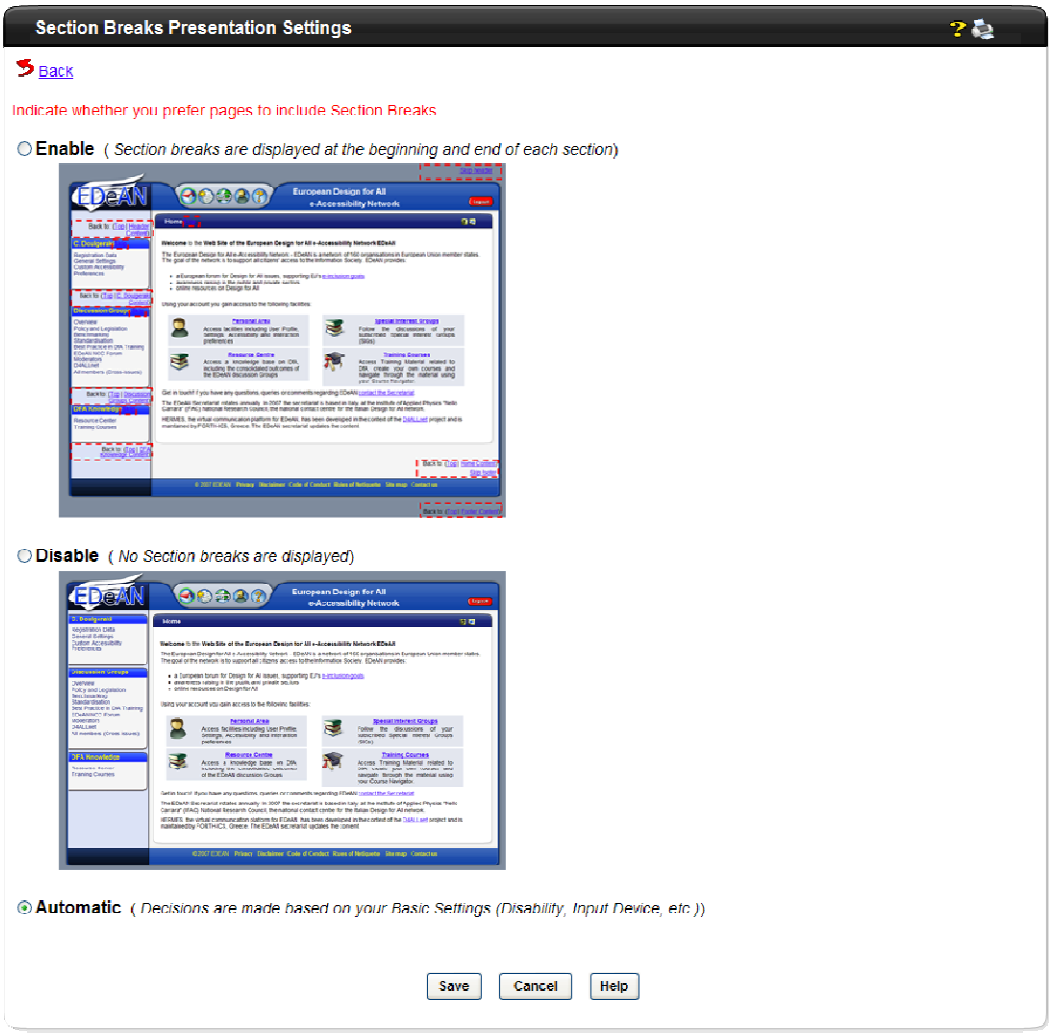


Figure 115: **Selecting whether to display Section Breaks**

Table linearization

The Table linearization setting affects initially whether tables presenting data will be linearized and furthermore the linearization scheme to be used. The interface used for selecting the linearization scheme is presented in Figure 116.

Figure 116: **The Table Linearization settings interface**

Charts

This setting affects the way that charts are presented. The available presentations schemes vary in relation to the palette used for rendering the chart. Additionally the option to render charts as tables is provided. The interface used for selecting the preferred chart presentation scheme is presented in Figure 117.

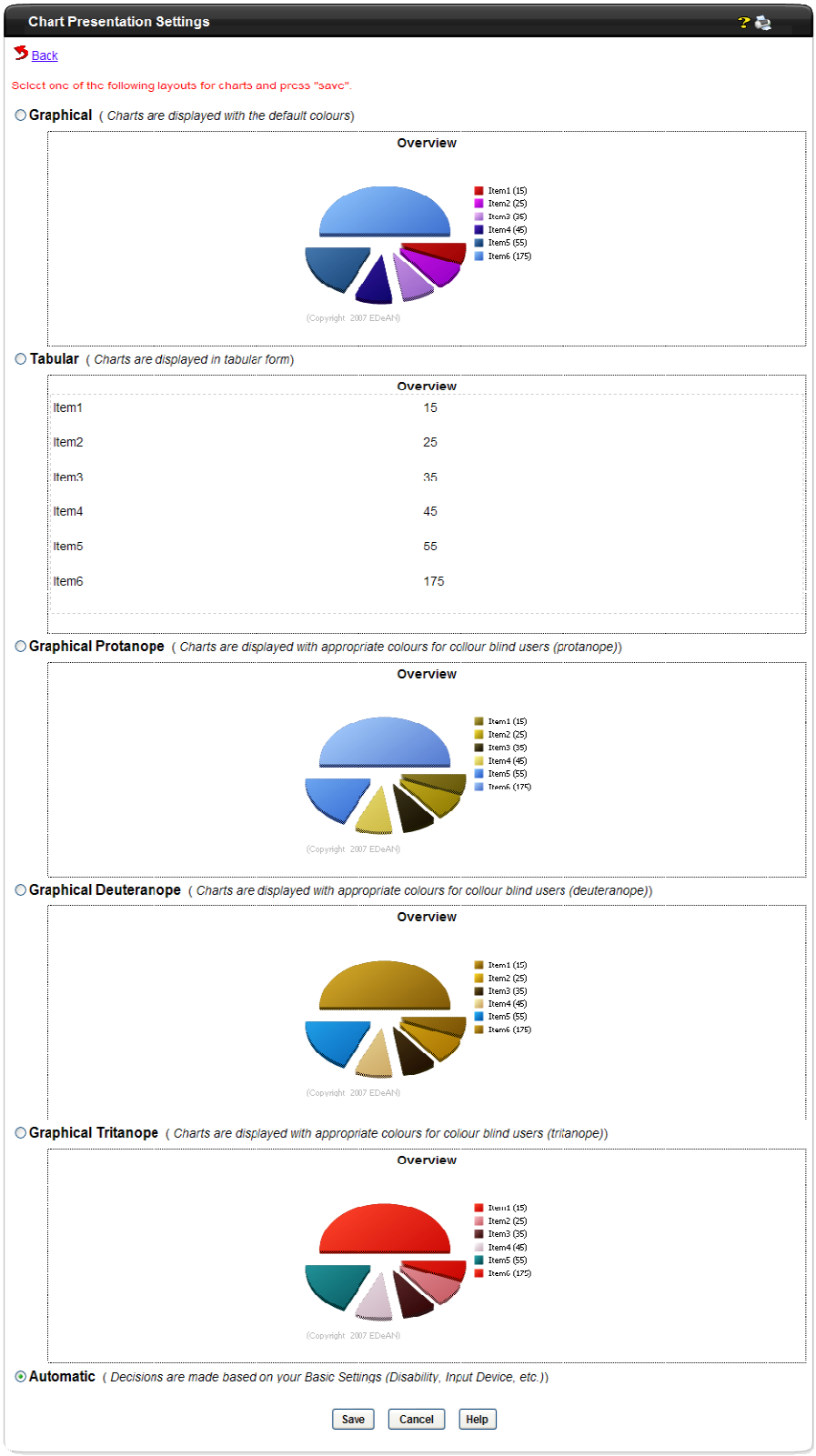


Figure 117: **The Chart settings interface**

Images

These settings can be used to define whether images will be rendered on page content and on page template. Additionally the way that images are displayed can be selected among a number of options (display as image, text etc). The interface provided for selecting among the aforementioned options is presented in Figure 118.

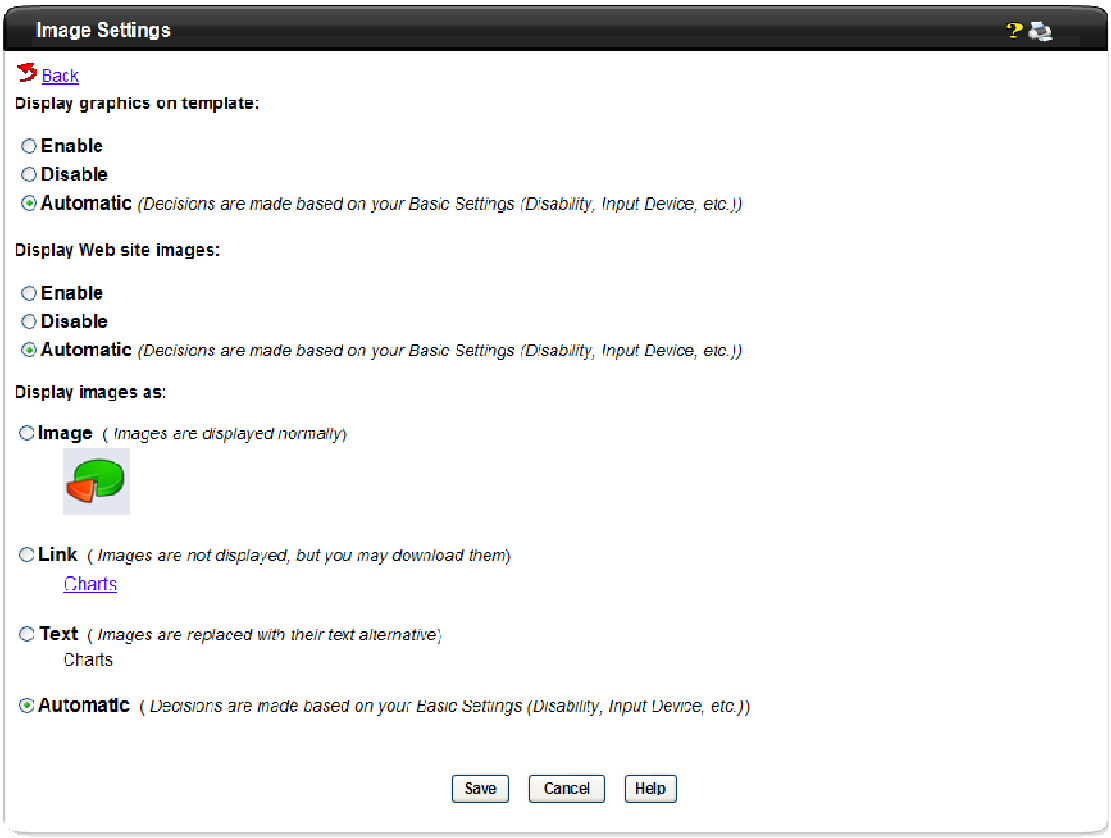


Figure 118: **The Image settings interface**

Fonts Sizes

The font settings affect the font family used for rendering interface and module content and moreover the font size used when rendering text. Users can select their preferred combination using the interface presented in Figure 119.

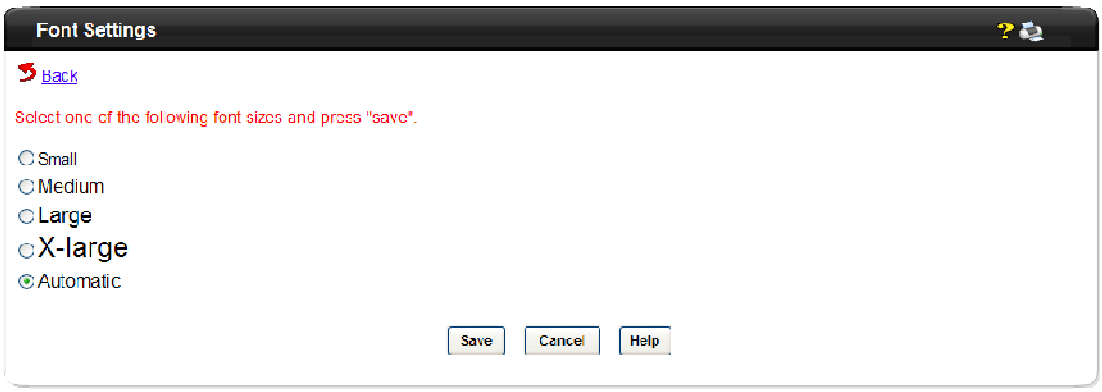


Figure 119: **The Font settings interface**

Colours

Colour settings affect the way the combination of background and foreground colours for links, buttons, labels and text. These settings are used to increase the contrast of these elements and therefore their visibility especially in case of colour blind users or users with low vision. Users through the interface presented in Figure 120 can preview and select their preferred combination.

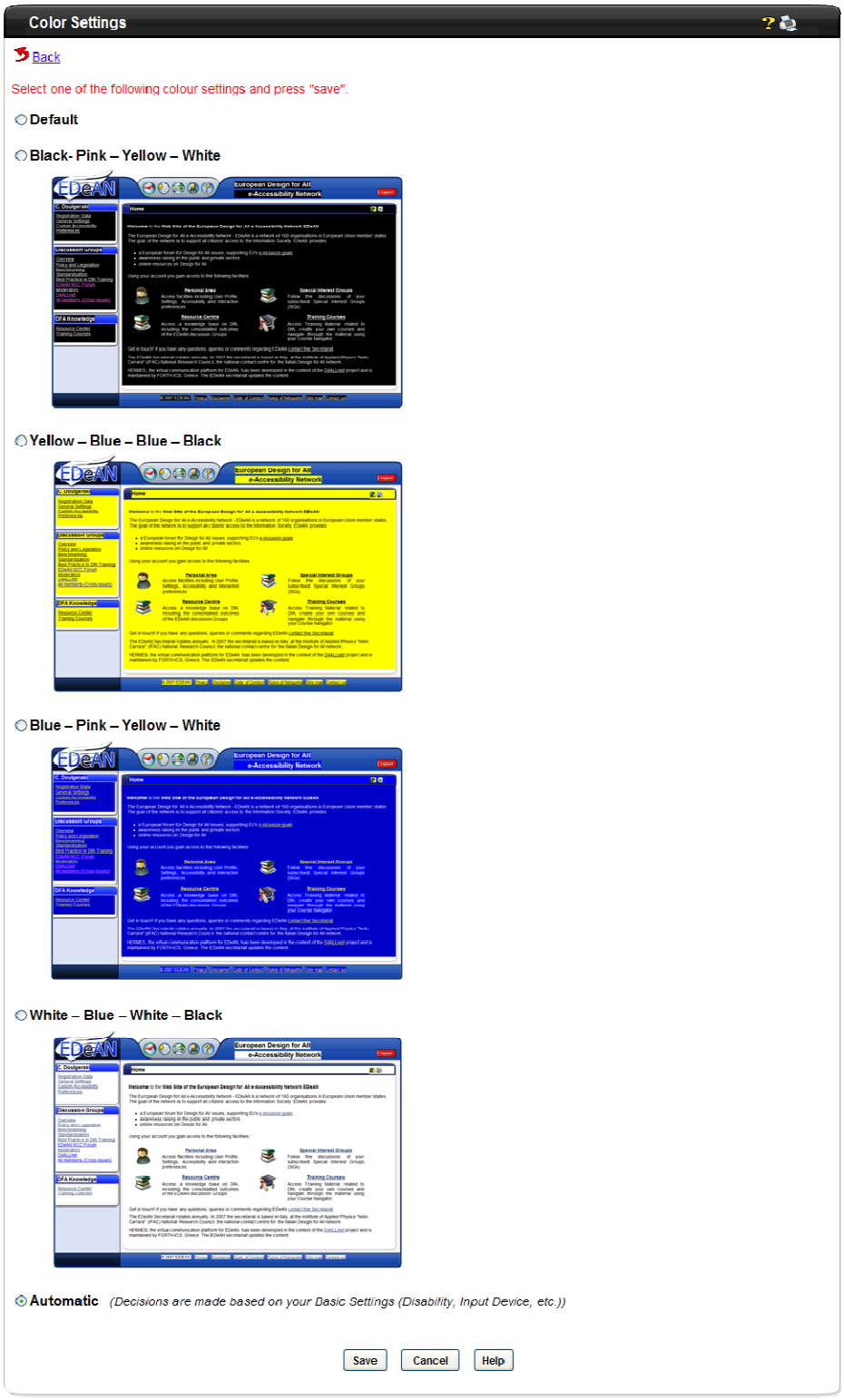


Figure 120: **The Colour settings interface**

Text entry

Text entry is very important for providing seamless access to all potential users of the application including users with disabilities. The way that content is inserted may vary according not only to the user characteristics but also according to the characteristics of the context of use. Potential variations may include virtual keyboards that simulate the physical QWERTY keyboards, virtual keyboards with reduced keys that can be operated together with scanning techniques of using mouse or with a pen on a PDA or tablet pc etc. The interface provided for selected among the different text entry alternative is presented in Figure 121.

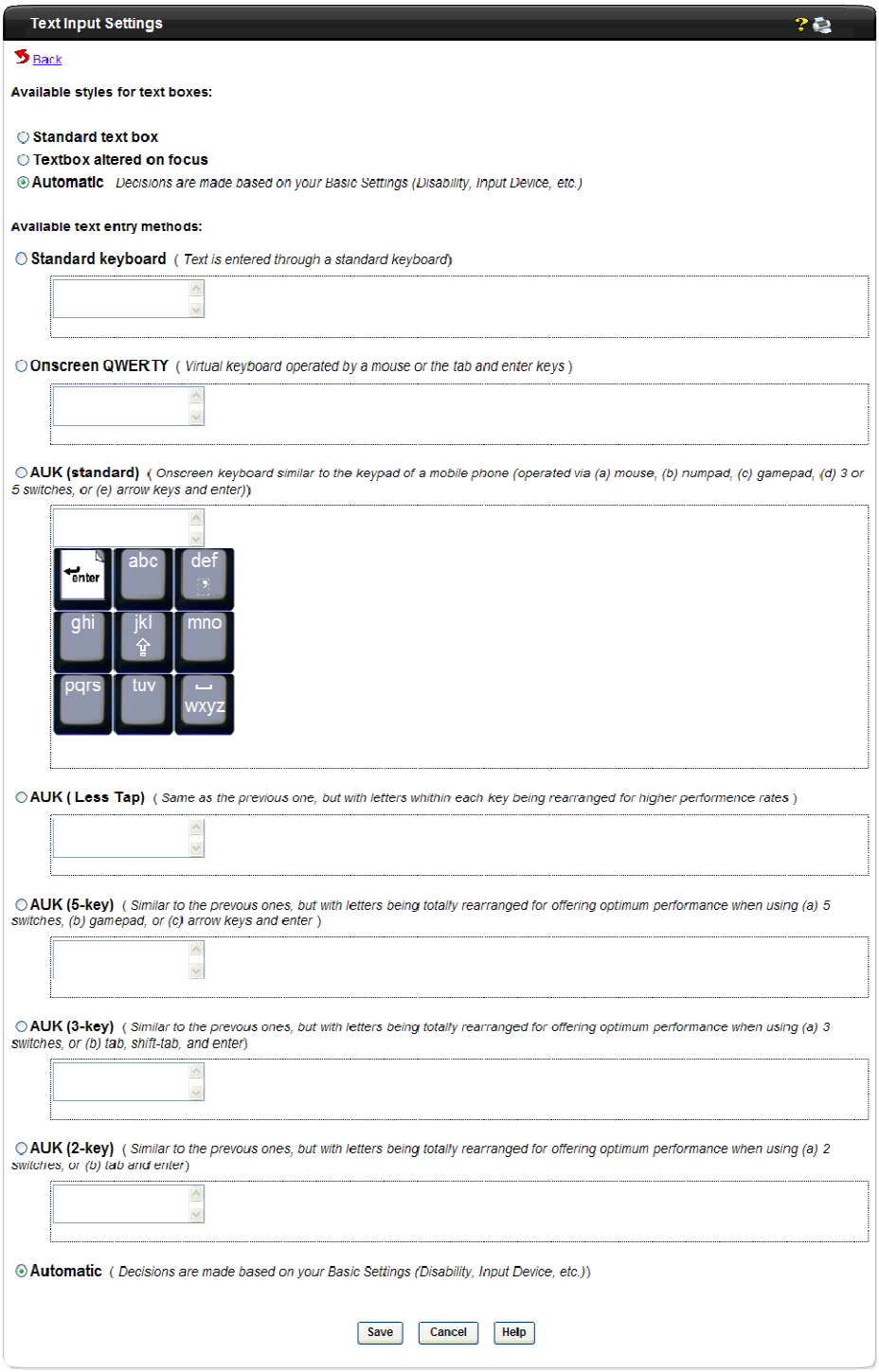


Figure 121: **The Text entry settings interface**

Fieldsets

Fieldsets are traditionally used in desktop and web application to group related information under a common title. The way that these fieldsets are presented affects the aesthetic representation of information but also its accessibility. In the case of sighted users a graphical fieldset can offer the visual feedback required but for non-sighted users the default representation must be used. The interface provided for selecting among the various fieldset presentations is displayed in Figure 122.

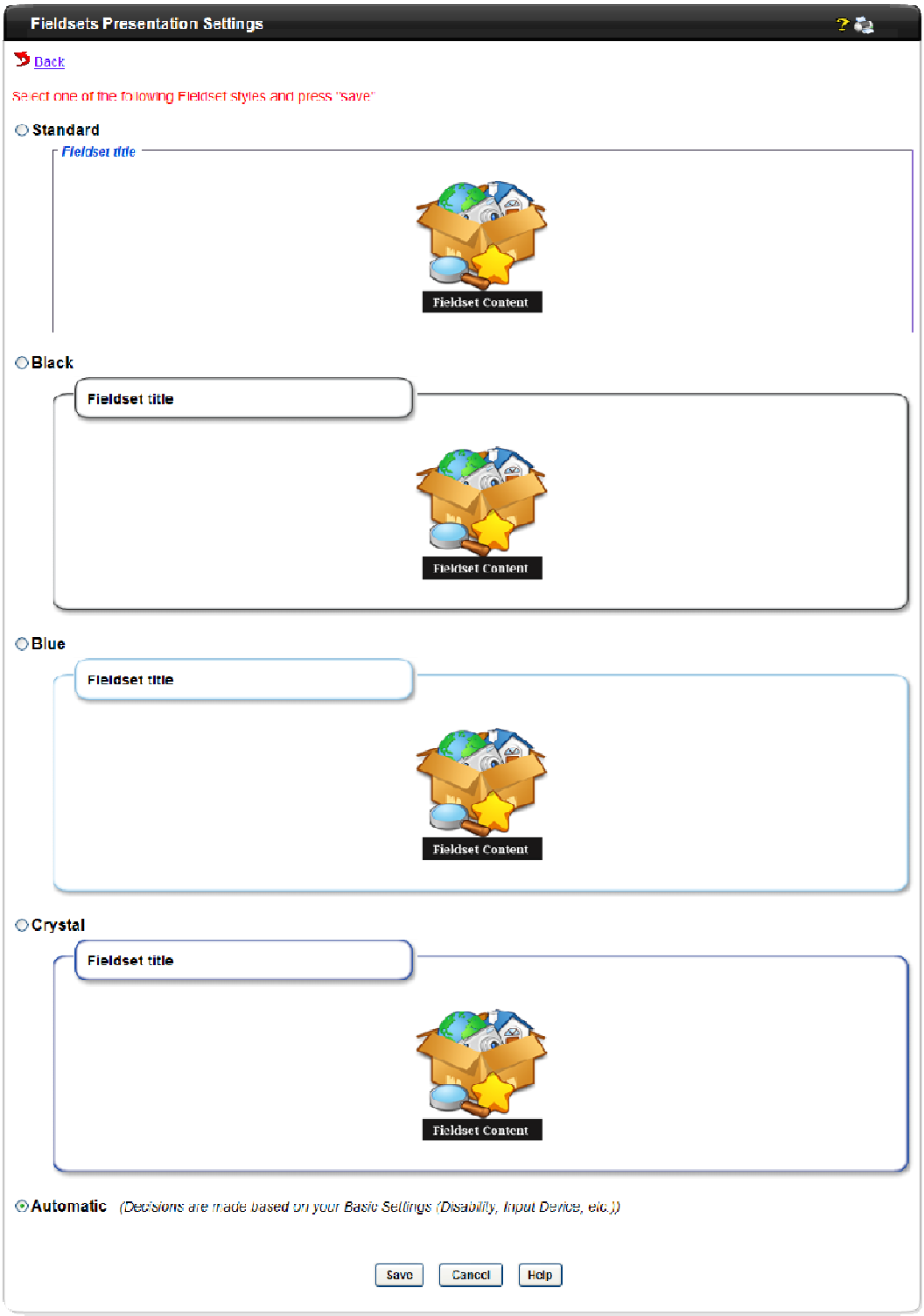


Figure 122: **The Fieldset presentation selection interface**

7.2.2.2 Special Interest Groups

Special Interest Groups (SIGs) are a recently established field of enquiry, and are usually defined as groups of people who interact in a virtual environment. They have a purpose, they are supported by technology, and they are guided by norms and policies [60]. A SIG is a virtual community composed of individuals who:

- Form and maintain online communication.
- Share common goals, interests and needs that provide the reason of community existence.
- Share common sets of rules and regulations that are accepted by all members and provide guidance to the community.
- Use a common technological platform that supports their networking and communication activities.
- Have common access to information.

SIG are, to an increasing extent, replacing or enhancing traditional ways of community formation, and cover a wider spectrum of thematic areas. The main objectives of online community support provision in the context of EDeAN were to help reducing time and personnel efforts, as well as preserving financial resources necessary for costly and time consuming frequent physical meetings. Additionally, the availability of a SIG infrastructure is targeted to allow the significant number of experts involved in this investigation to better focus on their scientific assignments.

Message Board

The message board is one of the most important virtual networking and communication facilities provided by the portal. It is specifically designed to support asynchronous one-to-many exchanges of messages between SIG members. Users of each SIG communicate by posting messages related to discussion topics. This structure was designed in order to group effectively and display a number of different discussions, which are grouped into different topics. User messages that are posted under the same topic are organized hierarchically. Additionally, the message board module offers the ability to SIG members to post replies to messages or to already replied messages. Through the messages interface of a topic, portal users have also the option to search the available messages using simple or search criteria.

Figure 123 provides an overview of the basic interfaces for browsing the available topics, topic messages and the facility offered for posting a new topic.

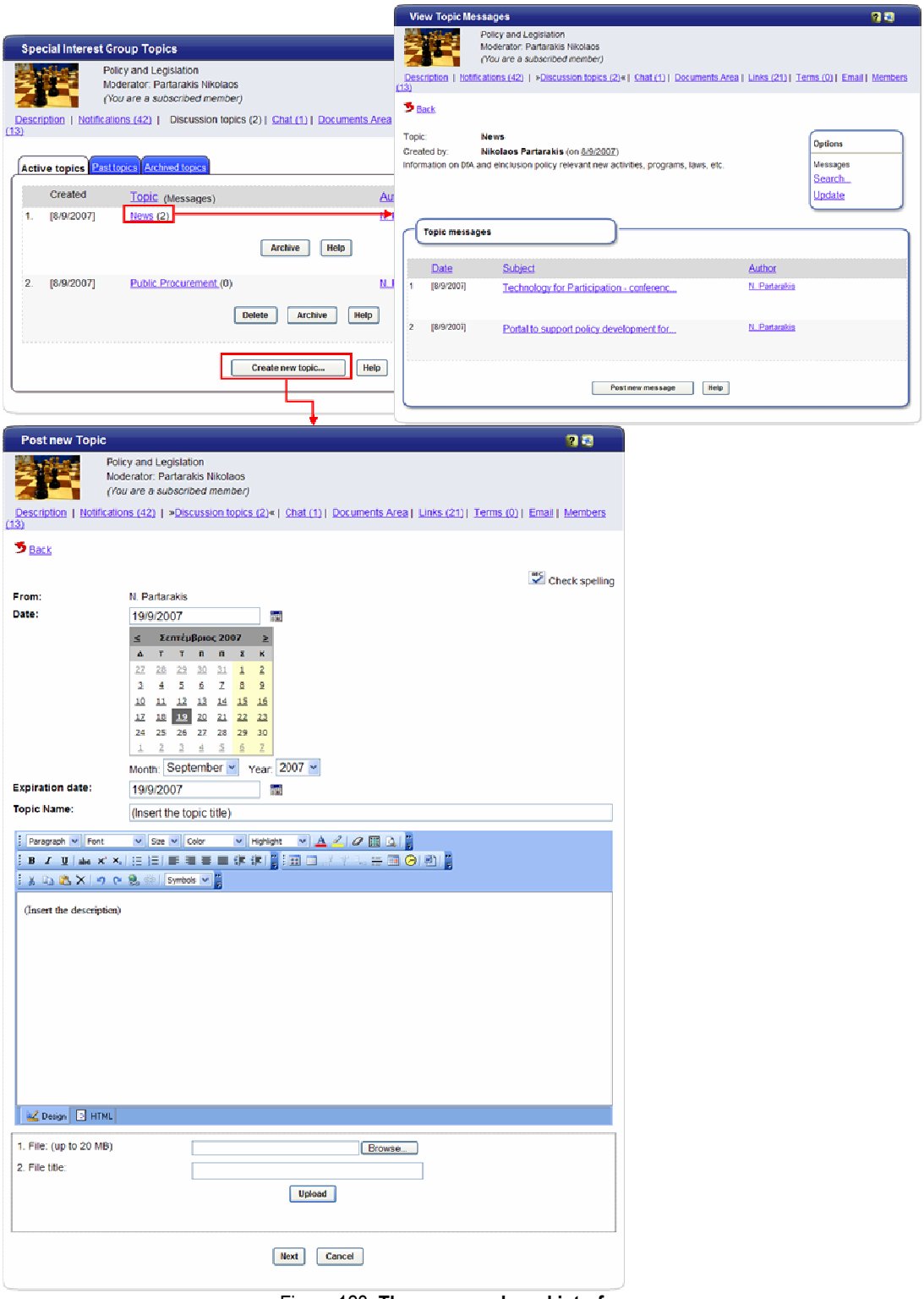


Figure 123: **The message board interfaces**

According to the specific user selections (setting, preferences and custom accessibility selections) the Message Board facility can be adapted to meet specific user requirements. A small subset of the possible adaptations for the post new topic interface is presented in Figure 124. More specifically, in this figure the following adaptations are highlighted:

1. The date selection dialogs are displayed as a collection of three dropdowns instead of their graphical counterpart.
2. A text editor with limited functionality is presented.
3. A file uploader with indirect manipulation of files is presented.
4. Module functions are presented with their default help description.

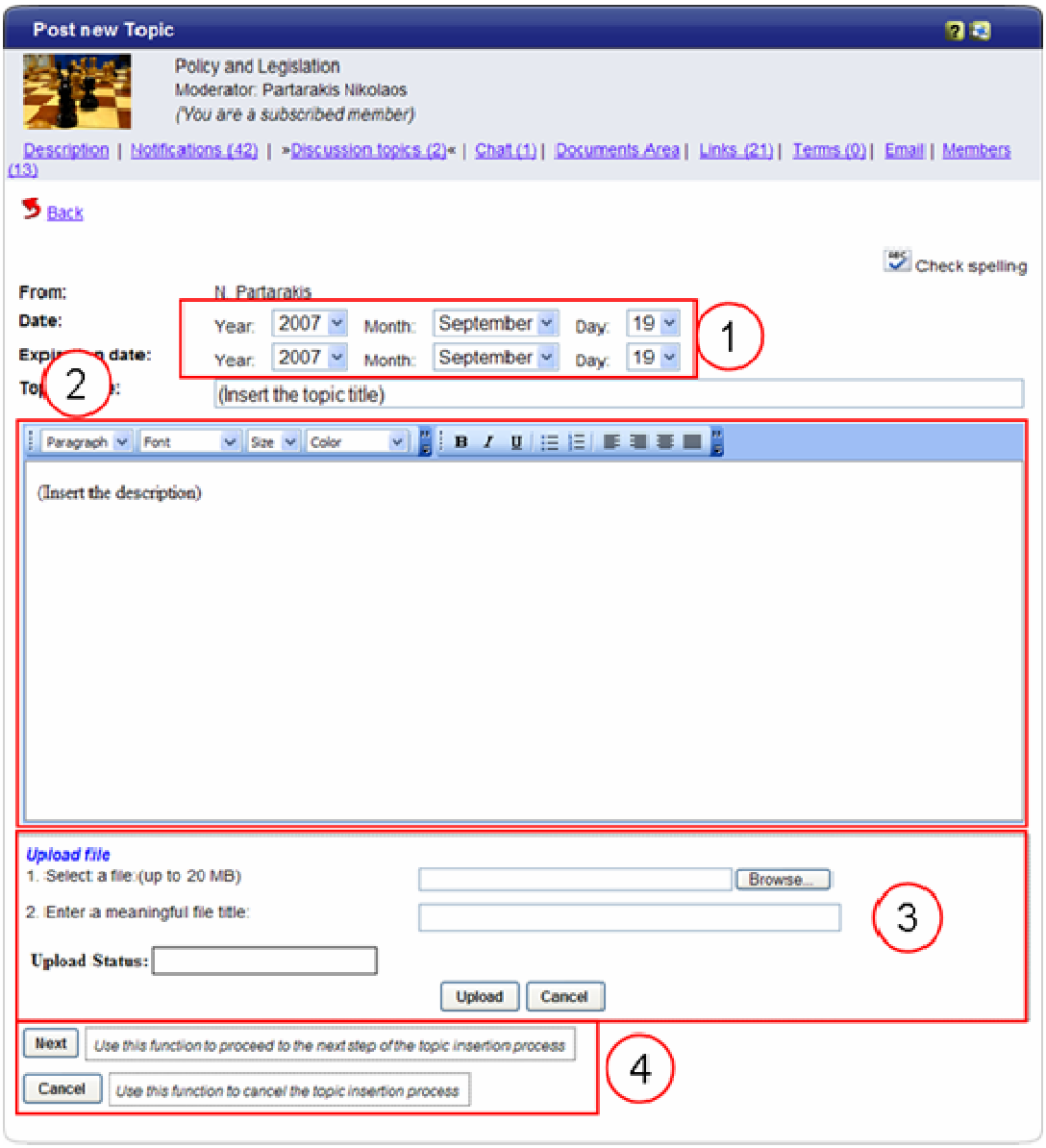


Figure 124: **A subset of possible adaptations for postings new topics**

Documents Area

The documents area of a SIG is a virtual space available for storing, organizing and retrieving documents and other files related to the activities of each SIG. Files contained in the documents area are organized in a file system like hierarchy of folders. This folder-file hierarchy can be expanded to contain up to n-levels of files and folders. All necessary functionality to ensure effective content management is provided. In particular, users are provided with facilities for:

- Browsing the hierarchy
- Searching files
- Administrating uploaded documents

Figure 125 provides an overview of the available browsing and administrating facilities offered by the SIG Documents Area.

Figure 125: **The Documents Area module (default view)**

According to the specific user selections (setting, preferences and custom accessibility selections) the Documents Area facility can be adapted to meet specific user requirements. A small subset of the possible adaptations for the documents area module is presented in Figure 126. In more detail, in this figure the following adaptations are highlighted:

1. The image buttons for help and print have been transformed to links
2. The options of the documents area module have been transformed to tab pages represented as links
3. The document hierarchy is always expanded
4. The module functions are rendered together with their help description
5. Section breaks are displayed at the top and bottom of the document area module.

Other adaptations include the transformation of the module window to a simple border and the removal of background colours from the module content.

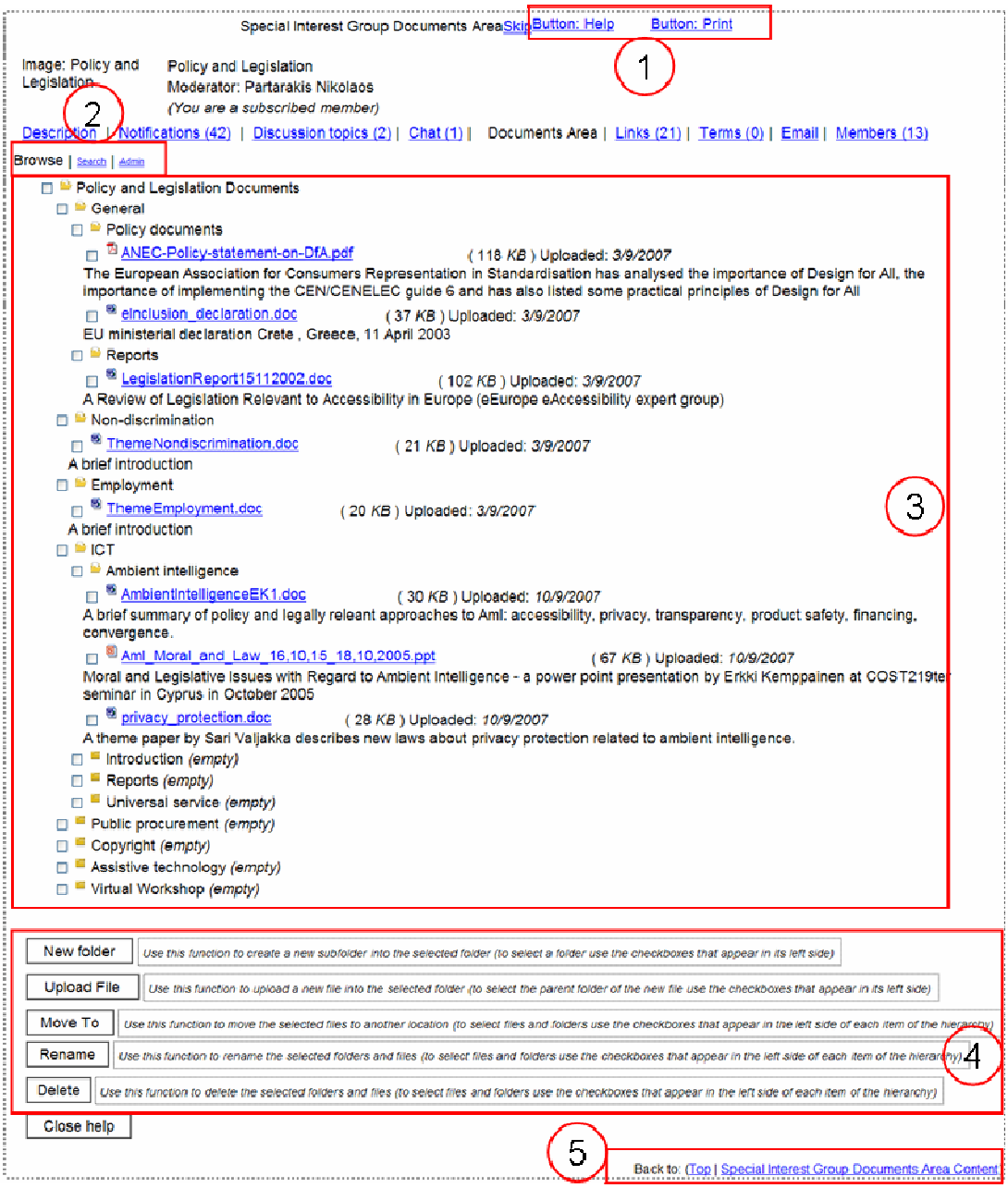


Figure 126: **The Documents Area module (adaptation example)**

Chat

This module is specifically designed to support real-time one-to-one or one-to-many exchanges of text messages between SIG members. Discussions are organized in a number of Chat rooms. Figure 127 provides an overview of the initial Chat Rooms interface followed by the actual discussion area.

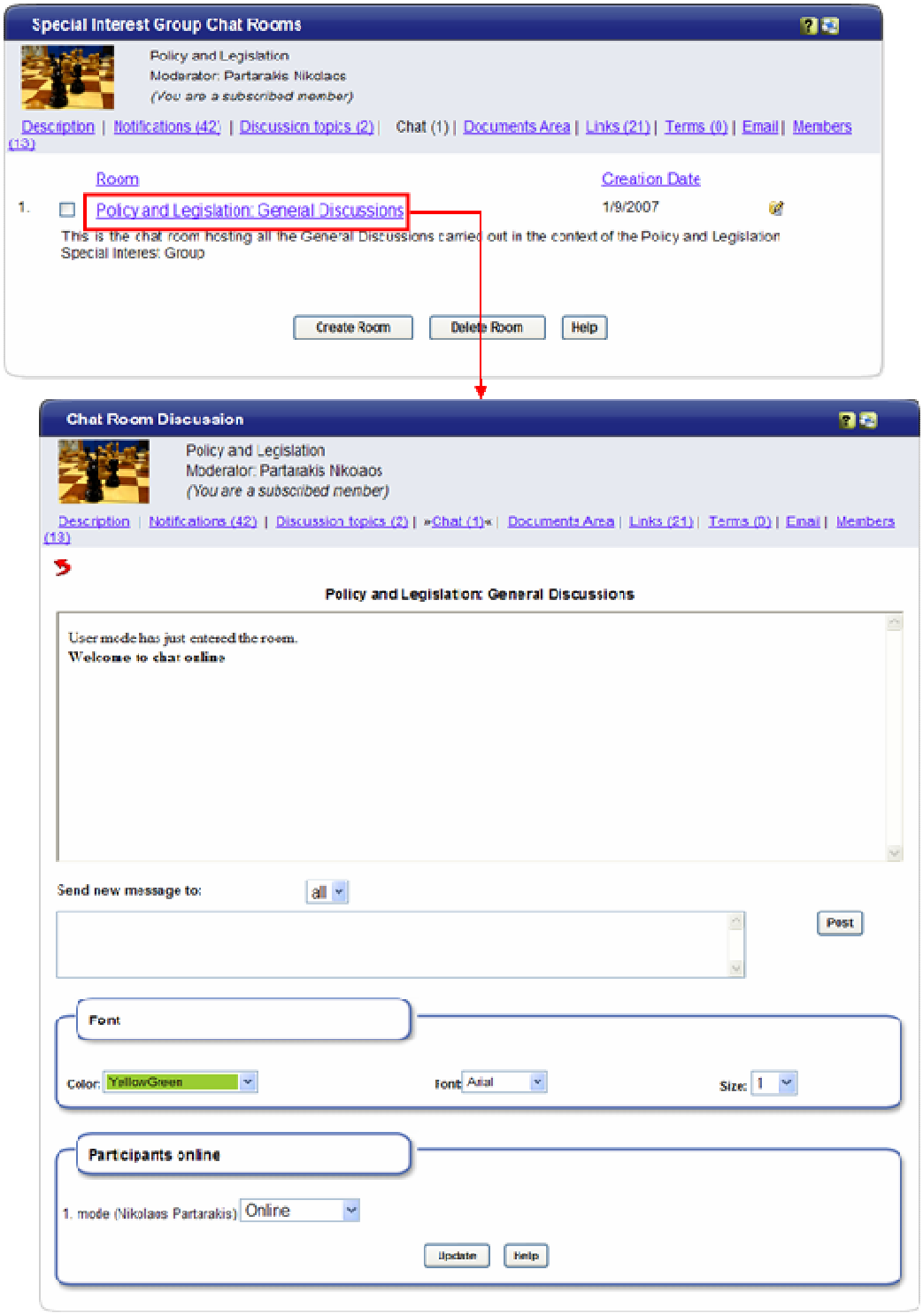


Figure 127: **The Chat facility (default view)**

According to the specific user selections (setting, preferences and custom accessibility selections), the Chat facility can be adapted to meet specific user requirements. A small subset of the possible adaptations for the chat module is presented in Figure 128. In this figure, the following adaptations are highlighted:

1. The Special Interest Group navigation options are transformed to buttons
2. Text fields provide visual feedback on focus
3. A software keyboard is presented for entering the message text
4. Fieldsets are presented with their standard HTML presentation instead of their graphical counterpart
5. Module functions are presented with the option to request optional help
6. Section breaks are displayed as buttons instead of links.

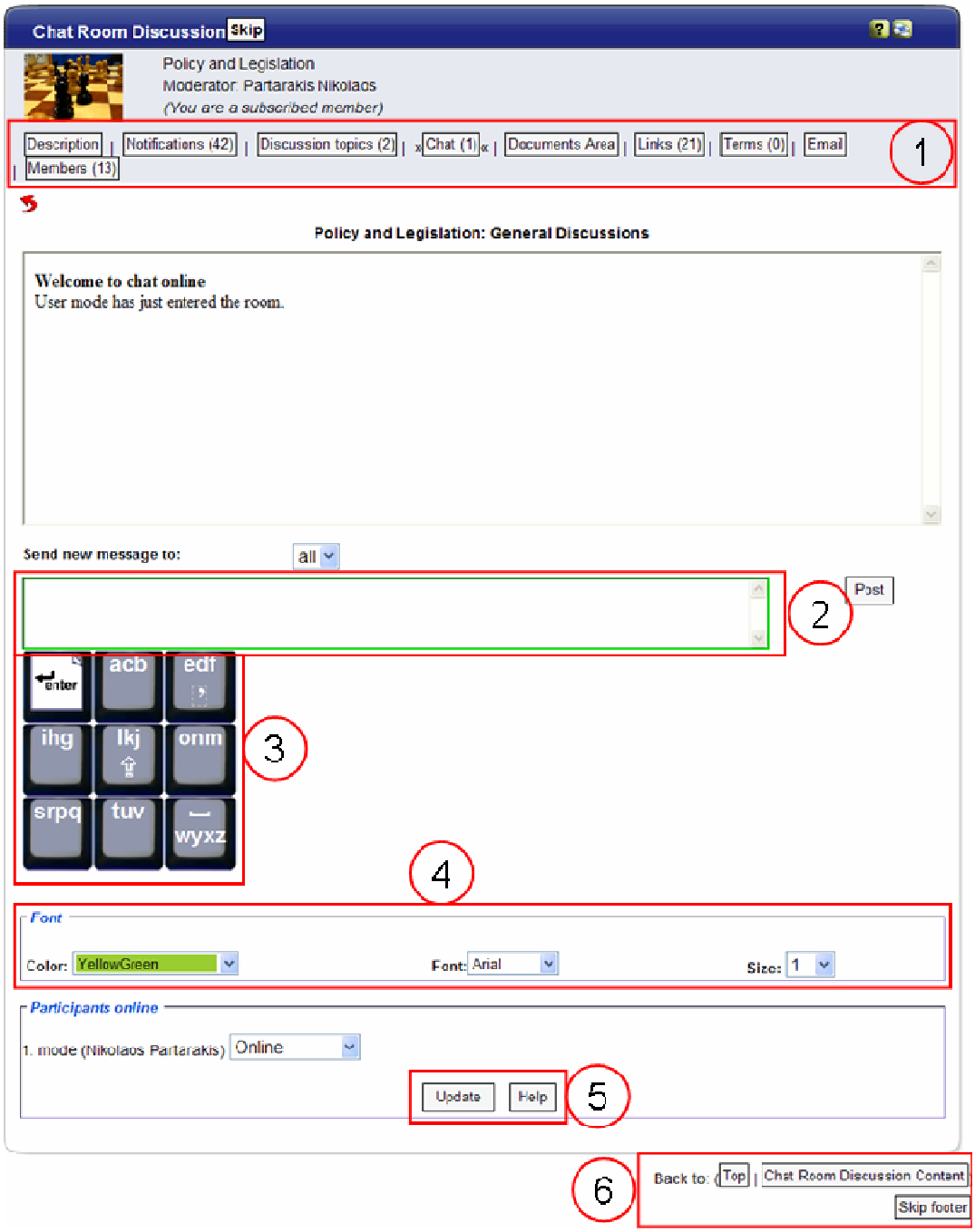


Figure 128: **The Chat facility (adaptation example)**

Links

The links module provides access to links that are related to the subject or scope of the specific SIG. List browsing facilities are provided to SIG members together with the possibility to access further information about a specific link, as shown in Figure 129. SIG moderators are responsible with the administration of the links repository provided with additional facilities for adding-editing or deleting.

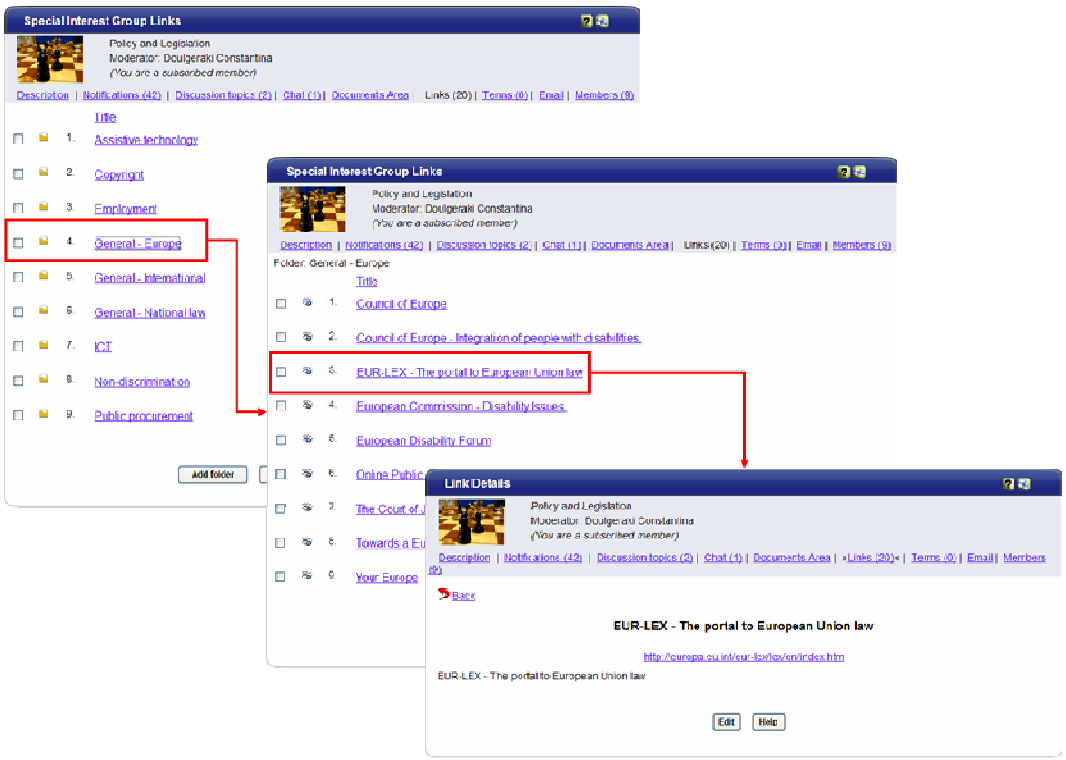


Figure 129: **Link browsing process (default view)**

According to the specific user selections (setting, preferences and custom accessibility selections), the Links facility can be adapted to meet specific user requirements. A small subset of the possible adaptations for the Links home page is presented in Figure 130. In this figure, the following adaptations are highlighted:

1. The table summary is displayed
2. The table representing the available folders is linearized
3. The module functions are presented using blue buttons.

Other adaptations include the transformation of the module window to a simple border, the removal of background colours from the module content, the presence of yellow colour for background and black for foreground and the significant increase of font size.

Figure 130: **Link browsing process (adaptation example)**

Notifications

The notifications facility plays a very important role in the synchronisation of activities within a SIG. The activities carried out through the portal involve potentially a very large number of users who must cooperate. Additionally, the concept of cooperation involves the need of notifying users about actions that affect the community, such as the initiation of new discussions or the addition of a new shared document. Towards facilitating collaboration, the notification functionality provides a user based repository of informative messages. These messages are categorised based on the related module. Figure 131 provides an overview of the initial notifications page, where notifications are categorised according to the available SIG modules and the page displaying the detailed view of the available Notifications for a specific category.

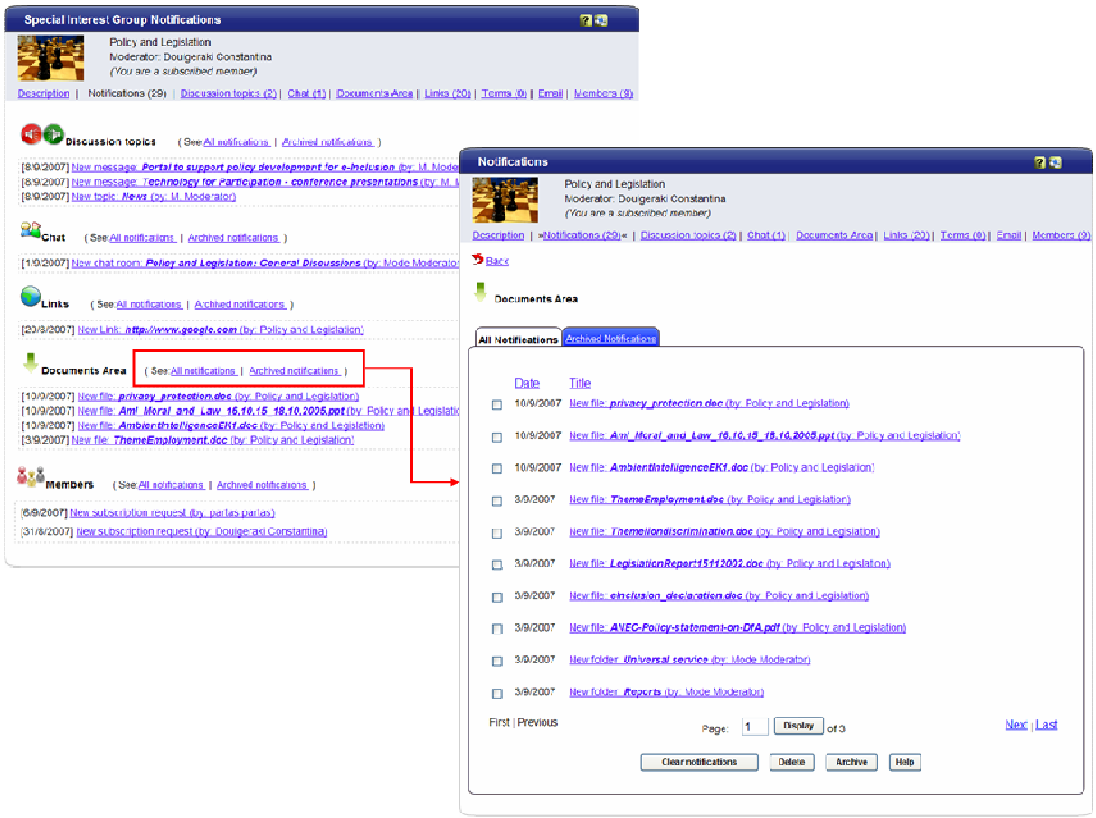


Figure 131: **Browsing Notifications (default view)**

According to the specific user selections (setting, preferences and custom accessibility selections) the Notifications facility can be adapted to meet specific user requirements. A small subset of the possible adaptations for the Notifications per Category page is presented in Figure 132. In this figure, the following adaptations are highlighted:

1. Links are presented as buttons
2. Tabs are displayed as links and due to adaptation #1 as buttons
3. The paging facility is transformed to a simple next-previous paging variation
4. Functions are presented for expert users without the help option.

Other adaptations include the presence of black colour for background and white for foreground, the presentation of buttons with yellow colour for background red for content and black for button text and the significant increase of font size.

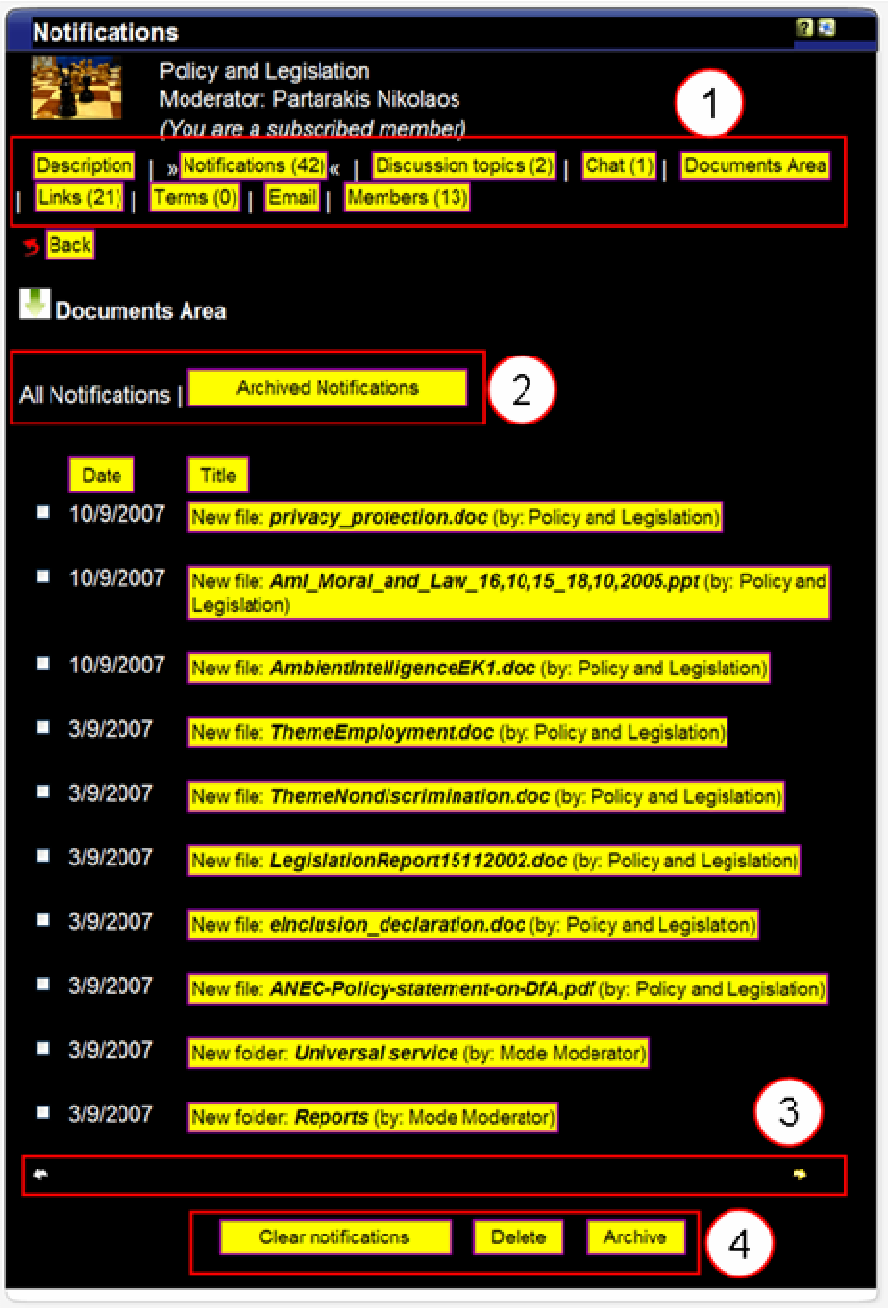


Figure 132: **Browsing Notifications (adaptation example)**

Terms (Glossary)

The terms module is a glossary providing access to acronyms definitions and abbreviations of terms that are of importance or relevance to the activities of a specific SIG. Members of a SIG have access to facilities for browsing and searching the repository of terms. Moderators have the responsibility of maintaining the repository of terms and therefore are provided with administration tools for adding, editing or deleting terms. Figure 133 provides an overview of the facilities offered by the SIG glossary for indexing or searching terms, followed by the interface presenting the detailed view of a selected term.

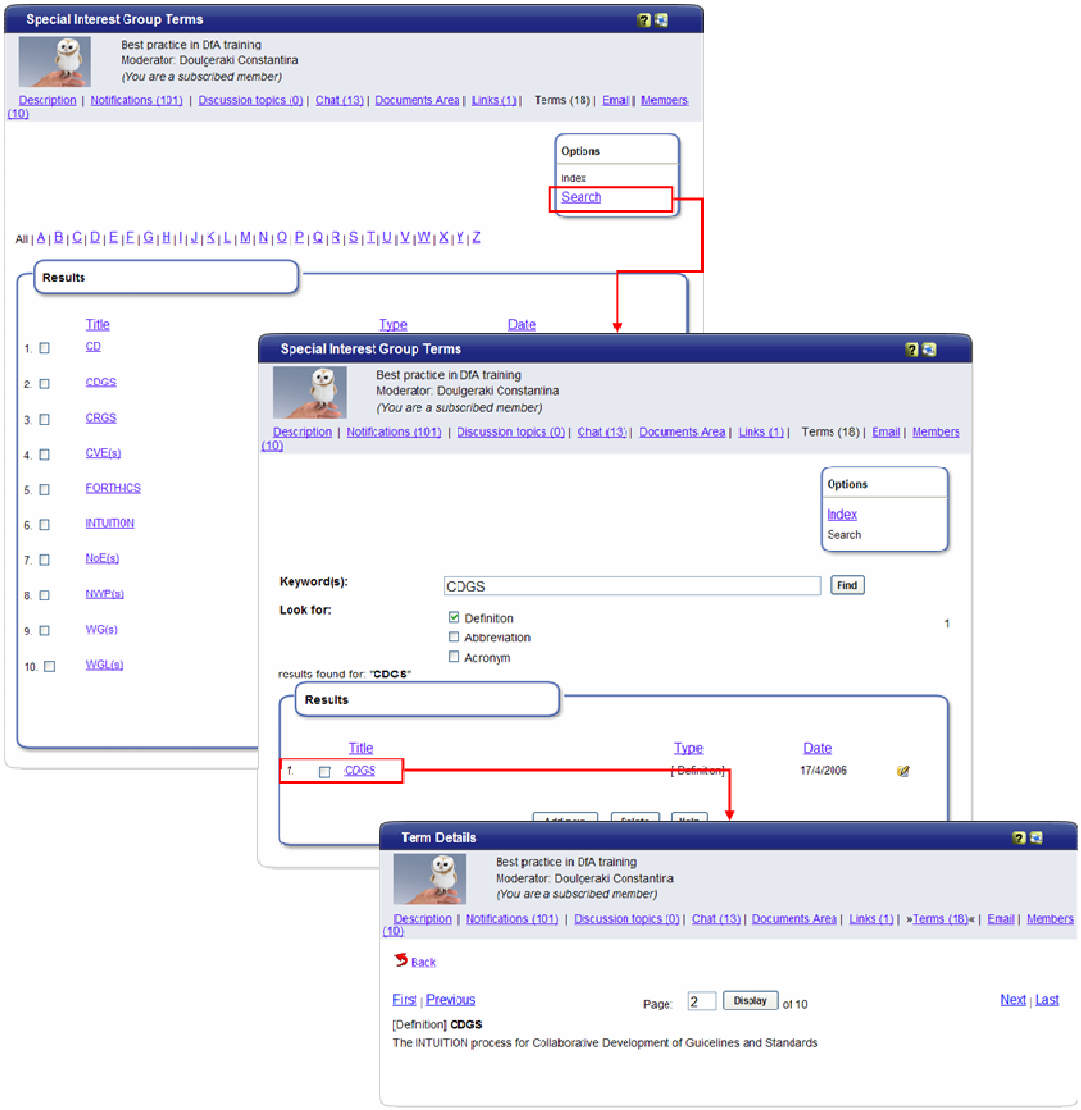


Figure 133: **The Terms module (default view)**

According to the specific user selections (setting, preferences and custom accessibility selections) the Terms facility can be adapted to meet specific user requirements. A small subset of the possible adaptations for the Terms Index page is presented in Figure 134. In this figure, the following adaptations are highlighted:

1. Text is presented using the Comic Sans MS font family and large font size
2. Page options are displayed as tabs using the crystal blue skin
3. Fieldsets are presented with their standard presentation instead of the graphical counterpart
4. All available terms are displayed in one page without paging.

Figure 134: **The Terms module (adaptation example)**

Email

The E-mail facility allows users of a SIG to communicate via email. This module provides access to the SIG subscribed users for enabling the selection of the email receivers. The E-mail facility provided through the Hermes portal is not a fully functional web-mail implementation, and facilitates only the process of submitting emails to SIG participants. The Email facility is presented in Figure 135.

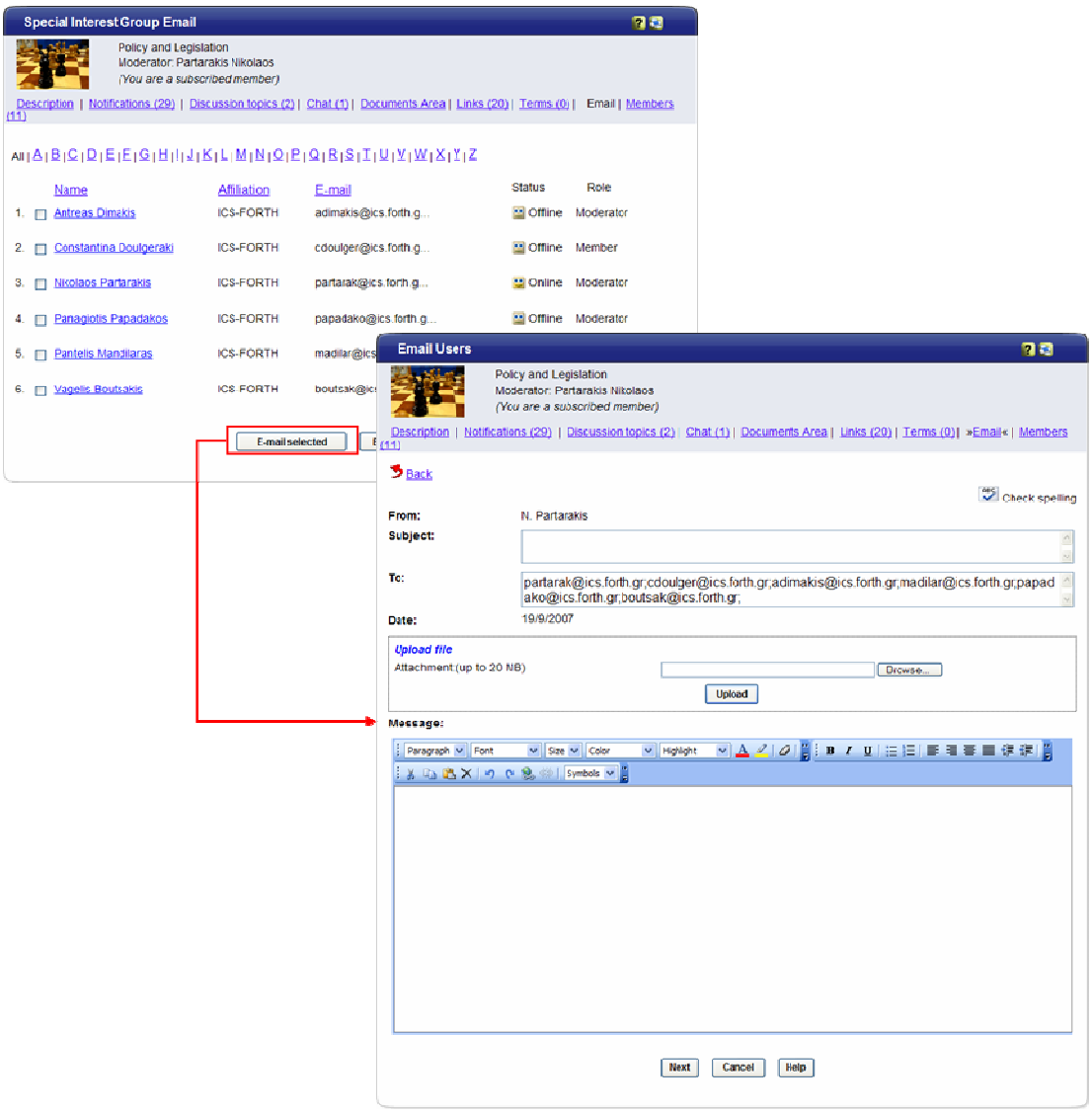


Figure 135: **The Email facility (default view)**

According to the specific user selections (setting, preferences and custom accessibility selections), the Email facility can be adapted to meet specific user requirements. A small subset of the possible adaptations for the Email page is presented in Figure 136. In this figure, the following adaptations are highlighted:

1. Text fields offer feedback when focused
2. A file uploader for expert users is presented facilitating the direct manipulation of files in the case of submission and deletion
3. A non graphical editor with a software R-standard keyboard is presented instead of its graphical counterpart.

Figure 136: **The Email facility (adaptation example)**

Members

This module was designed and implemented in order to support the Moderators' day-by-day administration of their SIGs. Through the functions provided, the SIG Moderator can view the list of SIG members and some of their personal details (e.g., contact information, member's participation). Moderators can view the personal details of a specific member and also perform registration operations using the invitation mechanism. Finally, Moderators are provided with ability to alter the message sent during invitation.

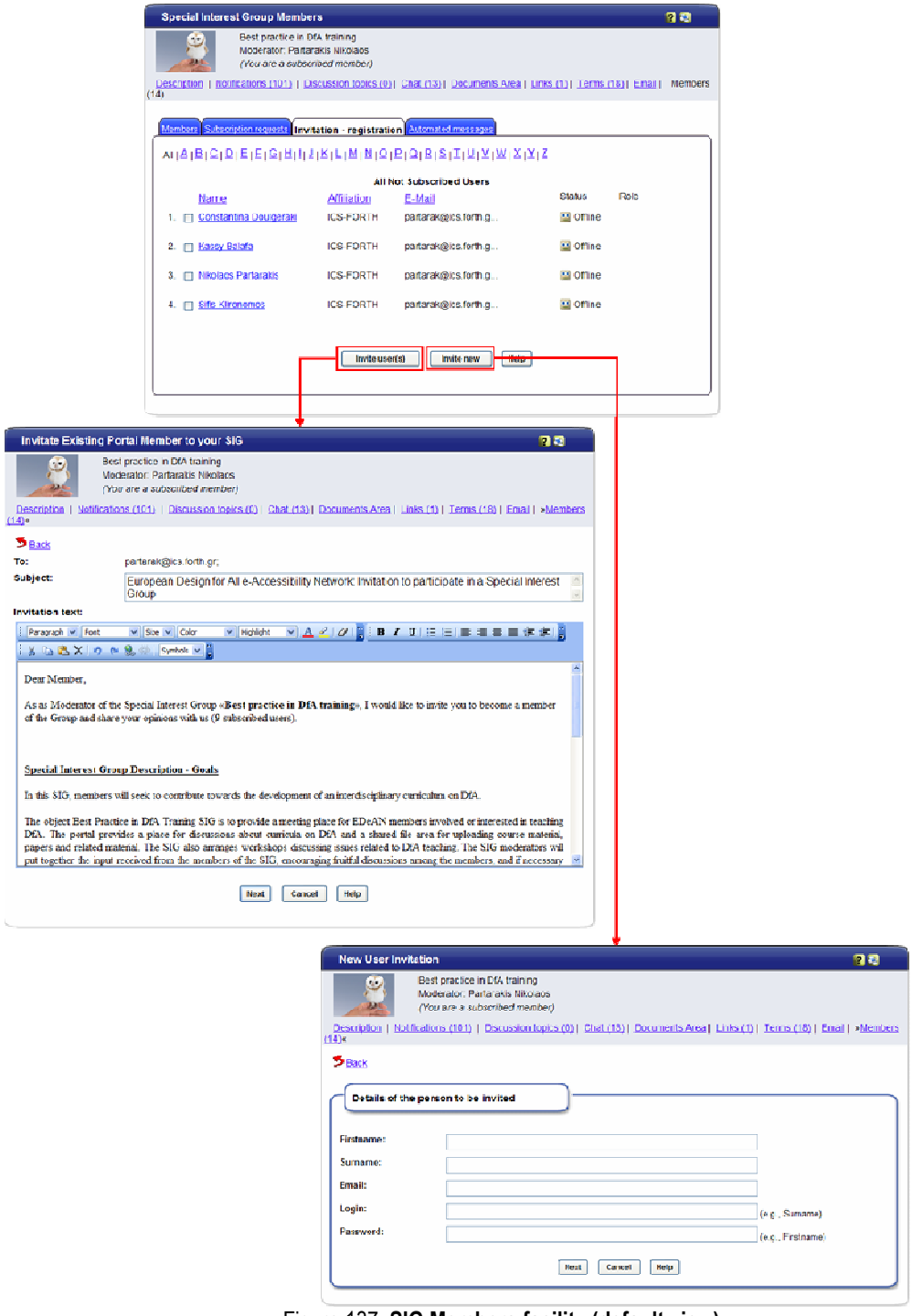


Figure 137: **SIG Members facility (default view)**

According to the specific user selections (setting, preferences and custom accessibility selections), the Members facility can be adapted to meet specific user requirements. A small subset of the possible adaptations for the Members home page is presented in Figure 138.

In this figure, the following adaptations are highlighted:

1. Tabs are transformed to options appearing inline with page content and on the right side
2. Table summary is displayed
3. The table that represents members is linearized by row
4. Functions are presented with their default help description and buttons are presented with yellow colour for background and red for button border

Other adaptations include the presence of blue colour for background and white for foreground and the presentation of links with pink colour.

Figure 138: **SIG Members facility (adaptation view)**

7.2.2.3 DfA Knowledge tools

E-learning facility

The E-learning facility is an alternative medium for disseminating the project's results. The contents of the E-learning facility are targeted to anyone interested in Design for All methods.

The E-learning facility offers the following material and functionality:

- Brief, as well as in depth information about design approaches and methods
- Several examples, exercises, presentations, references and links related to these methods
- Interactive tools for comparing and selecting appropriate methods
- A mechanism for personalizing the course's content according to different user
- A tool for developing alternative custom courses
- A course history facility
- A search facility.

Figure 139 provides an overview of the process followed for browsing training material, while Figure 140 presents the available facilities for accessing personal courses.

Figure 139: **EDeAN Training Course Browsing Facilities (default view)**

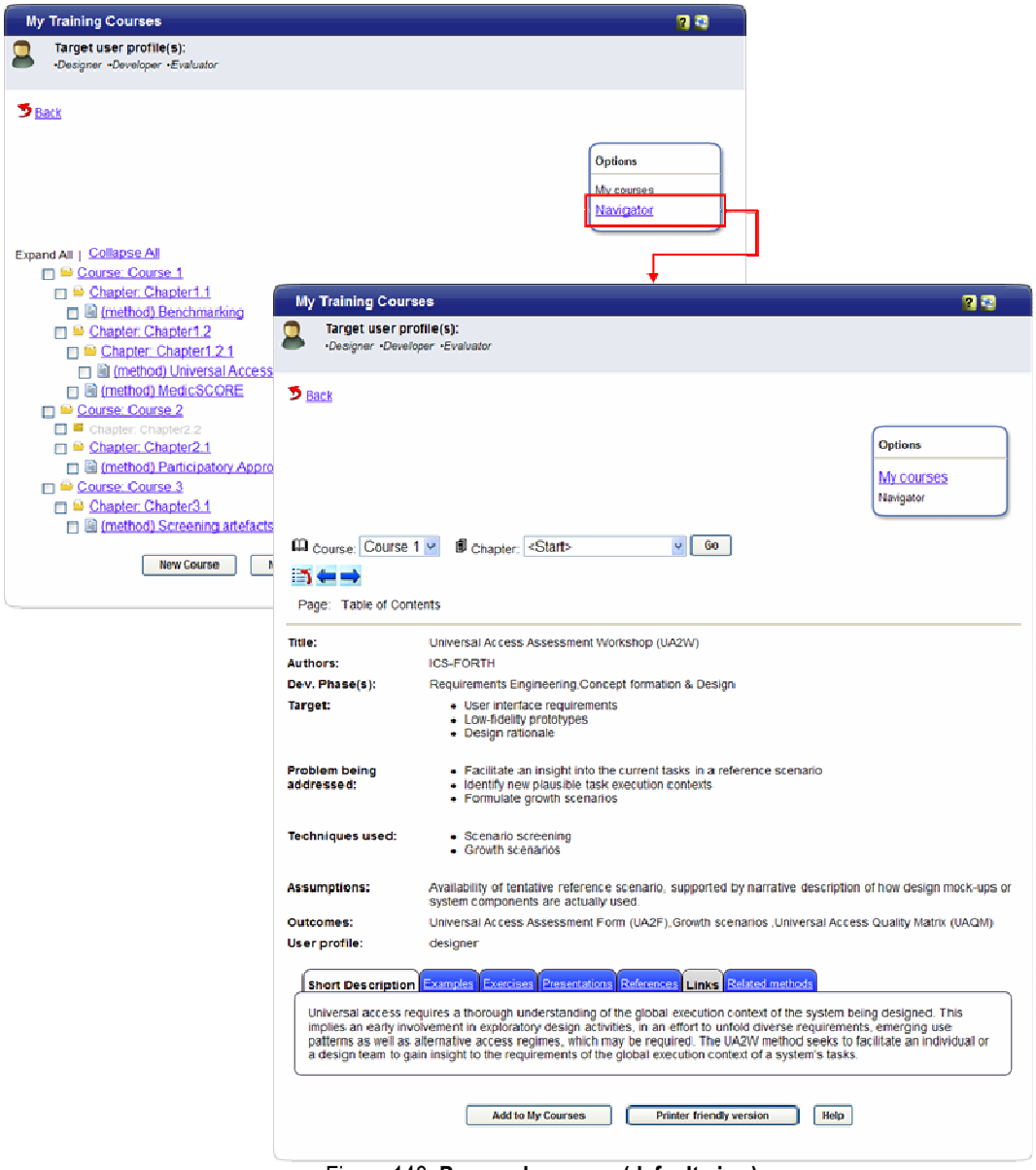


Figure 140: **Personal courses (default view)**

The facilities offered by the EDeAN online Training Course can be adapted to meet specific user requirements according to the specific user selections (setting, preferences and custom accessibility selections). A small subset of the possible adaptations for the Training Course Navigator page is presented in Figure 141. In this figure, the following adaptations are highlighted:

1. Images are displayed as text
2. Page options are transformed to links in conjunction to their graphical counterparts
3. Image buttons are transformed to links
4. The tab that presents the material details is linearised
5. Functions are presented with their default help description.

Other adaptations include the transformation of the page window to a simple border and the presence of section breaks.

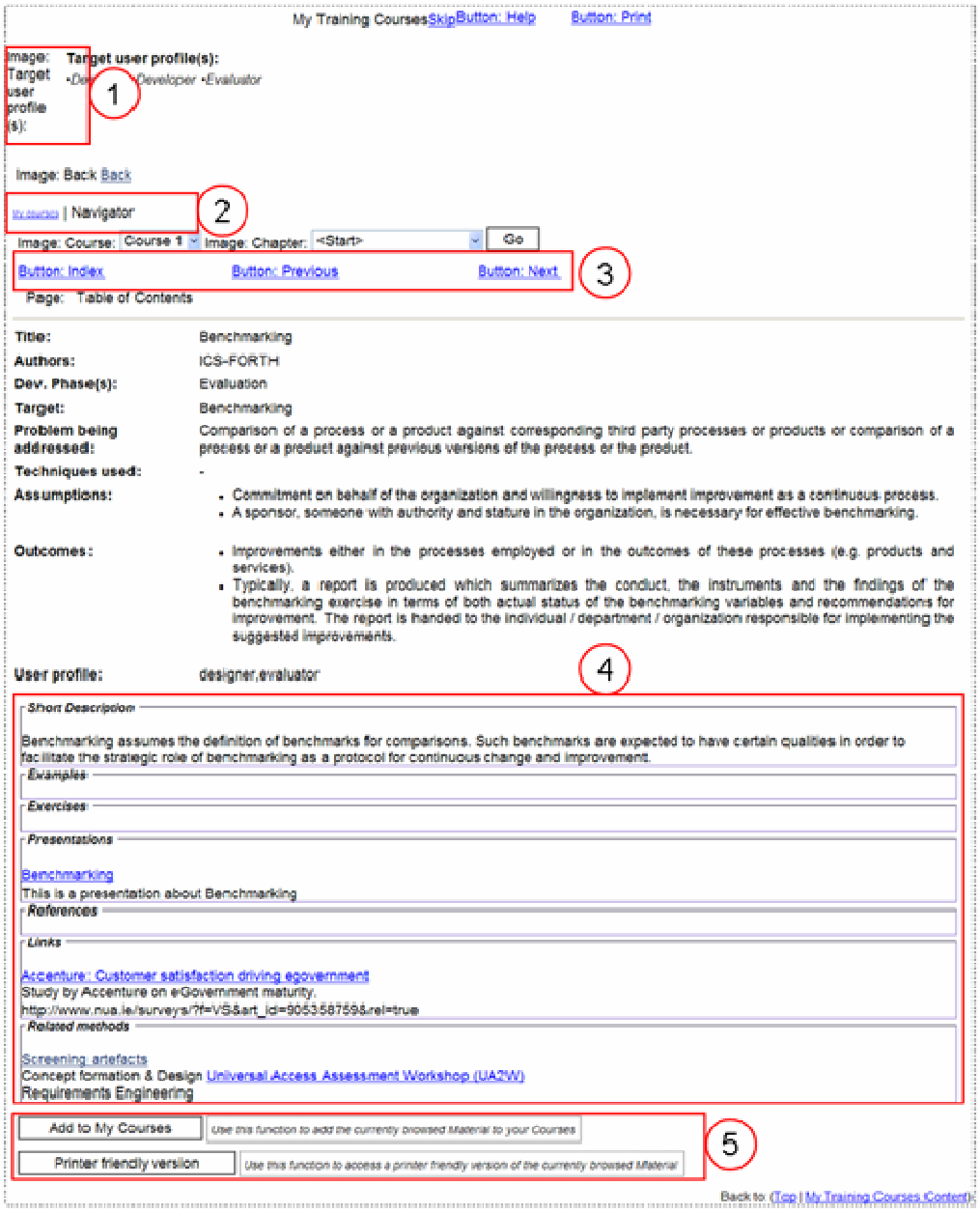


Figure 141: **Adaptation examples for the EDeAN online Training Course Navigator**

Digital Library (Ariadne Resource Centre)

The Ariadne Resource Centre module is an online structured repository of a wide variety of resource types. Its functionality is mainly based on searching and browsing mechanisms. Keywords, type or thematic areas can be the search criteria for resources.

More specifically, the facilities offered by the Ariadne Resource Centre include:

- **Browse resources by** (see Figure 142)
  - **Resource type:** Browse resources by the specified basic categories
  - **Resource topic:** Browse resources by the specified Resource Topic
  - **New resources:** Browse the newly added resources.
- **Search resources** (see Figure 143): Thought the search interface, users can perform a keyword-based search on the documents contained in the digital library. Two different search variations are provided, and namely:
  - **Simple search:** Simple search enables users to quickly search resources based only on a search query and additional search variations that affect the strictness that the search string will be used.
  - **Advanced search:** Advanced search enables users to select among additional options, such as the resource types to be contained in the results, the rating of the resources to be retrieved and the publication date.
- **Access information about a Resource** (see Figure 144): Specific information available for a resource. The general attributes that are displayed for all resources are: Title, Resource Type, user rating and popularity.

Figure 142: **Ariadne Browsing Facilities (default view)**

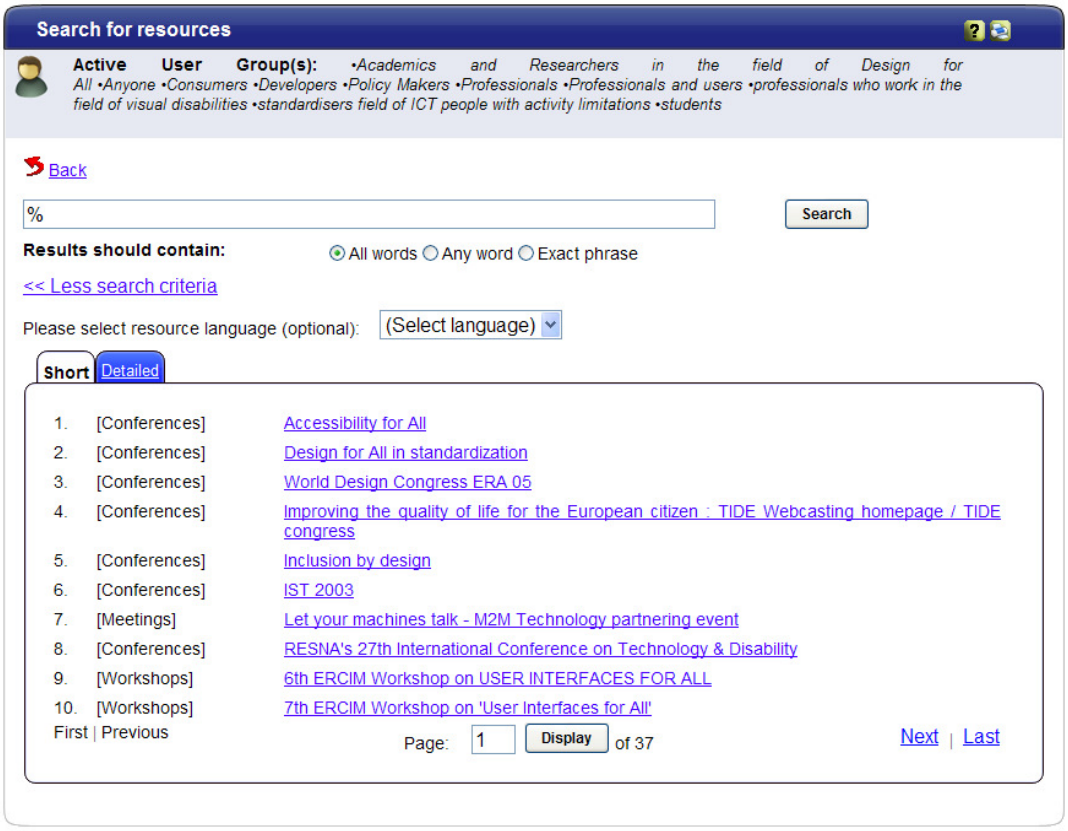


Figure 143: **Ariadne: Search Resources**

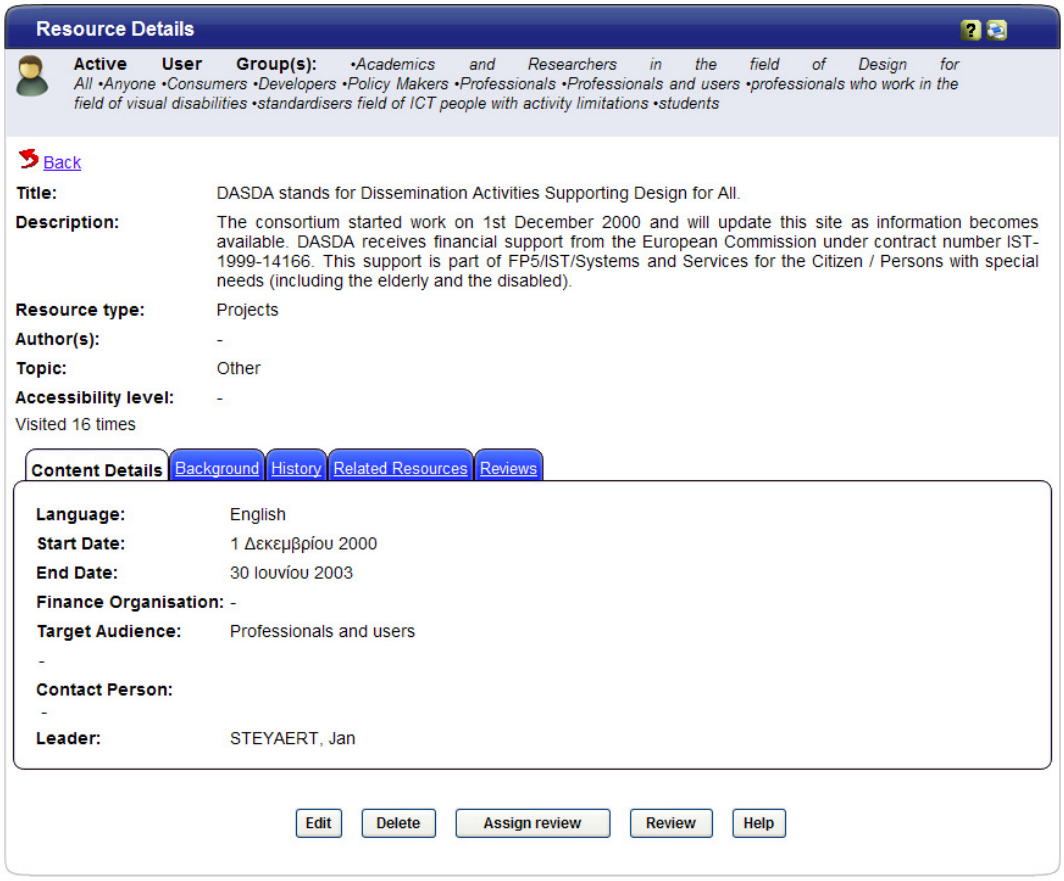


Figure 144: **Ariadne: Resource Details (default view)**

Following the quick presentation of the Ariadne resource centre and the facilities offered, some examples of the way that the interfaces offered can be adapted using the EAGER framework are provided. Figure 145 provides an overview of the Ariadne Home Page and some of the possible variations according to specific user parameters.

In this figure, the following adaptations are highlighted:

1. Page content is presented using extra large font size and Book Antiqua font family
2. Links are displayed as buttons and due to the selected colour setting with yellow colour for background and black for foreground.

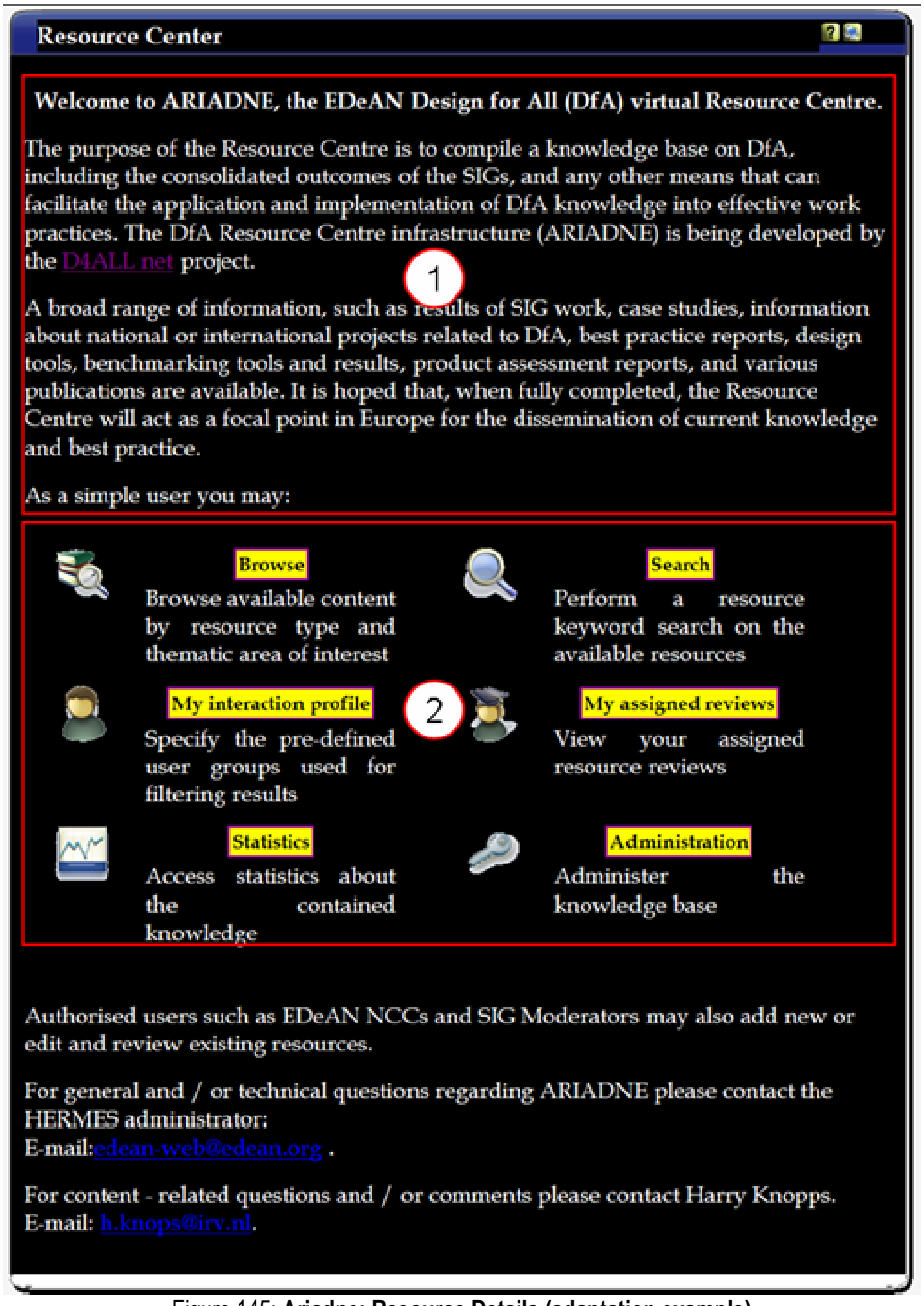


Figure 145: **Ariadne: Resource Details (adaptation example)**

7.2.3 Network Administrator's Interface

The administration part of an interactive application and especially of a web portal is usually invisible to the target user population. Although invisible, its importance for the seamless coordination of activities and the everyday running of the application is great. The administrator should be provided with all the facilities needed for enabling the easy and quick integration of changes in the portal. The Hermes portal incorporates a fully functional administration interface. An example of the EDeAN Content Administration interface is presented in Figure 146.

Figure 146: **Home page of Network Administrators (various scins)**

As shown in this figure, the EDeAN Content Administrators have access to the same personalisation facilities in terms of their setting, preferences and custom accessibility with the portal subscribed users. In terms of administrative functionality, Content Administrators have access to tools for administrating:

- **Resource centre:** Administration tool for the EDeAN Ariadne Resource Centre
- **Content management:** Administration tool for editing portal content
- **Portal users:** User administration facilities
- **Training courses:** EDeAN Training Course administration
- **News:** Facilities for inserting, editing and archiving News
- **Profile statistics:** A user profile statistics facility
- **Profile administration:** A user profile adnministration facility
- **Links:** Facilities for inserting, editing and archiving Links
- **Technical Support:** Facilities for replying to questions submitted to the portal technical support group
- **Frequently Asked Questions:** Facilities for inserting, editing and archiving Frequently Asked Questions
- **News Letters:** Facilities for inserting, editing and archiving News Letters.

The following section focuses on presenting the basic administrative facilities related with the adaptation aspects of the portal such as the profile administration and profile statistics modules.

7.2.3.1 Profile Statistics

Using the profile statistics interface, Content Administrators can access statistics on three basic dimensions:

- Statistics regarding the total popularity of settings (see Figure 147)
- Statistics regarding the popularity of preferences (see Figure 148)
- Statistics regarding the popularity of custom accessibility settings (see Figure 149).

For each of the aforementioned categories of statistics, Content Administrator can access a full report containing all statistics for a specific category or request to view the popularity graph for a specific user setting of a category.

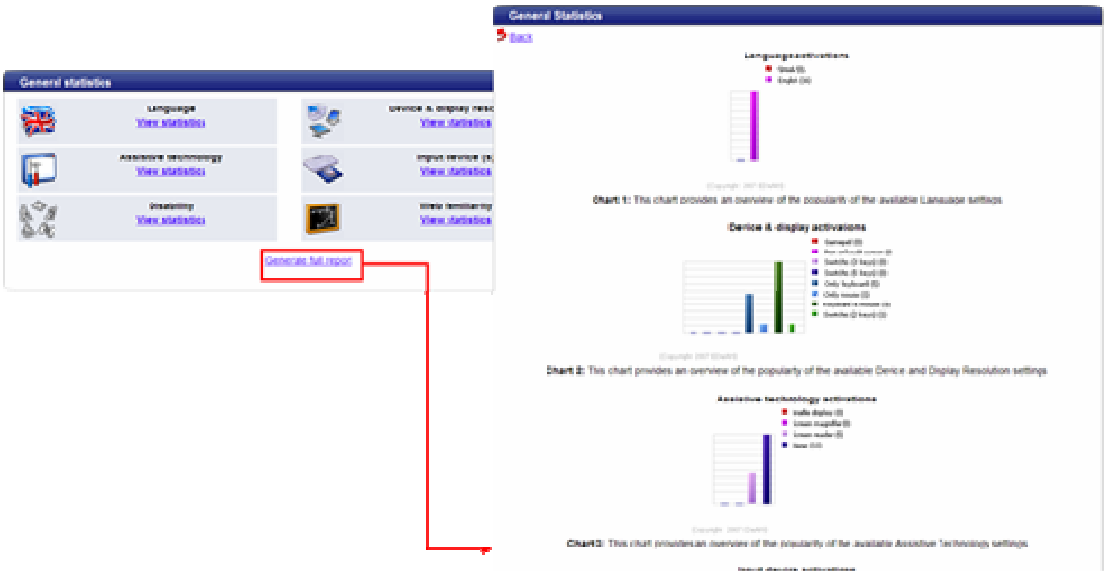


Figure 147: **Statistics regarding the popularity of settings**

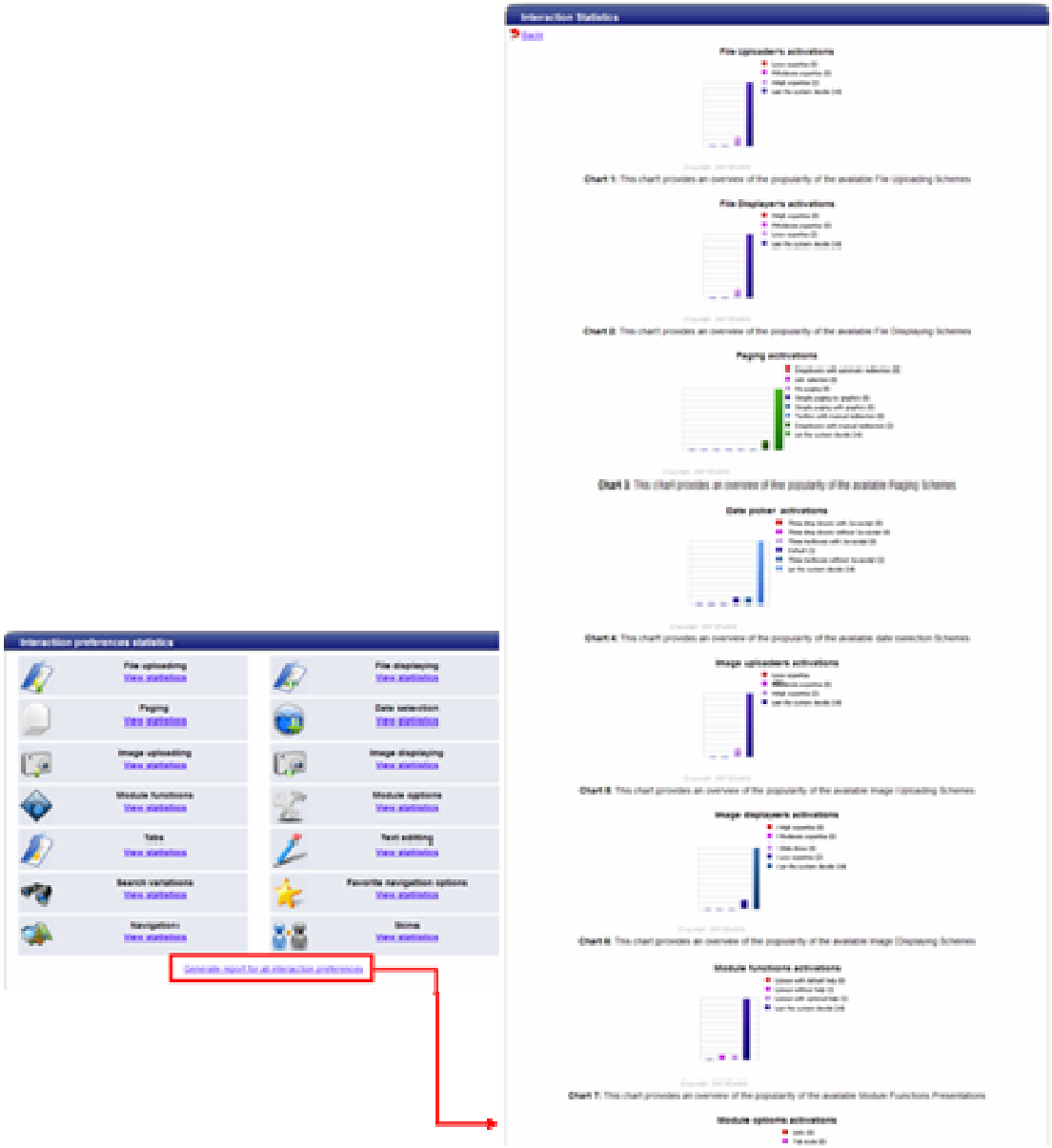


Figure 148: **Statistics regarding the popularity of preferences**

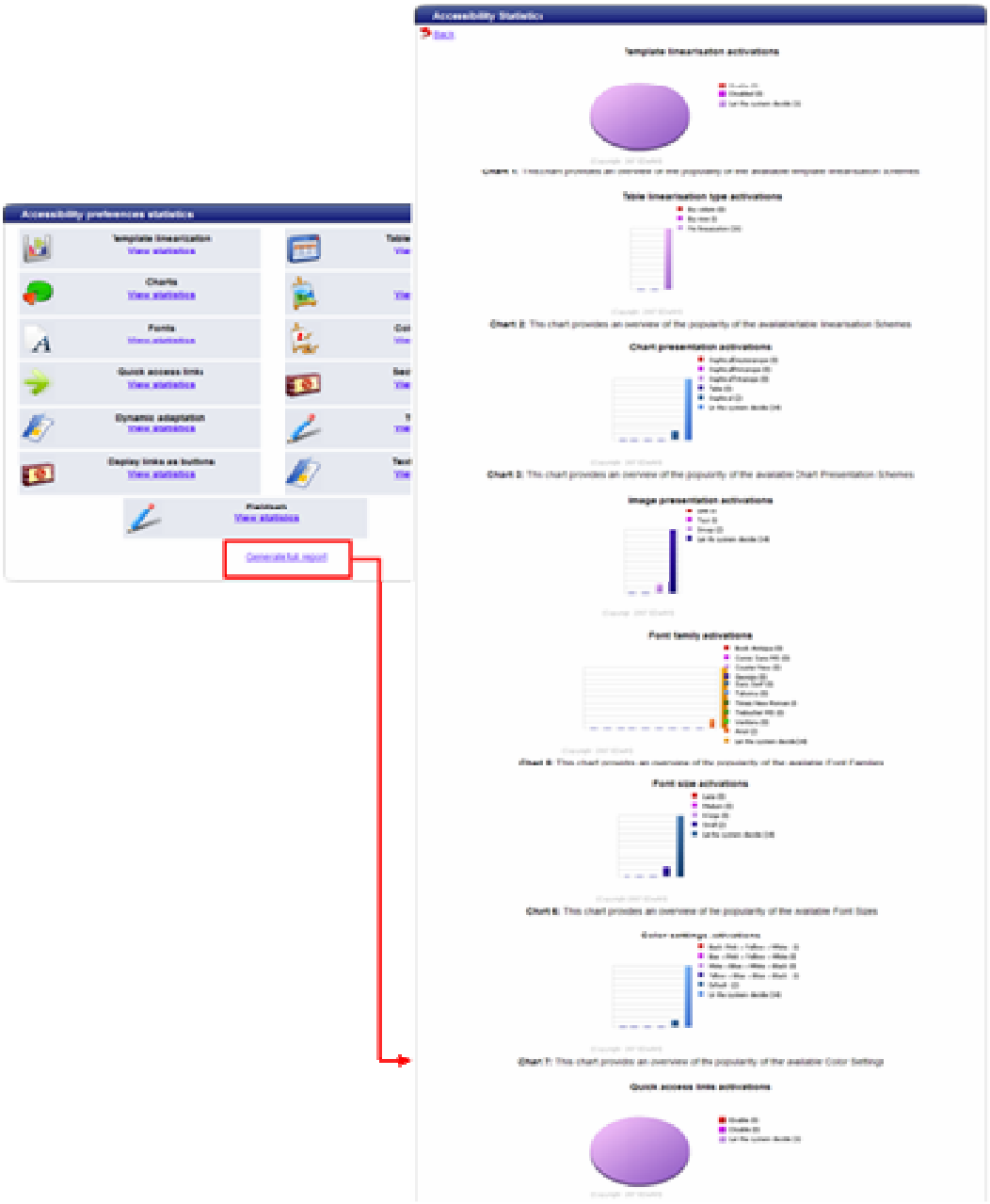


Figure 149: **Statistics regarding the popularity of custom accessibility settings**

7.2.3.2 Profile Administration

Using the Profile Administration facility, Content Administrators can insert, edit, and delete the predefined profiles provided to all portal members. A specialised administrative interface is provided to Content Administrators, as presented in Figure 150.

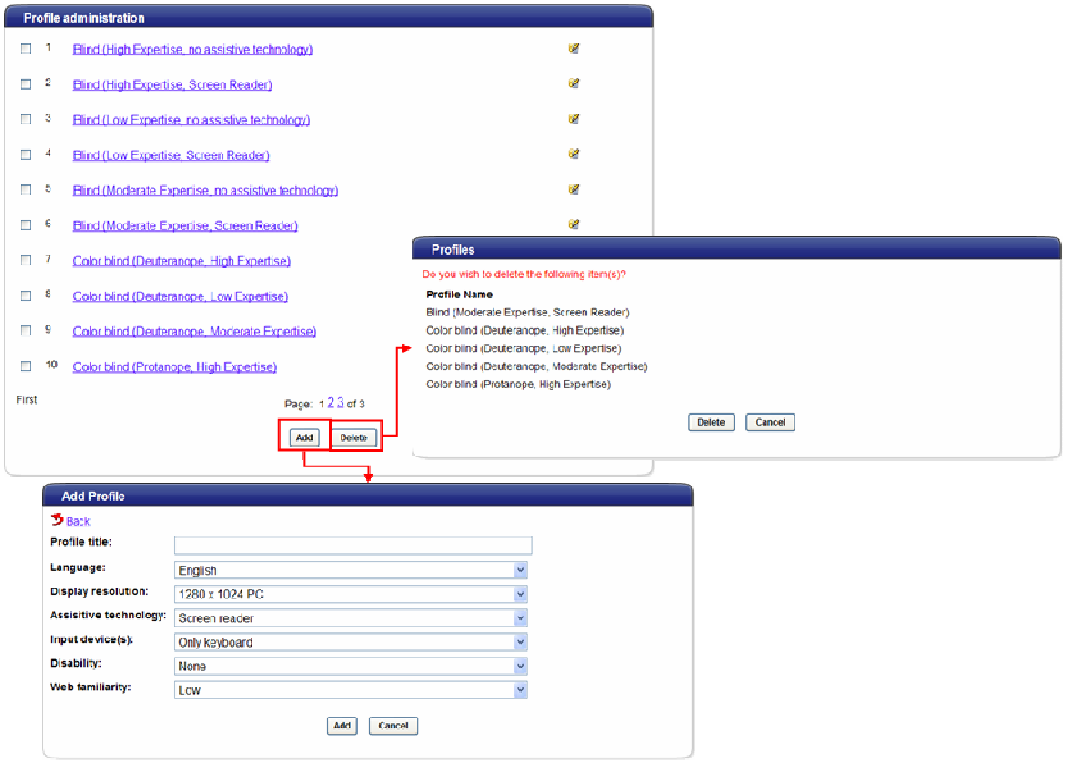


Figure 150: **The Profile Administration facility**

Chapter 8
EVALUATION - VALIDATION

In this section an overview of the methods selected for assessing the proposed UWI framework and its potential outcomes is described. Subsequently to the implementation phase of the UWI framework, an exhaustive evaluation was conducted focusing accessibility, as well as on user-experience in general of the EAGER generated outcomes (e.g., the prototype portal EDeAN). Accessibility issues were evaluated using the methodology that it is presented in 8.1 according to W3C. The user-experience of the EDeAN prototype was evaluated using an innovative inspection tool (ORIENT) presented in detail in subsection 8.2.

8.1 Accessibility evaluation of UI elements

8.1.1 Procedure

In order to evaluate all the outcomes of EAGER, the following decisions were taken after brainstorming with experienced accessibility evaluators:

- All the autonomous user interface elements have to be evaluated in all the alternative styles externally from a portal using at least one evaluation accessibility tool checking for conformance with W3C Web Content Accessibility Guidelines (WCAG) and U.S. Section 508. Evaluating these elements in a portal's environment is particularly difficult due to the wide range of alternative UI elements combinations.
- All the user interface elements that are related to colour have to be evaluated in order to assess colour effectiveness.

Taking into account the above decisions, an exhaustive evaluation of all the UI elements was carried out using the Watchfire Bobby[33] tool that checks conformance with W3C Web Content Accessibility Guidelines (WCAG) and U.S. Section 508 in parallel. A report with the errors and warnings found was produced. For the errors that were identified, it was initially explored if it was possible to correct them. The errors that were correctable were corrected at once. For the rest of the errors, the accessibility level that the specific controls conform was noted. For the warning that were identified the same procedure as for errors was carried out.

For the user interface elements that are related to colours, W3C guidelines were followed. To conform to Web Content Accessibility Guidelines, foreground and background colour combinations should provide sufficient contrast when viewed by someone with low vision or colour blindness, or when viewed on a black and white screen. The formula suggested by the World Wide Web Consortium (W3C) to determine the brightness of a colour is:

((Red value X 299) + (Green value X 587) + (Blue value X 114)) / 1000

Two colours provide good colour visibility if the brightness difference is greater than 125 and the colour difference is greater than 500.

[33] Watchfire Bobby Version 5.10.2.3

There are several accessibility tools that support contrast checking, but they examine only the difference between foreground and background colours for text elements without taking into account the absolute position of the html elements, and it is possible to have visually a different background on the site compared with the one from tool. Additionally, dynamic changes and mouse over effects (such as hover) are ignored by the tool. Having in mind the above, in order to evaluate different colour combinations, the algorithm presented below was applied to background – foreground colours for the UI elements that offer colour variety. For the UI elements that provide image output, the online tool *Vischeck*[34] was used to transform output images to images that people with colour-blindness see in order to evaluate them.

8.1.2 Results (Conformance to guidelines)

Initially the target conformance level of WCAG 1.0 was decided to be of Level "Triple-A" satisfying all Priority 1, 2, and 3 checkpoints in order to ensure maximum access to Web documents. The detailed procedure of the performed evaluation was recorded and is presented in the following sections.

For all the autonomous UI elements, all the alternative user interfaces for each element were produced on a blank html page that was evaluated using the desktop accessibility evaluation tool Watchfire Bobby. Errors for all priority 1, 2, and 3 checkpoints were identified, recorded and solved when possible. Warnings for all the priorities were also collected and checked manually. For these that were actually confirmed, the solution that was applied was recorded. On the other hand, for the errors that where not fully recovered (e.g., UI elements which require Java Script), we ensured that their alternatives had no accessibility errors. The "erroneous" designs, however, where not removed from the designs base, as they can still meet the preferences of some users who are not really concerned of accessibility. This evaluation results are summarised in the following tables.

File uploader – Direct / mixed manipulation			
Errors			
Priority	Guideline	Action needed	Solution
2	12.4	Explicitly associate form controls and their labels with the LABEL element.	For the labels "File" and "File title:" a "for" attribute was used containing the "id" attribute of the assigned textbox.
3	10.4	Include default, place holding characters in edit boxes and text areas.	The default text value "Please enter the file title" was placed to the textbox for file title.
Warnings			
Priority	Guideline	Description	Solution
n/a	n/a	n/a	-
Achieved WAI conformance level after addressing the problem(s): **AAA**			

[34] http://www.vischeck.com/vischeck/vischeckImage.php

File uploader - Indirect manipulation			
Errors			
Priority	Guideline	Action needed	Solution
1	6.2	Each FRAME must reference an HTML file.	The frame of the file uploader progress bar references an aspx page equivalent to an html page.
1	12.1	Give each frame a title.	Title "File uploader progress bar" title is used as a name for the frame.
2	12.4	Explicitly associate form controls and their labels with the LABEL element.	For the labels "File" and "File title:" a "for" attribute was used containing the "id" attribute of the assigned textbox.
3	10.4	Include default, place-holding characters in edit boxes and text areas.	The default text value "Please enter the file title" was placed to the textbox for file title.
Warnings			
Priority	Guideline	Description	Solution
1	8.1	Provide accessible alternatives to the information in scripts, applets, or objects.	For each one of the script elements contained in the file uploader a nonscript element was placed to the source code, describing the script's functionality.
2	12.2	Add a description to a frame if the TITLE does not describe its contents.	The frame title describes the frame contents
Achieved WAI conformance level after addressing the problem(s): **AAA**			

File Displayer – High expertise, Moderate expertise, Low expertise			
Errors			
Priority	Guideline	Action needed	Solution
n/a	n/a	n/a	n/a
Warnings			
Priority	Guideline	Action needed	Solution
2	3.2	Make sure your document validates to formal published grammars.	The character < was replaced with the html code < that validates successfully to formal published grammars.
Achieved WAI conformance level after addressing the problem(s): **AAA**			

Paging – No paging, Links paging, Dropdowns paging with manual direction, TextBox paging, Simple paging with graphics, Simple paging no graphics
Errors: No errors
Warnings: No warnings
Achieved WAI conformance level after addressing the problem(s): **AAA**

Paging – Dropdowns paging with automatic direction			
Errors			
Priority	Guideline	Action needed	Solution
n/a	n/a	n/a	n/a
Warnings			
Priority	Guideline	Action needed	Solution
1	8.1	Provide accessible alternatives to the information in scripts, applets, or objects.	For the script element that fires a postback event a nonscript element was placed to the source code, describing the script's functionality
1	6.3	Make sure pages are still usable if programmatic objects do not function.	This warning is generated due to the existence of javascript for automatic redirection. To overcome this issue when maximum accessibility is required its alternative representation with manual redirection is provided.
Achieved WAI conformance level after addressing the problem(s): (no conformance)			

Date picker – Graphical			
Errors			
Priority	Guideline	Action needed	Solution
1	12.4	Provide alternative text for all image-type buttons in forms.	Alternative text was added to the image button used for opening the graphical date picker.
2	12.4	Explicitly associate form controls and their labels with the LABEL element.	For the labels "Date", "Month" and "Year", "for" attributes were used containing the "id" attribute of the corresponding textbox or dropdown.
3	5.5	Provide a summary for tables.	The graphical calendar control is used as it is from the ASP.NET framework and cannot be altered to contain a table summary.
Warnings			
Priority	Guideline	Action needed	Solution
1	5.1	If this is a data table (not used for	The graphical calendar control is

		layout only), identify headers for the table rows and columns.	used as it is from the ASP.NET framework and cannot be altered to introduce identification for headers.
Achieved WAI conformance level after addressing the problem(s): **AA**			

Date picker – Dropdown automatic redirection			
Errors			
Priority	Guideline	Action needed	Solution
2	12.4	Explicitly associate form controls and their labels with the LABEL element.	For the labels “Day”, “Month” and “Year”, "for" attributes were added containing the “id" attribute of the corresponding dropdown control.
Warnings			
Priority	Guideline	Action needed	Solution
n/a	n/a	n/a	n/a
Achieved WAI conformance level after addressing the problem(s): **AAA**			

Date picker – Dropdown manual redirection			
Errors			
Priority	Guideline	Action needed	Solution
2	12.4	Explicitly associate form controls and their labels with the LABEL element.	For the labels “Day”, “Month” and “Year”, "for" attributes were added containing the “id" attribute of the corresponding dropdown control.
Warnings			
Priority	Guideline	Action needed	Solution
2	9.2	Make sure that all elements that have their own interface are operable without a mouse.	All elements having their own interface are operable without a mouse.
Achieved WAI conformance level after addressing the problem(s): **AAA**			

Date picker – TextBox automatic redirection, TextBox manual redirection			
Errors			
Priority	Guideline	Action needed	Solution
2	12.4	Explicitly associate form controls and their labels with the LABEL element.	For the labels “Day”, “Month” and “Year”, "for" attributes were added containing the “id" attribute of the

			corresponding dropdown control.
Warnings			
Priority	Guideline	Action needed	Solution
2	6.4	If objects use event handlers, make sure they do not require use of a mouse.	All elements having their own interface are operable without a mouse.
Achieved WAI conformance level after addressing the problem(s): **AAA**			

Image uploader - Direct manipulation, Mixed mode manipulation			
Errors			
Priority	Guideline	Action needed	Solution
2	12.4	Explicitly associate form controls and their labels with the LABEL element.	For the labels “Image” and “Image title:” a "for" attribute was used containing the “id" attribute of the assigned textbox.
3	10.4	Include default, place holding characters in edit boxes and text areas.	The default text value “Please enter the image title” was placed to the textbox for file title.
Warnings			
Priority	Guideline	Action needed	Solution
n/a	n/a	n/a	n/a
Achieved WAI conformance level after addressing the problem(s): **AAA**			

Image uploader - Indirect manipulation			
Errors			
Priority	Guideline	Action needed	Solution
1	6.2	Each FRAME must reference an HTML file.	The frame of the image uploader progress bar references an aspx page equivalent to an html page.
1	12.1	Give each frame a title.	Title “Image uploader progress bar” title is used as a name for the frame.
2	12.4	Explicitly associate form controls and their labels with the LABEL element.	For the labels “Image” and “Image title:” a "for" attribute was used containing the “id" attribute of the assigned textbox.
3	10.4	Include default, place-holding characters in edit boxes and text areas.	The default text value “Please enter the image title” was placed to the textbox for file title.
Warnings			

Priority	Guideline	Action needed	Solution
1	8.1	Provide accessible alternatives to the information in scripts, applets, or objects.	For each one of the script elements contained in the image uploader a nonscript element was placed to the source code, describing the script's functionality.
2	12.2	Add a description to a frame if the TITLE does not describe its contents.	The frame title describes the frame contents
Achieved WAI conformance level after addressing the problem(s): **AAA**			

Image Displayer – High expertise, Moderate expertise, Low expertise			
Errors			
Priority	Guideline	Action needed	Solution
n/a	n/a	n/a	n/a
Warnings			
Priority	Guideline	Action needed	Solution
2	3.2	Make sure your document validates to formal published grammars.	The character < was replaced with the html code &#60.
Achieved WAI conformance level after addressing the problem(s): **AAA**			

Module functions – Without help, with optional help, with default help
Errors: No errors
Warnings: No warnings
Achieved WAI conformance level after addressing the problem(s): **AAA**

Module options – All graphical options, tabbed options, Linear options
Errors: No errors
Warnings: No warnings
Achieved WAI conformance level after addressing the problem(s): **AAA**

Tabs – Graphical tabs, tab formed as option, Linear tab (WAI AAA conformance)
Errors: No errors
Warnings: No warnings
Achieved WAI conformance level after addressing the problem(s): **AAA**

Text editing– Full functional, moderated functionality, limited functionality
Errors

Priority	Guideline	Action needed	Solution
1	1.1	Provide alternative text for all images.	The graphical editor controls build on a freely available third party control with no source redistribution license. No alternative text can be placed on images.
1	12.1	Give each frame a title.	The graphical editor controls build on a freely available third party control with no source redistribution license. No title can be assigned to frames.
2	12.4	Explicitly associate form controls and their labels with the LABEL element.	There aren't labels that are associated to form controls.
3	5.5	Provide a summary for tables.	The graphical editor controls build on a freely available third party control with no source redistribution license. No summary for internal tables can be added.
3	10.4	Include default, place-holding characters in edit boxes and text areas.	The default text value "Please enter text" was placed to the text editor.
Warnings			
Priority	Guideline	Action needed	Solution
n/a	n/a	n/a	n/a
Achieved WAI conformance level after addressing the problem(s): **no conformance**			

Text editing– Non graphical			
Errors			
Priority	Guideline	Action needed	Solution
2	9.3	Make sure event handlers do not require use of a mouse.	Events can be activated via keyboard
2	13.1	Create link phrases that make sense when read out of context.	Meaningful link phrases added
3	10.5	Separate adjacent links with more than whitespace.	A not visible 'l' was added between adjacent links
Warnings			
Priority	Guideline	Action needed	Solution
n/a	n/a	n/a	n/a
Achieved WAI conformance level after addressing the problem(s): **AAA**			

Search– Simple, optional advanced options, advanced options			
Errors			
Priority	Guideline	Action needed	Solution
2	12.4	Explicitly associate form controls and their labels with the LABEL element.	For the label “Results should contain” "for" attributes was added containing the “id" attribute of the corresponding radio buttons.
3	10.4	Include default, place-holding characters in edit boxes and text areas.	The default text value “Enter search keywords” was placed to the text field.
Warnings			
Priority	Guideline	Action needed	Solution
n/a	n/a	n/a	n/a
Achieved WAI conformance level after addressing the problem(s): **AAA**			

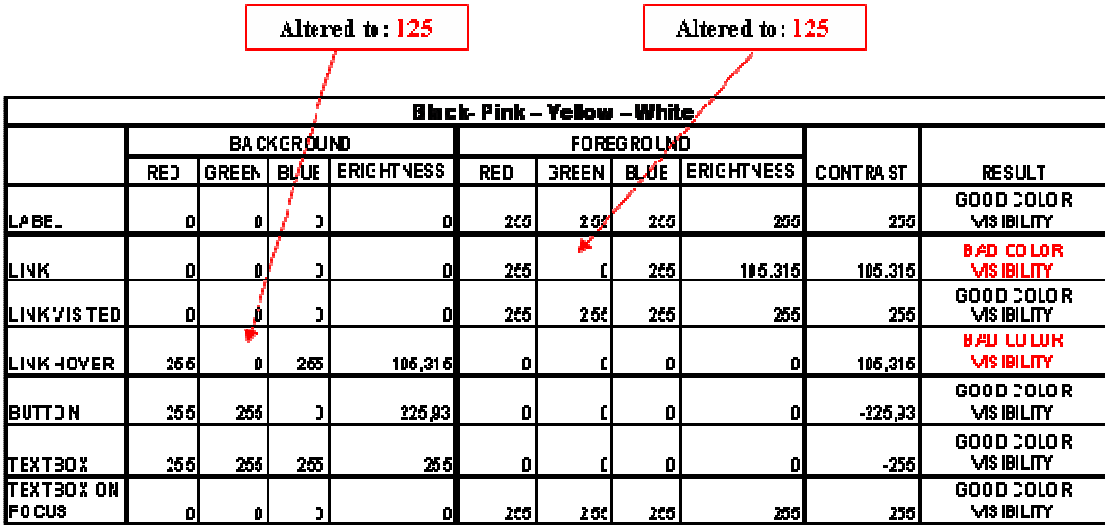
Tables – All table settings
Errors: No errors
Warnings: No warnings
Achieved WAI conformance level after addressing the problem(s): **AAA**

Images – image, link, label
Errors: No errors
Warnings: No warnings
Achieved WAI conformance level after addressing the problem(s): **AAA**

8.1.3 Results (colour effectiveness)

In the following tables the algorithm described in section 8.1 was applied to the different colour combinations of the proposed designs. Each table includes the background and foreground RGB colours (Red, Green and Blue) for each html element: label, link, visited link, link on hover, button, textbox and textbox on focus. Finally, for each html element, the background and foreground brightness are calculated in terms of the contrast between them. The last column specifies if the given colour combinations provide good colour visibility.As it is shown in the following tables, some colour visibility failures were identified and corrected. More specifically, the (255, 0, 255) RGB colour *(Fuscia)* induced low colour visibility, which was address by using the (255, 125, 255) RGB colour *(light pink)*.

Table 27: **Black-Pink-Yellow-White colour setting evaluation**

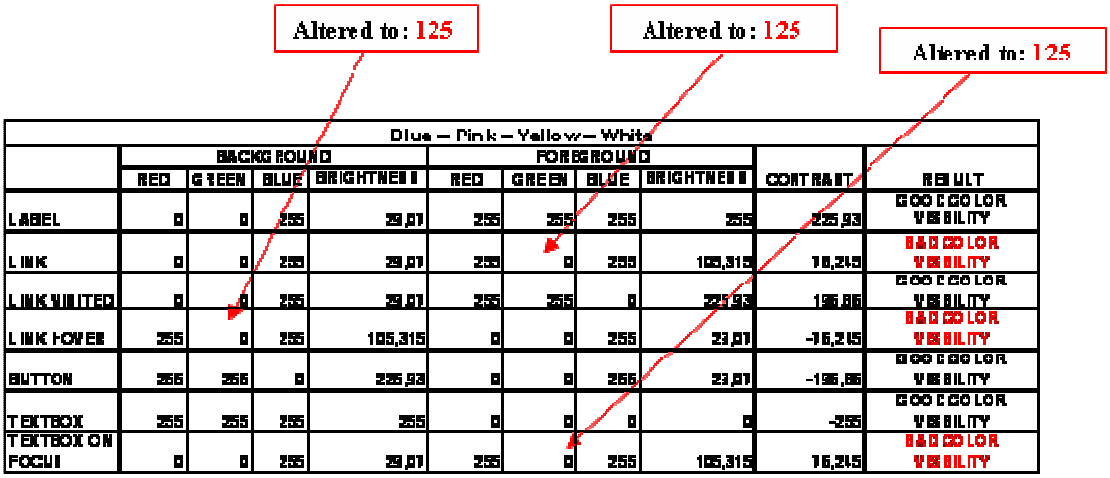


Black- Pink – Yellow – White										
	BACKGROUND				FOREGROUND					
	RED	GREEN	BLUE	BRIGHTNESS	RED	GREEN	BLUE	BRIGHTNESS	CONTRAST	RESULT
LABEL	0	0	0	0	255	255	255	255	255	GOOD COLOR VISIBILITY
LINK	0	0	0	0	255	0	255	105,315	105,315	BAD COLOR VISIBILITY
LINK VISITED	0	0	0	0	255	255	255	255	255	GOOD COLOR VISIBILITY
LINK HOVER	255	0	255	105,315	0	0	0	0	105,315	BAD COLOR VISIBILITY
BUTTON	255	255	0	225,93	0	0	0	0	-225,93	GOOD COLOR VISIBILITY
TEXTBOX	255	255	255	255	0	0	0	0	-255	GOOD COLOR VISIBILITY
TEXTBOX ON FOCUS	0	0	0	0	255	255	255	255	255	GOOD COLOR VISIBILITY

Table 28: **Yellow-Blue-Blue-Black colour setting evaluation**

Yellow – Blue – Blue – Black										
	BACKGROUND				FOREGROUND					
	RED	GREEN	BLUE	BRIGHTNESS	RED	GREEN	BLUE	BRIGHTNESS	CONTRAST	RESULT
LABEL	255	255	0	225,93	0	0	0	0	-225,93	GOOD COLOR VISIBILITY
LINK	255	255	0	225,93	0	0	255	29,07	-196,86	GOOD COLOR VISIBILITY
LINK VISITED	255	255	0	225,93	0	0	0	0	-225,93	GOOD COLOR VISIBILITY
LINK HOVER	0	0	255	29,07	255	255	0	225,93	196,86	GOOD COLOR VISIBILITY
BUTTON	0	0	255	29,07	255	255	255	255	225,93	GOOD COLOR VISIBILITY
TEXTBOX	255	255	255	255	0	0	0	0	-255	GOOD COLOR VISIBILITY
TEXTBOX ON FOCUS	255	255	0	225,93	0	0	0	0	-225,93	GOOD COLOR VISIBILITY

Table 29: **Blue-Pink-Yellow-White colour setting evaluation**



Blue – Pink – Yellow – White										
	BACKGROUND				FOREGROUND					
	RED	GREEN	BLUE	BRIGHTNESS	RED	GREEN	BLUE	BRIGHTNESS	CONTRAST	RESULT
LABEL	0	0	255	29,07	255	255	255	255	225,93	GOOD COLOR VISIBILITY
LINK	0	0	255	29,07	255	0	255	105,315	76,245	BAD COLOR VISIBILITY
LINK VISITED	0	0	255	29,07	255	255	0	225,93	196,86	GOOD COLOR VISIBILITY
LINK HOVER	255	0	255	105,315	0	0	255	29,07	-76,245	BAD COLOR VISIBILITY
BUTTON	255	255	0	225,93	0	0	255	29,07	-196,86	GOOD COLOR VISIBILITY
TEXTBOX	255	255	255	255	0	0	0	0	-255	GOOD COLOR VISIBILITY
TEXTBOX ON FOCUS	0	0	255	29,07	255	0	255	105,315	76,245	BAD COLOR VISIBILITY

Table 30: **White-Blue-White-Black colour setting evaluation**

White – Blue – White – Black										
	BACKGROUND				FOREGROUND					
	RED	GREEN	BLUE	BRIGHTNESS	RED	GREEN	BLUE	BRIGHTNESS	CONTRAST	RESULT
LABEL	255	255	255	255	0	0	0	0	-255	GOOD COLOR VISIBILITY
LINK	255	255	255	255	0	0	255	29,07	-225,93	GOOD COLOR VISIBILITY
LINK VISITED	255	255	255	255	49	71	121	70,122	-134,878	GOOD COLOR VISIBILITY
LINK HOVER	0	0	255	29,07	255	255	255	255	225,93	GOOD COLOR VISIBILITY
BUTTON	255	255	255	255	0	0	0	0	-255	GOOD COLOR VISIBILITY
TEXTBOX	255	255	255	255	0	0	0	0	-255	GOOD COLOR VISIBILITY
TEXTBOX ON FOCUS	0	0	0	0	255	255	255	255	255	GOOD COLOR VISIBILITY

As it was described in the previous sections, an exhaustive accessibility evaluation procedure was applied to the autonomous user interface outcomes of the framework as well as to a great range of pages for different predefined profiles of the portal to ensure maximum accessibility. The errors that were identified during the evaluation procedure were solved and the specific solutions were recorded analytically. At last, in order to identify if the described portal is accessible for blind, low vision and motor-impaired users, corresponding assistive technologies were used through the portal and proved that it offers maximum accessibility for people with disability.

8.2 User-experience evaluation

8.2.1 Procedure

In order to evaluate the EDeAN portal from the perspective of user-experience, and thereby validate EAGER, a special inspection tool named *Orient* [54], and the underlying methodology [55] for assessing online services were used. More specifically, the aforementioned evaluation methodology, using the online tool, was applied to the fully functional prototype portal that is described in detail in Chapter 7 (EDeAN). The aforementioned methodology is described in brief in here, along with the steps that were followed during the evaluation along with the inspection.

Orient facilitates and guides electronic services assessment in terms of the degree to which their users' needs and requirements are met through out the user experience lifecycle, and thereby predicts their "*adoptability*", as the degree of likelihood to be adopted from target users [56]. The Orient tool was chosen as it provides a simple-to-use, quick and efficient way to simulate a user's reasoned action process and derive conclusions about the overall user-perceived quality of a system. The Orient tool implements a behavioural walkthrough evaluation [55], which, unlike traditional cognitive walkthroughs that check a user's ability to perform a given task, simulates a user's reasoned action process at each step in the human-computer dialogue, checking to see if the simulated user's beliefs, external stimuli (such as system design features) and intentions to perform can be assumed as a prerequisite to lead to the next interaction step. Reference points for a system's evaluation are its visibility, perceived usefulness and ease of use, availability/approachability, quality of interaction experience, and relationship maintainability. Thus, due to the nature of the method, the tool is intended to assist experts in the fields of usability, accessibility, software design and others in conducting evaluations in a Universal Access perspective.

The Orient inspection process is conducted in four phases by a multidisciplinary team of inspectors. The tool is composed of open-ended questionnaire forms that help individual inspectors make their user-orientation comments and assign positive or negative scores to these. The first phase is the preparation phase, where the objectives and limitations of the inspection are explored and the composition of the inspection team, both in terms of number and expertise, is envisaged. During the set-up phase, the team of inspectors identifies user groups for the given system, based on user goals, and lists the system functions. Also at this phase the conditions of use for each user group are laid down. The core phase of the evaluation follows, where each inspector follows a step-by-step procedure to assess the user-orientation of the system as a whole, and to investigate its individual functions. Finally, during the reporting phase, the inspection leader collects and summarises the results from individual forms, and if required, synthesises the final report, which contains both quantitative data (scores) and qualitative data (expert comments and proposals for improvement). A detailed example evaluation by means of the ORIENT tool is reported in [57].

8.2.1.1 System description

Access information:	http://hci-web.ics.forth.gr/Edean
Provider:	HCI laboratory, ICS Forth Institute.
Developer:	Constantina Doulgeraki
Platform:	Web-based
Current lifecycle stage:	Pilot use version
Objectives:	The main objectives with the establishment of EDeAN were a) to provide input for European Curricula in Design for All, b) to create a forum for Design for All issues, c) to promote the sharing of ideas through joint activities such as conferences, symposia and exchanges of students and scholars. EDeAN was also targeted towards fostering awareness and promoting changes of culture in the public and private sectors, as well as establishing links with appropriate education channels to embed Design for All best practices in new curricula.
Target user population:	The goal of the EDeAN portal to support networking among DFA experts
Application field(s):	
Navigation styles:	Web-like
Networking and communications supported:	
Supported communication media:	Emails, Chat, Asynchronous communication message boards
Resources of the system, in terms of hardware, software and personnel at the provider site:	A web-based, stand-alone application

8.2.1.2 Inspection team

Inspection leader: **P. M.**	
Native language:	Greek
Sex:	Male
Familiarity with Orient:	Excellent (Developer of ORIENT)
Expertise:	Senior
English fluency:	Excellent
Background:	CSD postgraduate student

Inspector: **P. P.**	
Native language:	Greek
Sex:	Male
Familiarity with Orient:	Low
Expertise:	Advanced
English fluency:	High
Background:	CSD postgraduate student

Inspector: **V. B.**	
Native language:	Greek
Sex:	Male
Familiarity with Orient:	Moderate
Expertise:	Advanced
English fluency:	High
Background:	CSD postgraduate student

Inspector: **A. D.**	
Native language:	Greek
Sex:	Male
Familiarity with Orient:	Low
Expertise:	Advanced
English fluency:	High
Background:	CSD postgraduate student

8.2.1.3 Target user groups

User Groups	
1. Blind users	Users that face vision problems and have not assistive technology
2. Colour-blind users	Users that face colour - vision problems
3. Motor-impaired users	Users that have motor-impairments and can not use keyboards

8.2.1.4 Functions per user group

All following functions, which are relevant to all user groups, were inspected:

Function No 1: Change Profile settings

- Change Basic settings → Web familiarity
- Change Custom Accessibility → Font size
- Change preferences →Font family

Function No 2: Post message (with attachment)

- Select Discussion Topics
- Create new topic
- Fill in form (Attach file…)

Function No 3: View resource in Resource Center

- Browse
- By thematic area
- Select type
- View resource details

8.2.2 Inspection results

User group 1: Blind users
Function 1: Change Profile settings
Visibility
⇑ The sitemap increases the visibility of the function. ⇓ At home page, settings would be more visible, if the link was placed before Help and FAQ
Usefulness
⇑ Predefined profiles are very helpful. ⇑ Adjusting accessibility features and preferences is a very useful function for disabled users. ⇓ The SALT plug-in fails to read the current location in the phrase "Image: You are at: location". ⇓ When I switched to blind users, at the main menu it was written "Image: You are at: Settings" but the screen reader could only read "Image: You are at:"
Availability – approachability
⇑ Menu links are read correctly by the SALT plug-in enabling the user to reach the function. ⇓ Some options do not enhance the interaction of blind users (e.g. "Change font family", "Change font size") and can be eliminated.
Quality of usage experience
⇑ The process is fast. ⇓ Consider omitting text based images that do not provide any additional information to the user.
Maintanability
⇑ Despite certain problems in navigation, the function is very important and users are expected to re-use it. Consider omitting text based images that do not provide any additional information to the user.

User group 1: Blind users
Function 2: Post a message in a discussion group
Visibility
⇑ Function is moderately visible from the main page (can be reached from the main menu in 3 steps).
Usefulness
⇑ "Cancel" of function and "Cancel" of file uploading may be confused. ⇓ The button of post message should be also at the start of the page.
Availability – approachability
⇑ Menu links are read correctly by the SALT plug-in enabling the user to reach the function. ⇓ Post message is at the bottom of the page ⇓ The blind user first listens to the resource, press Tab to listen the number of resources and then go back to select the resource. Could be together, 1 step
Quality of usage experience
⇓ The user is instructed to "fill in the fields with an asterisk". However, blind users never hear the indication of an asterisk. ⇓ Once the user "enters" the area of the editor, there is no way to "exit". ⇓ The "browse" button for attaching a file is not read by the SALT plug-in.
Maintanability
-

User group 1: Blind users
Function 3: View resource in Resource centre
Visibility
⇑ Function resides on the 2nd level of the main menu hierarchy (relatively visible).
Usefulness
-
⇓ Availability – approachability ⇓ Menu links are read correctly by the SALT plug-in enabling the user to reach the function. ⇓ The blind user first listens to the resource, press Tab to listen the number of resources and then go back to select the resource. Could be together, 1 step

Quality of usage experience
⇓ Number of resources for each thematic area is read separately. Hence it makes no sense.
⇓ Certain links are not read by the SALT plug-in.
⇓ Acronyms with adjacent initials are not read correctly by the SALT plug-in.
⇓ Status bar (You are at:...) skips reading the current location of the user.-
Maintanability
⇑ There are descriptive labels and proper grouping of resource details.
⇑ Users are likely to use the function again..

User group 2: Colour-Blind users
Function 1: change profile settings
Visibility
⇑ This function can be reached by 3 routes.
Usefulness
⇑ The help and FAQ sections do not contain information about the utility of the function neither on how to use this function
⇑ The SALT plug-in fails to read the current location in the phrase "Image: You are at: location".
Availability – approachability
-
Quality of usage experience
⇓ Star symbol for favourites is not very intuitive and not at all explanatory. Alt text is "*", instead of an informative text.
Maintanability
⇑ Function is efficient and straightforward.
⇑ If function used frequently, then it is in favourites

User group 2: Colour-Blind users
Function 2: Post a message in a discussion group
Visibility
⇑ It is quite easy to locate this function from the discussion groups from the home page.
Usefulness
⇓ Delete function is an step by step process. Consider replacing the text "Delete" with "Delete..." in delete button.
⇓ The provided text editor does not appear to have high contrast options and text input.
Availability – approachability
-
Quality of usage experience
⇓ Star symbol for favourites is not very intuitive and not at all explanatory. Alt text is "*", instead of an informative text.
Maintanability
⇑ Function is efficient and straightforward.
⇑ Users are likely to use the function again.-

User group 2: Colour-Blind users
Function 3: View resource in Resource centre
Visibility
⇑ This function can be reached through different shortcuts and from navigation menu.
⇑ It is visible, as it is the third option of the home page.
Usefulness
⇑ Information is presented in a well structured way.
⇓ Consider providing a simpler (visually) version of tabs, alternatively to graphical tabs.
Availability – approachability
-

Quality of usage experience
⇑ Star symbol for favourites is not very intuitive and not at all explanatory. Alt text is "*", instead of an informative text.
Maintanability
⇑ If function used frequently, then it is in favourites ⇑ This function uses descriptive labels.

User group 3: Motor-impaired users
Function 1: Change Profile settings
Visibility
⇑ . This function can be reached through different shortcuts and from navigation menu. ⇓ At home page, settings would be more visible, if the link was placed before Help and FAQ.
Usefulness
⇑ The function is clear and easy to use. There are descriptive names and icons for the links and the buttons. ⇑ Predefined profiles are very helpful. ⇑ Guidelines at the beginning of the "Settings" section help the user to quickly understand the purpose of the function. ⇓ The help and FAQ sections do not contain information about the utility of the function neither on how to use this function
Availability – approachability
⇓ Since the user has not signed in, the user can change the profile settings from the link "setting". After the user has signed in, the link is named "accessibility option", which confuses the user and thinks that this is something different.
Quality of usage experience
⇑ It is very good practice the use of links/ button that guide the user to go to the top/bottom/header of the page or skip navigation etc.
Maintanability
⇑ If function used frequently, then it is in favourites.

User group 3: Motor-impaired users
Function 2: Post a message in a discussion group
Visibility
⇑ This function can be reached fairly easily through the navigation menu at the left part of each web page. ⇓ Users need to make a number of steps to reach the function starting from the homepage. Hence the function is not very visible.
Usefulness
⇑ Delete function is a step by step process. Consider replacing the text "Delete" with "Delete...". ⇓ The button of post message should also be at the start of the page
Availability – approachability -
Quality of usage experience
⇓ When the user selects to expand a menu panel, it would be useful to start from the top of that panel when the page reloads.
Maintanability
⇑ If function used frequently, then it is in favourites ⇓ Users may find this function too complicated. Consider providing a step by step process.

User group 3: Motor-impaired users
Function 3: View resource in Resource centre
Visibility
⇑ This function can be reached through different shortcuts and from navigation menu.

Usefulness
⇑ The alternative tab styles (selected, empty, available) are useful and intuitive.
⇑ The function is easy to accomplish.
Availability – approachability
-
Quality of usage experience
-
Maintanability
⇓ This function uses descriptive labels and easy to accomplish.
⇓ If function used frequently, then it is in favourites

In conclusion, the individual rates achieved by the current (prototype) design of the EDeAN portal are more than satisfying (see Table 31). More specifically, for all three indicative user types inspected (blind, colour-blind and motor impaired) achieved high rates of availability-approachability (reflecting in a broad sense accessibility) and of quality of interaction (reflecting in a broad sense usability); in almost all cases the rates were above the acceptance borderline (zero value) and ranged from 1 and 3, which reflect slight to major examples of good UI design practices. Yet, as shown in the above received qualitative evaluation data, there were few design cases identified which need to be reconsidered or slightly improved.

Table 31: **Overview of the quality of user experience of the EDeAN prototype**

Overall quality of interaction experience

	First-time and novice users					Moderate users			Expert users			Total
	V	U	A	Q	M	A	Q	M	A	Q	M	
User Group "Blind users"	0	1	1	1	0	2	2	0	0	2	0	0.81
User Group "Color-blind"	1	0	1	2	1	2	2	0	1	2	0	1.09
User Group "Motor-impaired"	2	2	2	2	0	2	2	0	2	3	0	1.54
Total	1.00	1.00	1.33	1.66	0.33	2.00	2.00	0.00	1.00	2.33	0.00	1.146
	1.066					1.33			1.33			

Where:

- **V** stands for **Visibility**.
- **U** stands for **Perceived usefulness & ease of use**.
- **A** stands for **Availability & approachability**.
- **Q** stands for **Quality of interaction experience**.
- **M** stands for **Relationship maintainability**.

- Boxes with values from "-4" to "-3" are coloured with red (i.e. user-orientation catastrophes).
- Boxes with values from "-3" to "-2" are coloured with orange (i.e. major problems).
- Boxes with values from "-2" to "-1" are coloured with yellow (i.e. minor problems).
- Boxes with values from "-1" to "1" are coloured with green (i.e. cosmetic problems and / or examples of good practice).
- Boxes with values from "1" to "2" are coloured with blue (i.e. minor examples of good design solutions).
- Boxes with values from "2" to "3" are coloured with indigo (i.e. major examples of good design solutions).
- Boxes with values from "3" to "4" are coloured with violet (i.e. best design solution example (maximal fit)).

Chapter 9
CONCLUSIONS AND FUTURE WORK

This work proposes a novel approach to the development of web user interfaces that is based on the Unified User Interfaces methodology. The proposed approach, intended as an alternative to traditional design for the 'average' user, guides the development of Unified Web Interfaces (UWI) and aims to ensure accessibility and usability for users with diverse characteristics. Then, the EAGER toolkit was implemented in order to further facilitate Web developers in effectively following the proposed approach in practice. In this context, a number of UI elements were designed in various forms (polymorphic task hierarchies) according to specific user- / context- parameter values. This phase provided feedback to the actual development process of EAGER, which involved the development of the alternative interaction elements and the mechanisms for facilitating the dynamic activation - deactivation of interaction elements and modalities based on individual user interaction and accessibility preferences.

The EAGER toolkit was then employed to develop a new portal for the *European Design for All and e-Accessibility* (EDeAN) network. In this context, several different modules were implemented, such as Special Interest Groups, Digital Library, Training courses, and a complete user profiling mechanism. This effort provided valuable feedback in a number of directions. Initially, the development of such a large scale application proved the viability and consistency of the toolkit and elucidated its ability to stand as a horizontal and efficient development tool. A number of alternative evaluation techniques were applied to EAGER and to the developed prototype portal. The alternative interaction elements and dialogue controls provided by the EAGER were evaluated in order to ensure their conformance to the W3C accessibility guidelines for Web content. Additionally, in order to verify the accessibility and usability offered via the development of a complete web portal by means of EAGER, an innovative, expert-based evaluation technique for universally accessible applications (the ORIENT inspection tool) was used. This assessment confirmed the expected benefits both in terms of accessibility and usability of the final product, taking also into account various levels of user expertise.

Another key feature of the EAGER toolkit is its ability to be extended and include an unlimited number of alternative interaction modalities and elements. This process entails, mainly, the design and coding of the alternative interactions styles. Then, they can be easily incorporated in the existing toolkit, simply by modifying the decision logic for supporting their conditional activation and deactivation. Additionally, existing Web applications or parts of applications implemented with .NET can be easily altered to encapsulate the EAGER toolkit attributes and, thereby, rendered accessible and usable for various user categories, including novice users, users of Assistive Technologies or portable devices, etc. Notably, an existing Web application, Ariadne, when implemented from scratch using the EAGER toolkit, ended up with 50% less lines of pure code in total, showing that, thanks to abstraction, EAGER generates shorter, more robust and comprehensive source codes.

In the EDeAN portal case study, the correlation of the various alternative designs of UI elements to user and context related parameters (i.e., the automatic adaptation according to generic, predefined profiles) has been made on a normative basis. Therefore, it can not currently be claimed to be optimal, and needs to be further verified in the future, for example through feedback from user trials in real contexts.

However, this work has made clear that the proposed approach allows embedding in Web-based applications such decision making logics and automatic adaptation facilities for the benefit of accessibility and better user experience. On the other hand, it has also proved that the proposed approach can produce Web applications that allow their users to select themselves (i.e., customise) the designs they prefer most.

Along the same line, the produced UI element designs are not claimed to optimum, for example in terms of aesthetics. For instance, a Web designer may have completely different ideas for designing buttons, menus, virtual keyboards, etc. However, it was made clear that it is feasible to develop Web-based interfaces that can support and import alternative designs for fully accessible and personalised ways of interactions, without payoffs in terms of aesthetics or inclusiveness.

Concerning additional enhancements of the EAGER toolkit, several advanced and intelligent techniques have been identified which can improve its effectiveness and efficiency of the system. For instance, since the EAGER toolkit currently allows retrieving statistics on the designs preferred among various types of users, an advanced mechanism could be built-in in the future for allowing users to rapidly configure the layout and behaviour of their personal interface according to what other users with similar profiles have selected.

On the other hand, once all design elements are assessed in terms of their conformance to the W3C guidelines and categorised according to their conformance level (i.e., single A, double A and triple A), another adaptation that could be supported by EAGER, is to allow users (or developers) to select automatically which particular level of conformance they require, instead of selecting among predefined profiles and the assigned designs.

Another potential direction for future work concerns the integration of EAGER with a design support tool for the Unified User Interface Design method. For example, the MENTOR tool [61] provides practical support for all phases of Unified User Interface Design, by appropriately guiding the process and structuring the outcomes of creative design steps through appropriate editing facilities for: (i) encoding declarations (signatures) of design parameters attributes and related value spaces; (ii) creating profiles of adaptation conditions (i.e., macros of instantiated combinations of design parameters), (iii) encoding polymorphic task hierarchies; and (iv) attaching information to artefacts (nodes) in such hierarchies. In addition, MENTOR performs automated verification of the designed adaptation logic, as well as automated generation of "ready-to-implement" interface specifications, including the adaptation logic. By integrating a web-based version of MENTOR in EAGER, it will be possible to achieve a tool which graphically supports the development process of UWIs from design to implementation, and allows to easily and semi-automatically extending the EAGER user and context profiles and adaptation logic.

Finally, another potential direction of future work is to render the EAGER toolkit a Web service so that Web developers using technologies other .NET will be able to incorporate the EAGER adaptation logic into their artefacts, by providing an interface for defining profiles and receiving decisions regarding the activation – deactivation of alternative UI elements. Then, the developer will only have to implement, if not available, the proposed alternative designs in their own development environments. Overall, the work presented here is considered as a significant contribution towards embedding accessibility, graceful transformation and ease of use for all in future and existing Web-based applications, and, ultimately, towards supporting individuals to fully participate in the knowledge society, especially people at risk of exclusion.

REFERENCES

1 Galitz, W.O. (1997) Essential guide to user interface design: An Introduction to GUI design principles and techniques. Canada, John Wiley and Sons, Inc. (ISBN: 0-471-15755-4).

2 Mayer, B.A., & Rosson, M.B. (1992) *Survey on user interface programming*. In Proc. of CHI'92, May 3-7, Monterey, USA.

3 Stephanidis, C. (2001) *The concept of Unified User Interfaces*. In C. Stephanidis (Ed.), User Interfaces for All - Concepts, Methods, and Tools. Mahwah, NJ: Lawrence Erlbaum Associates, pp. 371-388 (ISBN 0-8058-2967-9).

4 Stephanidis, C., Savidis A., & Akoumianakis D. (1997) *Unified Interface Development: Tools for Constructing Accessible and Usable User Interfaces*, Tutorial no 13 in HCI International '97 Conference, 24-29 August, San Francisco, USA.

5 Stephanidis, C. (Ed.), Salvendy, G., Akoumianakis, D., Bevan, N., Brewer, J., Emiliani, P.L., Galetsas, A., Haataja, S., Iakovidis, I., Jacko, J., Jenkins, P., Karshmer, A., Korn, P., Marcus, A., Murphy, H., Stary, C., Vanderheiden, G., Weber, G., & Ziegler, J. (1998) *Toward an Information Society for All: An International R&D Agenda*. International Journal of Human-Computer Interaction, 10 (2), 107-134.

6 Bergman, E., & Johnson, E. (1995) *Towards Accessible Human-Computer Interaction*. In J. Nielsen (Ed.), Advances in Human-Computer Interaction (vol. 5) (pp. 87-113). New Jersey: Ablex Publishing Corporation.

7 Story, M.F. (1998). *Maximising Usability: The Principles of Universal Design*. Assistive Technology, 10, 4-12.

8 Vanderheiden, G. (1998) *Universal Design and Assistive Technology in Communication and Information Technologies: Alternatives or Compliments?* Assistive Technology, 10 (1), 29-36.

9 De La Rue, M. (2002) Linux Accessibility: HOWTO v3.1 (Retrieved on September 2007 from http://tldp.org/HOWTO/Accessibility-HOWTO/)

10 World Wide Web Consortium (1999) *Web Content Accessibility Guidelines 1.0*. W3C Recommendation, (Retrieved on September 2007 from http://www.w3.org/TR/WCAG10/)

11 Vanderheiden, G., Chisholm, W., and Ewers, N. (1996) *Design of HTML pages to increase their accessibility to users with disabilities, Strategies for today and tomorrow*. Technical Report, Trace R&D Centre, University of Wisconsin - Madison.

12 Richards, J. (1996) *Guide to Writing Accessible HTML*. University of Toronto.

13 Gunderson, J. (1996) *World Wide Web Browser Access Recommendations*. University of Illinois at Urbana / Champaign.

14 De Quincey, T. (1842) Ricardo and Adam Smith (Part III). Blackwoods Magazine, 52, pp. 718-739.

15 ISO 9241 (1998) *ISO DIS 9241 – Part 11: Guidance on usability*. ISO standard.

16 Nielsen J. (1993), Usability Engineering, Academic Press Limited.

17 Shneiderman B (1980) *Software Psychology* (ISBN 0-87626-816-5).

18 Instone, K. (2000). *Usability Heuristics for the Web* (Retrieved on September 2007 from: http://web.archive.org/web/19990429162604/webreview.com/wr/pub/97/10/10/usability/sidebar.html)

19 Stephanidis, C., Paramythis, A., Karagiannidis, C., & Savidis, A. (1997). *Supporting Interface Adaptation: The AVANTI Web-Browser*. In C. Stephanidis & N. Carbonell (Eds.), Proceedings of the 3rd ERCIM Workshop "User Interfaces for All", Obernai, France, 3-4 November.

20 Hermsdorf, D. (1998*) WebAdapter: A Prototype of a WWW Browser with new Special Needs Adaptations*. In Edwards, A., Arato, A., Zagler, W. (eds.): Proceedings of the XV IFIP World Computer Congress, Computers and Assistive Technology ICCHP 98, Vienna/Budapest 1998, pp. 151-160.

21 Henricksen, K. & Indulska, J. (2001) *Adapting the Web Interface: An Adaptive Web Browser*. In Proceedings of the Australasian User Interface Conference 2001, Australian Computer Science Communications, Volume 23, Number 5.

22 Alexandraki, C., Paramythis, A., Maou, N., & Stephanidis, C. (2004). *Web Accessibility through Adaptation*. In: K. Miesenberger, J. Klaus, W. Zagler, & D. Burger (Eds.), Proceedings of Computers Helping People with Special Needs: 9th International Conference, ICCHP 2004 Paris, France, July 7-9, 2004 (pp. 302 - 309). Berlin / Heidelberg: Springer, Lecture Notes in Computer Science, Volume 3118 / 2004 (ISSN: 0302-9743)

23 Iaccarino, G., Malandrino, D., & Scarano, V. (2006) *Personalizable edge services for web accessibility*. In Proceedings of the 2006 international Cross-Disciplinary Workshop on Web Accessibility (W4a): Building the Mobile Web: Rediscovering Accessibility? (Edinburgh, U.K., May 22 - 22, 2006). W4A, vol. 134. ACM Press, New York, NY, 23-32.

24 Brown, S. S., & Robinson P. (2001) A World Wide Web Mediator for Users with Low Vision. In Proceedings of CHI 2001 Workshop No. 14.

25 Parmato B., Ferydiansyah R., Zeng X., Saptono A., & Sugiantara I. W. (2005) *Accessibility Transformation Gateway*. In Proc. of 38th Hawaii International Conference on System Sciences.

26 Han R., Bhagwat P., Lamaire R., Mummert T., Perret V., & Rubas J. (1998) *Dynamic Adaptation In an Image Transcoding Proxy For Mobile Web Browsing.* IEEE Personal Communications, 5(6), December 1998.

27 Gupta, S., Kaiser, G. (2005) *Extracting content from accessible web pages*. In W4A '05: Proceedings of the 2005 International Cross-Disciplinary Workshop on Web Accessibility (W4A), pages 26–30, New York, NY, USA, 2005. ACM Press.

28 Gupta, S., Kaiser, G. E., Grimm, P., Chiang, M. F., & Starren J. (2005*) Automating Content Extraction of HTML Documents*. World Wide Web, 8(2):179–224.

29 Barrett, R., & Maglio, P. (1998) *Adaptive Communities and Web Places*. In Proceedings of the 2th Workshop on Adaptive Hypertext and Hypermedia, HYPERTEXT 98.

30 Barrett, R., & Maglio P. (1999) *Intermediaries: An approach to manipulating information streams*. IBM Systems Journal, 38(4):629–641.

31 Barrett, R., & Maglio P (1999) *WebPlaces: Adding people to the Web*. In Proceedings of 8th International World Wide Web Conference, Toronto (Canada), ACM Press.

32 Barra, M., Grieco, R., Malandrino, D., Negro, A., & Scarano V. (2003) *TextToSpeech: an heavy-weight Edge computing Service*. In Poster Proc. of 12th International World Wide Web Conference. ACM Press, May 2003.

33 Chen, Y., Xie, X., Ma, W.-Y., Zhang, H.-J. (2005) *Adapting Web Pages for Small-Screen Devices*. IEEE Internet Computing, v.9 n.1, p.50-56.

34 Graef, G. (2000) *Adaptation of Web-Applications Based on Automated User Behaviour Analysis*. In Proceedings of the Second Annual Conference on World Wide Web Applications (WWW 2000), Johannesburg, Südafrika, pp.72-75.

35 Taib, R., & Ruiz, N. (2006) *Multimodal Interaction Styles for Hypermedia Adaptation*. In Proc. International Conference on Intelligent User Interfaces (IUI'06), Sydney, Australia, pp. 351-353.

36 Stephanidis, C. (2001). New Perspectives into Human – Computer Interaction. In C. Stephanidis (Ed.), User Interfaces for All - Concepts, Methods, and Tools Mahwah, NJ: Lawrence Erlbaum Associates, pp. 3-20 (ISBN 0-8058-2967-9).

37 Stephanidis, C., Savidis, A., & Akoumianakis, D. (1995). *Tools for User Interfaces for all*. In I.Placencia-Porrero & R. Puig de la Bellacasa (Eds.), The European context for Assistive Technology, Proceedings of 2nd TIDE Congress, Brussels, Belgium, Amsterdam: IOS Press, pp. 167-170.

38 Savidis, A., Stephanidis, C. (2004). *Unified User Interface Design: Designing Universally Accessible Interactions*. International Journal of Interacting with Computers.

39 Savidis, A., Stephanidis, C. (2004). *Unified User Interface Development: Software Engineering of Universally Accessible Interactions*. *Universal Access in the Information Society*, 3 (3) (Managing Editor: Alfred Kobsa, University of California, Irvine, USA).

40 Akoumianakis, D., Savidis, A., and Stephanidis, C. (2000). *Encapsulating Intelligent Interactive Behaviour in Unified User Interface Artefacts*. International Journal of Interacting with Computers, special issue on "The Reality of Intelligent Interface Technology", 12 (4), 383-408.

41 Savidis, A., Stephanidis, C., and Emiliani, P.L. (1997). *Abstract Task Definition and Incremental Polymorphic Physical Instantiation: The Unified Interface Design Method*. In G. Salvendy, M.J. Smith & R.J. Koubek (Eds.), Design of Computing Systems: Cognitive Considerations [Proceedings of the 7th International Conference on Human-Computer Interaction (HCI International '97)], San Francisco, USA, 24-29 August (vol. 1, pp. 465-468). Amsterdam: Elsevier, Elsevier Science.

42 Stephanidis, C., Akoumianakis, D., Sfyrakis, M., and Paramythis, A. (1998). *Universal accessibility in HCI*: Process-oriented design guidelines and tool requirements. In C. Stephanidis & A. Waern (Eds.), Proceedings of the 4th ERCIM Workshop on "User Interfaces for All", Stockholm, Sweden, 19-21 October (15 pages).

43 Stephanidis, C., Savidis, A., and Akoumianakis, D. (2001). *Tutorial on "Universally accessible UIs: The unified user interface development"*. Tutorial in the ACM Conference on Human Factors in Computing Systems (CHI 2001), Seatle, Washington, 31 March - 5 April.

44 Stephanidis, C. (2001). New Perspectives into Human – Computer Interaction. In C. Stephanidis (Ed.), User Interfaces for All - Concepts, Methods, and Tools (pp. 3-20). Mahwah, NJ: Lawrence Erlbaum Associates (ISBN 0-8058-2967-9, 760 pages).

45 Stephanidis, C., Emiliani, P.L. (1999). *Connecting to the Information Society: a European Perspective*. Technology and Disability Journal, 10 (1), 21-44.

46 Savidis, A., Stephanidis, C. (2001). *The Unified User Interface Software Architecture*. In C. Stephanidis (Ed.), User Interfaces for All - Concepts, Methods, and Tools (pp. 389-415). Mahwah, NJ: Lawrence Erlbaum Associates (ISBN 0-8058-2967-9, 760 pages).

47 ACM *Special Interest Group on Computer - Human Interaction Curriculum Development Group*. ACM SIGCHI curricula for human-computer interaction. Technical report, ACM, New York, 1992.

48 Savidis, A., Akoumianakis, D., and Stephanidis, C. (2001). *The Unified User Interface Design Method*. In C. Stephanidis (Ed.), User Interfaces for All - Concepts, Methods, and Tools (pp. 417-440). Mahwah, NJ: Lawrence Erlbaum Associates (ISBN 0-8058-2967-9, 760 pages).

49 Hoare, C.A.R. (1978). *Communicating Sequential Processes*. Communications of the ACM, 21 (8), 666-677.

50 Hartson, H.R., Siochi, A.C., and Hix, D. (1990). The UAN: *A User-Oriented Representation for Direct Manipulation Interface Design*. ACM Transactions on Information Systems, 8 (3), 181-203.

51 Fraternali P., (1999), *Tools and Approaches for Developing Data-Intensive Web Applications: A survey, ACM Computing Surveys*, Volume 31, Number 3, pp. 227-263, New York, NY, USA:ACM Press (ISSN 0360-0300).

52 Jorgensen D. (2002) *Developing .Net Web Services with XML*, Syngress Publishing Inc.Liberty J. (February 2002) Programming C#, 2nd Edition, O'Reilly.

53 Basiura, R., Batongbacal, M., Bohling, B., Clark, M., Eide, A., Eisenberg, R., Loesgen, B., Miller, C.L., Reynolds, M., Sempf, B., Sivakumar, S. (2001). Professional ASP.NET Web services, (November 2001), Birmingham, UK, Wrox Press Ltd (ISBN 1-86100-545-8).

54 Mourouzis, A., Antona, M., Kastrinaki, A., & Stephanidis, C. (2006). *ORIENT: An Expert-based Tool for Assessing the User-orientation of eServices*. In P. Cunningham and M. Cunningham (Eds.), Exploiting the Knowledge Economy: Issues, Applications and Case Studies, Proceedings of eChallenges e-2006 Conference, Barcelona, Spain, 25 - 27 October (pp. 443-450). Amsterdam, The Netherlands: IOS Press.

55 Antona, M., Mourouzis, A., & Stephanidis, C. (2007). Towards a Walkthrough Method for Universal Access Evaluation. In C. Stephanidis (Ed.), Universal Access in HCI – PART I of the Proceedings of 4th International Conference on Universal Access in Human-Computer Interaction (UAHCI 2007 - held as Part of HCI International 2007), Beijing, China, July 22-27 (pp. 325-334), LNCS 4554, Springer-Verlag, Berlin Heidelberg (ISBN: 978-3-540-73278-5).

56 Mourouzis, A., Antona, M., Boutsakis, E., Kastrinaki, A., & Stephanidis, C. (2006). User-orientation Evaluation Framework for eServices: Inspection tool and usage guidelines. FORTH-ICS Technical Report, TR-372.

57 Antona, M., Mourouzis, A., Kastrinaki, A., Boutsakis, E., & Stephanidis C. (2006) User-orientation inspection of ten European eServices: Results and lessons learned. FORTH-ICS Technical Report, TR-373.

58 Gaedke, M., Schempf, D., Gellersen, H. 1999. WCML: An enabling technology for the reuse in object-oriented Web Engineering, in: Poster-Proceedings of the 8th International World Wide Web Conference (WWW8), Toronto, Ontario, Canada.

59 Nielsen, J. (2006) *Screen Resolution and Page Layout*. Nielsen, J., Alertbox, Available electronically at: http://www.useit.com/alertbox/screen_resolution.html.

60 Preece, J. (2000). Online communities: Designing usability, supporting sociability. Chichester, England: John Wiley & Sons.

61 Antona, M., Savidis, A., & Stephanidis, C. (2006). A Process–Oriented Interactive Design Environment for Automatic User Interface Adaptation. International Journal of Human Computer Interaction, 20 (2), 79-116.

APPENDIX A: Development of alternative designs

Following the U^{2}I Design method, the design of the user interface of EAGER dialogue controls followed three main stages:

- Enumeration of different design alternatives to cater to the particular requirements of the users and the specific context of use.
- Encapsulation of the design alternatives into appropriate abstractions and integration into a polymorphic task hierarchy.
- Development and documentation of the design rationale that will drive the run-time selection between the available alternatives.

The following section presents the design outcomes, including the polymorphic task hierarchies, the design space populated by the produced physical designs and for each polymorphic artefact in the hierarchy a design rationale recording its run-time adaptation logic based on user- and context-related parameters.

Template alternatives styles – size

A portal template generally maps to the generic scheme that incorporates the containers hosting contents. As presented in Figure 45 (pp. 68), two generic template styles were designed. The linearized template style contains all the containers (top navigation, content, bottom navigation) in a linear form. On the other hand, the columns template style has three alternative styles where top and bottom navigation are placed on the top and bottom positions and the centered container is split in two, three or four columns respectively for the two, three, four columns template.

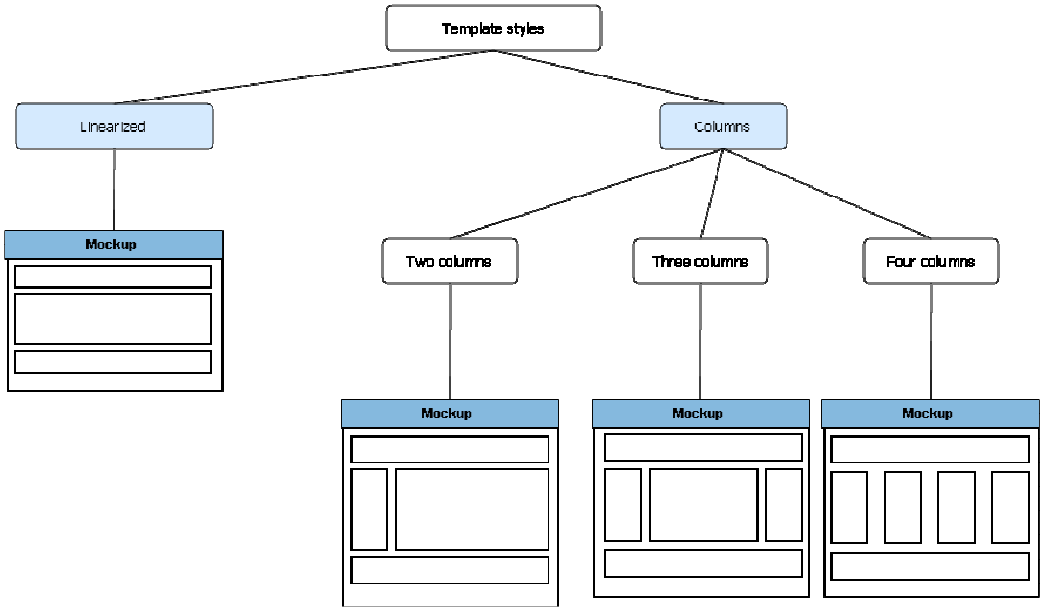


Figure 151: **Template alternative styles**

According to the design rationale presented in Table 22, the linearized template supports speed, naturalness and flexibility for blind or low vision users whereas the columns templates sustain speed, flexibility and cover the optimum screen size for

users with no visual impairments. The alternative columns templates intend to be used in order to support content flexibility.

Table 32: **Design rationale of the template styles**

Task: Template styles		
Style:	Linearized	Columns
Targets:	Accessibility, speed, naturalness, flexibility	Speed, flexibility, cover optimum screen size
Parameters:	User (Blind, Low vision)	User (No visual impairments)
Properties:	-	-
Relationships:	Exclusive	Exclusive

Template size constitutes another significant aspect that is associated with the screen resolution in which the portal will be presented. According to (Arditi, 1996), a web page has to be optimised for 1024 x 768 resolution, but has to stretch well for any resolution, from 800 x 600 to 1280 x 1024 using a liquid layout. As it is presented in Figure 46 and Table 23 the template size may be resized according the device screen resolution in order to cover the optimum screen size.

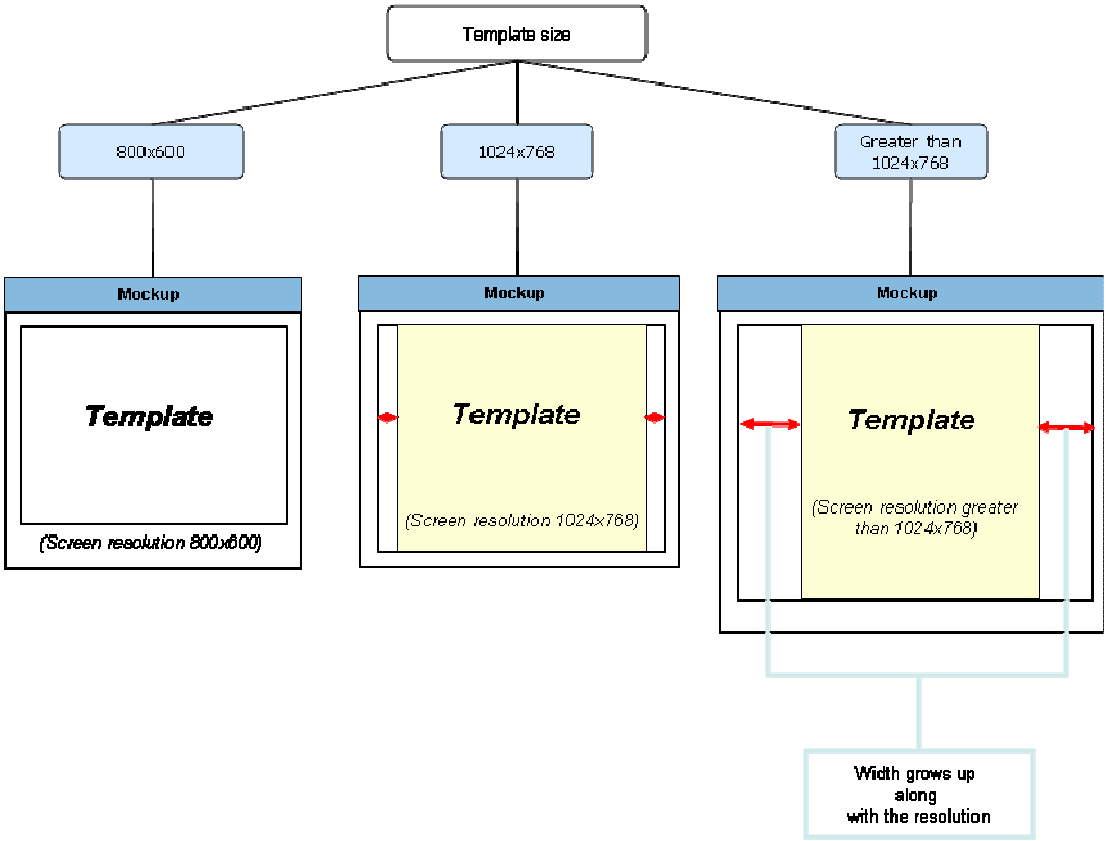


Figure 152: **Template alternatives according to device resolution**

When resolution is 800 x 600, the template covers all the surface of the screen, whereas for 1024 x 768 resolutions the template lays left and right a small unexploited area, in order to maximize readability of the contents. For resolutions greater than 1024 x 768, the width of the empty area left and right of the template are increased according to screen resolution.

Table 33: **Design rationale of template alternatives according to device resolution**

Task: Template styles			
Style:	800x600	1024x768	Greater than 1024x768
Targets:	Cover optimum screen size	Cover optimum screen size	Cover optimum screen size
Parameters:	Device resolution: 800x600	Device resolution: 1024x768	Device resolution: greater than 1024x768
Properties:	-	-	-
Relationships:	Exclusive	Exclusive	Exclusive

Window alternatives

Template containers may include windows. In Figure 153 all the design alternatives that were produced during the design phase are outlined. As shown in Table 34, the alternative windows styles refer to user individual preferences and constitute part of the skin that the user selects.

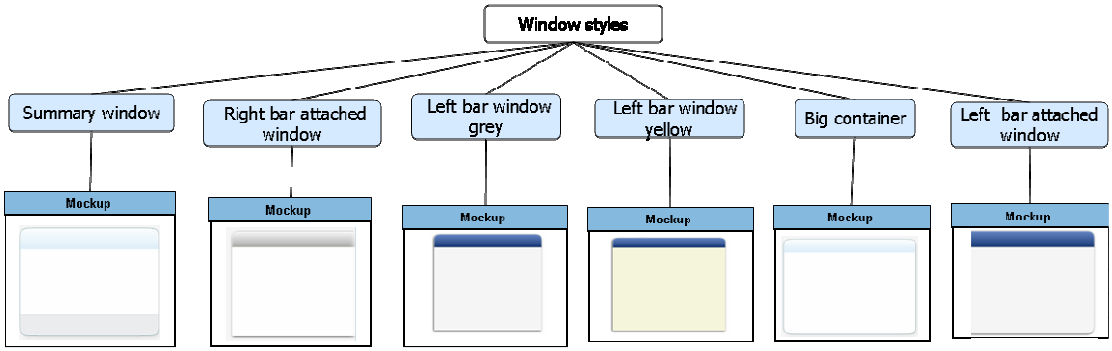


Figure 153: **Windows alternative styles**

Table 34: **Design rationale of window alternative styles**

Task: Window styles						
Style:	Summary window	Right bar attached window	Left bar window grey	Left bar window yellow	Big container	Left bar attached window
Targets:	Usability, flexibility	Usability, flexibility	Usability, flexibility	Usability, flexibility	Usability, flexibility	Usability, flexibility
Parameters:	User preferences	User preferences	User preferences	User preferences	User preferences	User preferences
Properties:	-	-	-	-	-	-
Relationships:	Exclusive	Exclusive	Exclusive	Exclusive	Exclusive	Exclusive

Navigation alternatives

Navigation constitutes one of the main mechanisms that a web portal user uses. Multiple alternatives of the navigation mechanism were designed in order to support individual user abilities and preferences, and are presented in Figure 154.

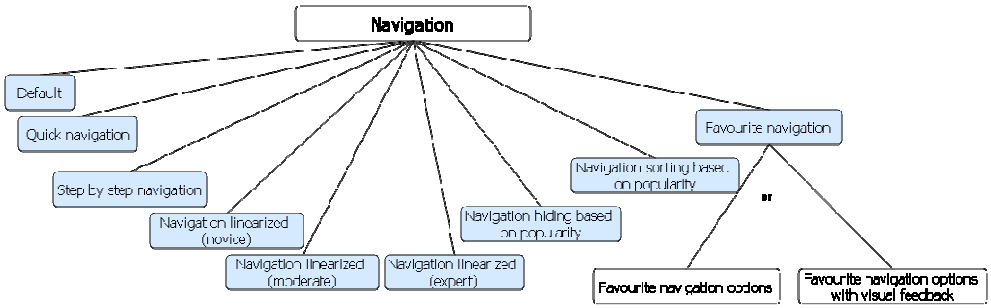


Figure 154: **Navigation alternatives**

The default navigation is presented as part of the Figure 155; each navigation hierarchy of the portal is grouped in a separate portal window or navigation bar and when a user selects a navigation element, the available navigation sub-elements are presented as links over it.

The quick navigation alternative was designed for users with high expertise in web applications; it has all the navigation hierarchies grouped in a portal window; when the user selects a navigation hierarchy, the navigation hierarchy elements appear over it. In this way, the expert user holds all the navigation hierarchies grouped in an area, thus enhancing navigation speed and user's flexibility.

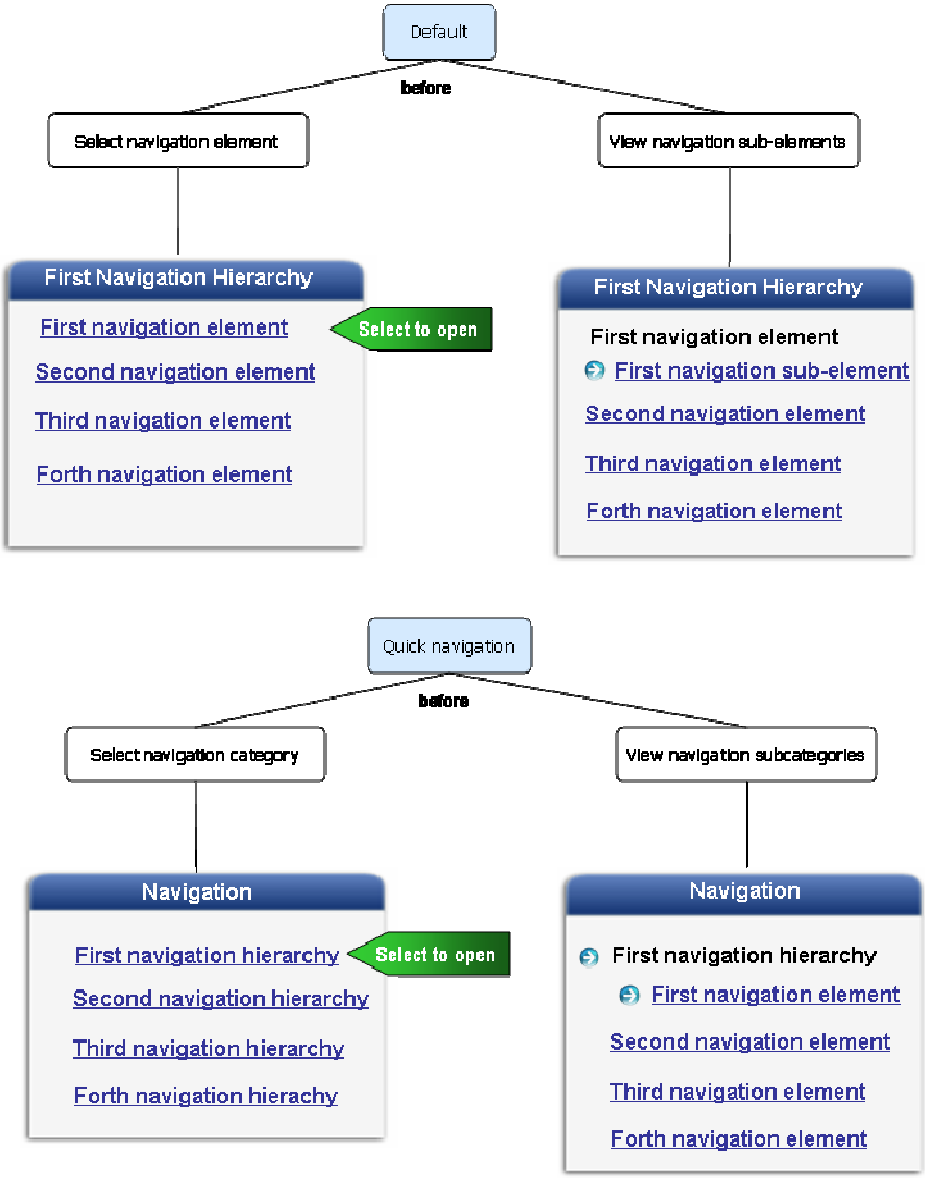


Figure 155: **Default – Quick navigation alternatives**

Table 35: **Design rationale of default and quick navigation alternatives**

Task: Navigation (1/3)		
Style:	Default	Quick navigation
Targets:	Usability	Speed, flexibility
Parameters:	Default User	User (Expert web knowledge)
Properties:	Navigation element first (from desired navigation hierarchy), navigation sub-element next	Navigation hierarchy first (from grouped navigation), navigation element next, navigation sub-element next
Relationships:	Exclusive	Exclusive

The step by step navigation alternative is designed to serve motor impaired users. In the step by step navigation hierarchy, windows are presented initially closed; when the user selects a navigation hierarchy, the window opens and the navigation elements appear above it. The sub-navigation elements appear when a navigation element is selected. In this way, the web page includes the minimum navigation links each time a motor impaired user tries to navigate through it.

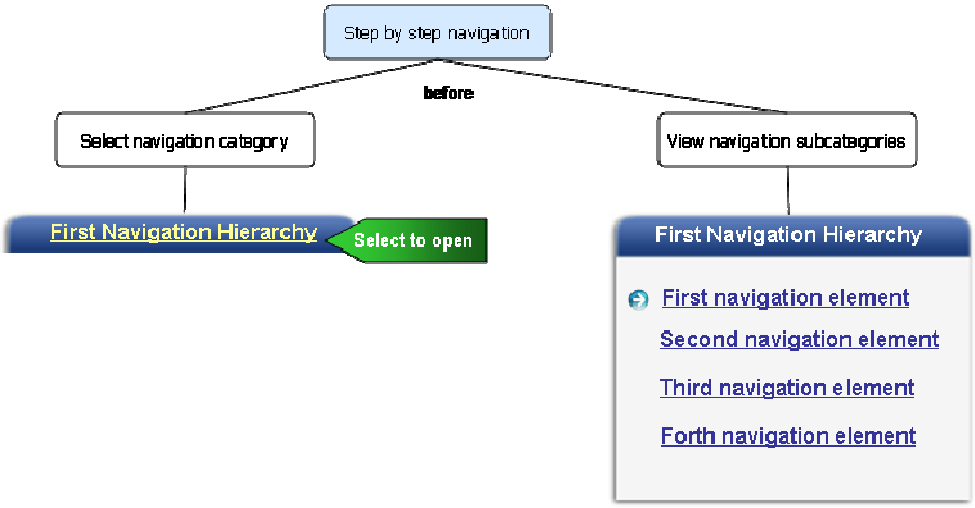


Figure 156: **Step by step navigation alternative**

The linearized navigation for novice users (see Figure 157) offers a linear form for all the navigation links of the portal, and in parallel a step by step navigation is supported. Initially, the user has to select among navigation hierarchies, next among entire navigation elements, and finally among entire navigation sub-elements. In each step, the previous hierarchy is available in order to navigate back to another navigation hierarchy or navigation element. This step by step navigation mechanism offers guided navigation to novice users that face vision impairment.

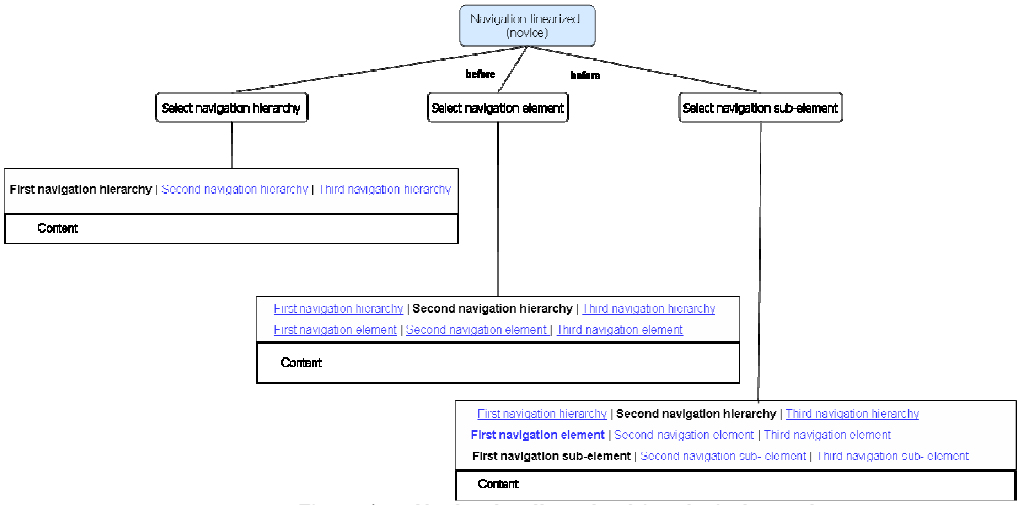


Figure 157: **Navigation linearized (novice) alternative**

The linearized navigation is targeted to moderate users that face visual impairments, and supports a linearized form of the entire navigation of the portal. Initially, the user select among navigation hierarchies and then the available navigation elements for the selected navigation hierarchy are presented, along with a navigation path through which the user may navigate back to the navigation hierarchy. Through this procedure, the user has to scan limited navigation options using the screen reader, knows each time which page are being browsed, and always has an efficient way to navigate back to the navigation hierarchies due to the path mechanism (see Figure 158).

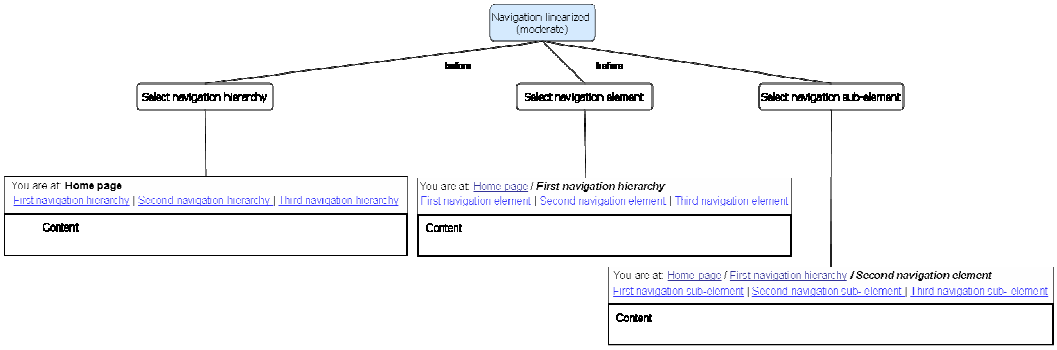


Figure 158: **Navigation linearized (moderate) alternative**

The linearized navigation, targeted to users that face visual impairments and are experts in web applications, resembles linearized navigation for moderate users, but without the path mechanism. The expert has the ability to navigate back to the navigation hierarchy, but is not notified about the browsed web page each time (see Figure 159). Therefore, the expert user can browse the navigation mechanism quickly, without having the screen reader always reading the path of the entire page.

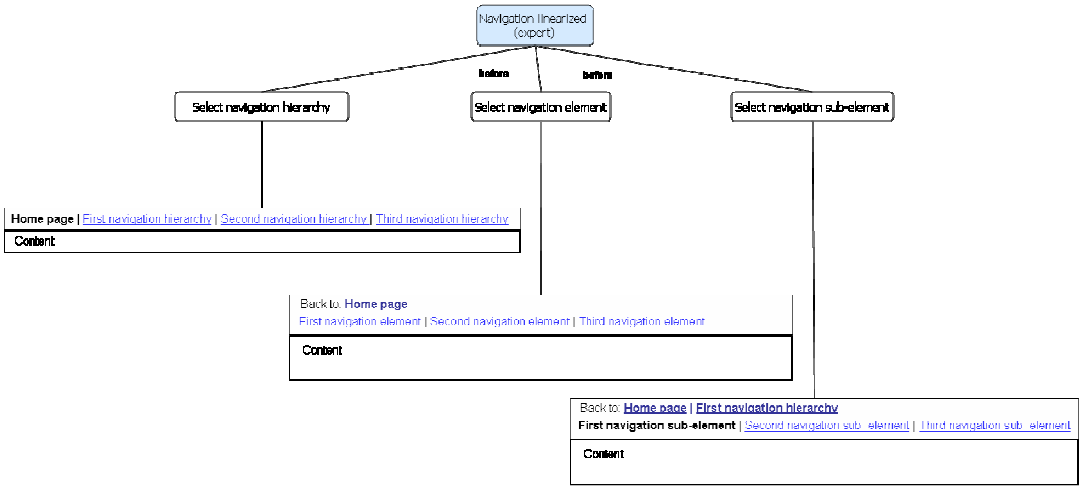


Figure 159: **Navigation linearized (expert) alternative**

Table 36: **Design rationale of the step by step navigation (novice, moderate, expert)**

Task: Navigation (2/3)				
Style:	Step by Step navigation	Navigation linearized (novice)	Navigation linearized (moderate)	Navigation linearized (expert)
Targets:	Easiness, minimum navigation elements to scan, speed	Accessibility, flexibility, usability	Accessibility, flexibility, usability, limited reading by the screen reader	Accessibility, speed, flexibility, usability, limited reading by the screen reader
Parameters:	User (motor impaired)	User (Blind or Low vision and novice web expertise)	User (Blind or Low vision and moderate web expertise)	User (Blind or Low vision and expert web expertise)
Properties:	Navigation hierarchy first, navigation element next(from desired navigation	Navigation hierarchy first, navigation element next	Navigation hierarchy first, navigation element next	Navigation hierarchy first, navigation element next (for desired navigation

	hierarchy navigation), navigation sub-element next (from desired navigation element)	(for desired navigation hierarchy), navigation sub-element next (for desired navigation element)	(for desired navigation hierarchy), navigation sub-element next (for desired navigation element)	hierarchy), navigation sub-element next (for desired navigation element)
Relationships:	Exclusive	Exclusive	Exclusive	Exclusive

Except from the navigation alternatives targeted to specific user needs such as visual impairments, other four navigation alternatives were designed in order to be selected by users optionally. These are presented in Figure 160. The first option "navigation hiding based on frequency of use" configures the default option navigation in such a way that the navigation elements that are not used for a long time by the specific user are hided, and only the popular navigation elements are visible to the user for each navigation hierarchy. The less frequently used navigation elements may be accessed by clicking the link "More…" The second option "navigation sorting based on frequency of use" does not hides the less frequent navigation element, but rearranges them based on frequency by positioning on the top the most used navigation elements by the user.

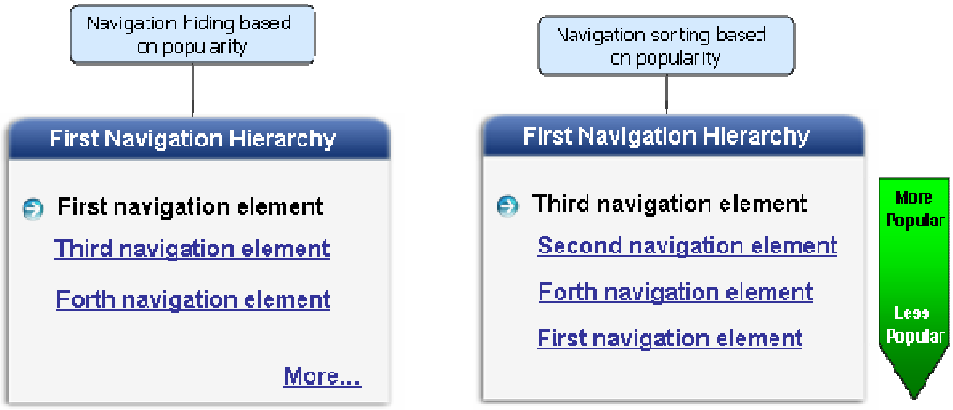


Figure 160: **Navigation hiding and sorting based on frequency of use**

There two alternative designs for the "favourites navigation" that are presented in Figure 161. In the first design artefact, the most used navigation elements are grouped and placed in a new navigation window named "my favourites tasks". This window offers to the user who selects this option all the familiar navigation elements together, and improves the speed of using portal. The "favourite navigation tasks with visual feedback" design alternative simply stamps the most used navigation elements with a star in order to be located more quickly by the user. The design rationale of the optional navigation alternatives is recorded in Table 37.

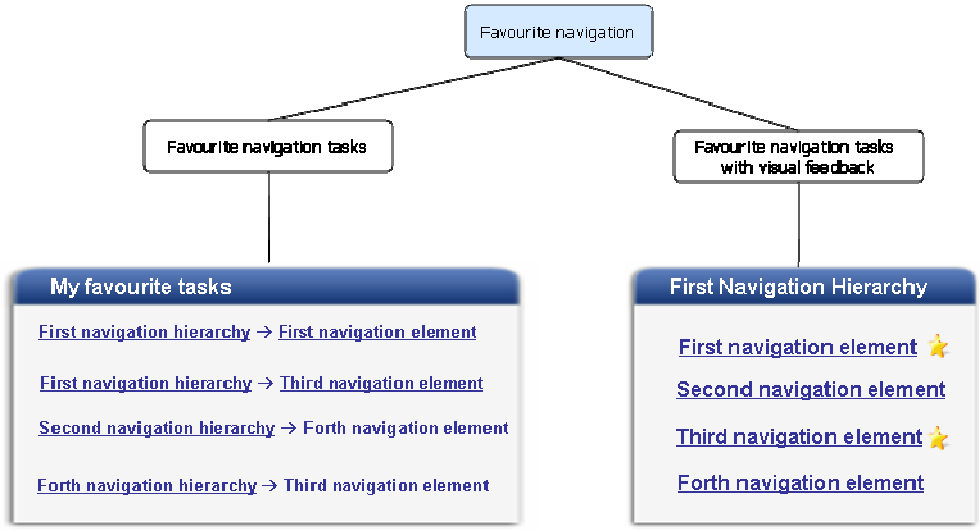


Figure 161: **Favourites navigation alternatives**

Table 37: **Design rationale of navigation hiding, sorting, bookmarking**

Task: Navigation (3/3)				
Style:	Navigation hiding based on frequency of use	Navigation sorting based on frequency of use	Favourites navigation tasks	Favourites navigation tasks with visual feedback
Targets:	Speed, most used navigation elements visible	Speed, most used navigation elements on the top	Speediness to localization, efficiency	Speediness to localization, efficiency
Parameters:	User preferences	User preferences	User preferences	User preferences
Properties:	-	-	-	-
Relationships:	Exclusive	Exclusive	Exclusive	Exclusive

Links - Buttons alternatives

Links and buttons comprise maybe the most used objects in a web page. Several alternatives were designed in order to offer effective interaction based on user abilities and context of use. As shown in Figure 162, links remains the same and the only difference concerns the background and text colour. Generally, in the web portal links are designed to be presented in blue. However, several different colour combinations for low vision users and user with colour blindness were produced based on (Arditi, 1996).

According to (Arditi, 1996) there are three basic guidelines for making effective colour choices that work for nearly everyone:

1. Exaggerating lightness differences between foreground and background colours and avoiding using colours of similar lightness adjacent to one another, even if they differ in saturation or hue (see Figure 162).

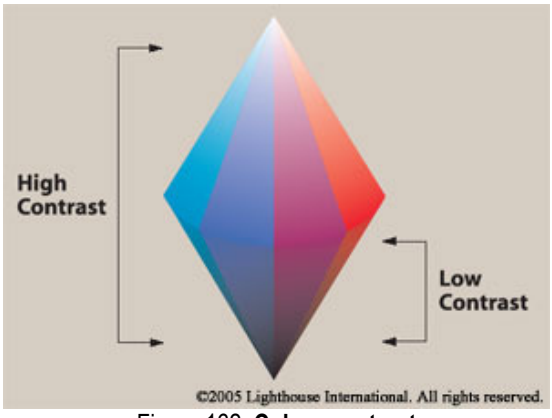


Figure 162: **Colour contrast**

2. Choosing dark colours with hues from the bottom half of this hue circle against light colours from the top half of the circle (see Figure 163). Avoiding contrasting light colours from the bottom half against dark colours from the top half. For most people with partial sight and/or congenital colour deficiencies, the lightness values of colours in the bottom half of the hue circle tends to be reduced.
3. Avoid contrasting hues from adjacent parts of the hue circle, especially if the colours do not contrast sharply in lightness.

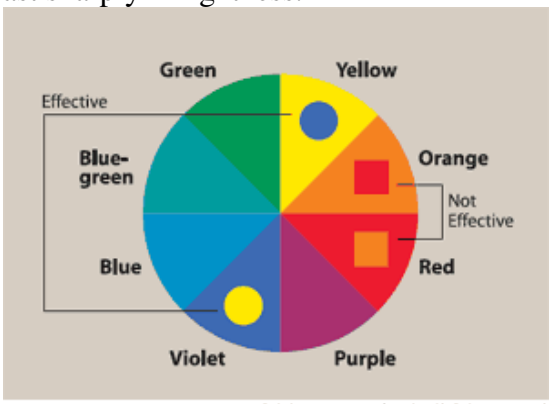


Figure 163: **Colour wheel**

Following the above guidelines, four different colour sets were chosen in order to generate correct contrast for all users that face visual impairments: blue link on white background, blue link on yellow background, orange link on blue background and green link on black background. In order to make a link more visible, the 'on focus' link artefact was designed (see Figure 164), in which a border with the text colour is shaped around the link.

A button representation was designed too in order for links to be presented as buttons in case that web portal is used in a tablet pc. In a tablet pc, the links are designed to be presented as buttons in order to be more clickable with the tablet pc's pen.

Continuing four alternative depictions were designed for buttons, containing different colour combinations that produce correct colour contrast for users that face visual impairments. The alternative design artefacts include buttons with black colour text on light gray background, white colour text on blue background, blue colour text on yellow background, and black colour text on yellow background. The alternative button representations may be selected by default portal users as appearance preferences.

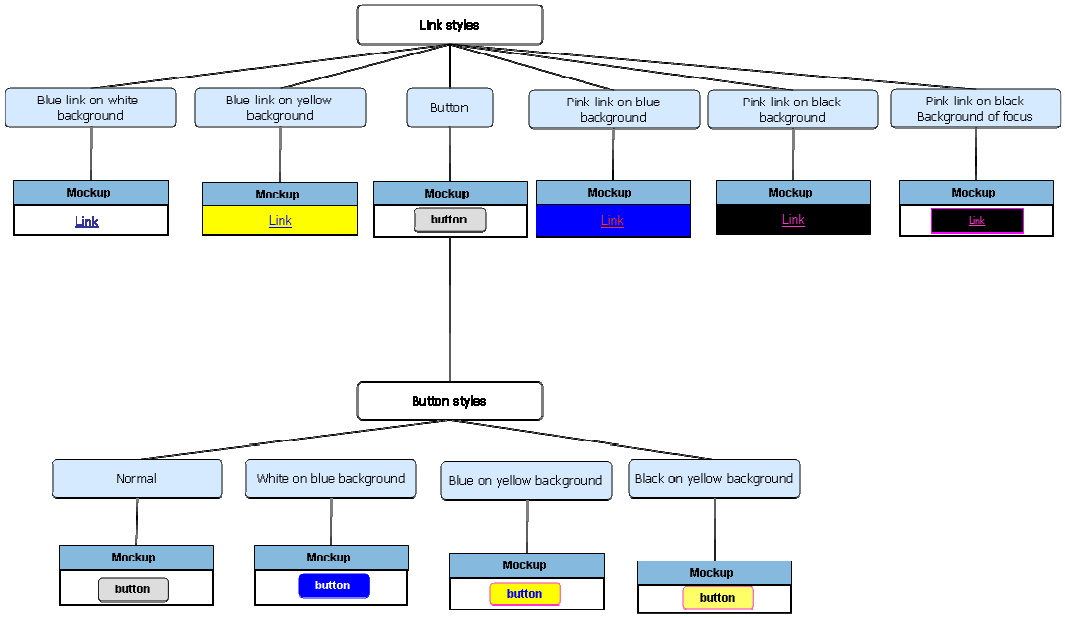


Figure 164: **Link – Button representations**

Table 38: **Design rationale of Link alternatives**

Task: Link styles						
Style:	Blue link on white background	Blue link on yellow background	Button	Light orange link on blue background	Light green link on black background	Light green link on black background on focus
Targets:	Accessibility, effectiveness	Accessibility, effectiveness	Accessibility, effectiveness	Accessibility, effectiveness	Accessibility, effectiveness	Accessibility, effectiveness
Parameters:	User (default)	User (blind, low vision colour blind)	Context (tablet pc)	User (blind, low vision colour blind)	User (blind, low vision colour blind)	User (blind, low vision colour blind) on focus
Properties:	-	-	-	-	-	-
Relationships:	Exclusive	Exclusive	Exclusive	Exclusive	Exclusive	Exclusive

Table 39: **Design rationale of Buttons alternatives**

Task: Button styles				
Style:	Black on light gray background	White on blue background	Blue on yellow background	Black on yellow background
Targets:	Accessibility, effectiveness, usability	Accessibility, effectiveness, usability	Accessibility, effectiveness, usability	Accessibility, effectiveness, usability
Parameters:	User (blind, low vision colour blind or user preference)	User (blind, low vision colour blind or user preference)	User (blind, low vision colour blind or user preference)	User (blind, low vision colour blind or user preference)
Properties:	-	-	-	-
Relationships:	Exclusive	Exclusive	Exclusive	Exclusive

Image alternatives

Images appear very often in the content of web portals. Blind or low vision users are not interested in viewing images, but only in reading the alternative text that describes

the image. In order to facilitate blind and low vision users, two design alternatives were produced, which are presented in Figure 165. The text representation of the image simply does not present the image, but only a label with the prefix 'Image:' and followed by the alternative text of the image. The second representation focusing to users with visual impairments is similar to the first, with the difference that, instead of a label, a link is included that leads to the specific image giving the ability of saving the image. In particular, a blind user may not wish to view an image, but to save it to disk and use it properly. Another design was produced that can be selected as a preference by web portal users, in which the images are represented as thumbnail bounding the size that holds on the web page. To view the image in normal size, the user may click on it. In Table 40 the design rationale of the alternative images design is presented.

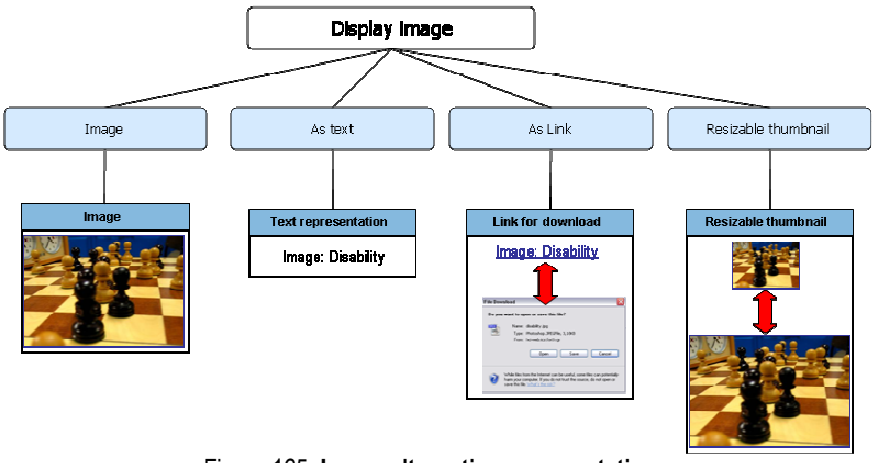


Figure 165: **Image alternative representations**

Table 40: **Design rational of the images alternatives**

Task: Display image				
Style:	Image	As text	As link	Resizable thumbnail
Targets:	-	Facilitate screen reader and low vision users in order not to be in difficulties with image viewing	Facilitate screen reader and low vision users in order not to be in difficulties with image viewing but with the capability to save or view an image	Viewing images in small size in order not to hold large size on the web page with the capability to enlarge the image to normal size when it is necessitated.
Parameters:	User(Default)	User(Blind or Low vision)	User(Blind or Low vision) and user preference)	User(preference)
Properties:	View image	Read image alternative text	Read image alternative text or and select linked named as the image alternative text to save or view the image	View image thumbnail and select it to view it in normal size
Relationships:	Exclusive	Exclusive	Exclusive	Exclusive

Textboxes alternatives

According to previous the section, the combination between text colours and background colours has to produce efficient contrast. This issue is applied to textboxes too, because when a user types text in a textbox, the web portal has to offer efficient colour contrast between the textbox background and the colour of the text

being typed. According to this issue, three alternative textboxes (on focus) were designed and rationalized except from the default textbox (see Table 41). These are light green text on black textbox background, light orange text on blue textbox background, and blue text on yellow textbox background.

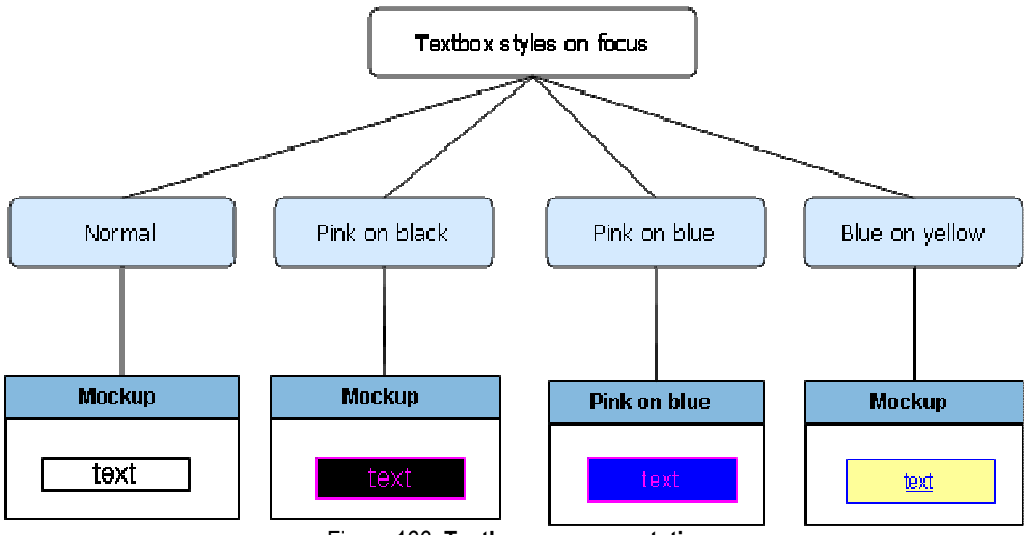


Figure 166: **Textboxes representations**

Table 41: **Design rationale of textbox representations**

Task: Textbox styles				
Style:	Normal	Light green on black	Light orange on blue	Blue on yellow
Targets:	Accessibility, effectiveness	Accessibility, effectiveness	Accessibility, effectiveness	Accessibility, effectiveness
Parameters:	User (default)	User (blind, low vision colour blind)	User (blind, low vision colour blind)	User (blind, low vision colour blind)
Properties:	-	-	-	-
Relationships:	Exclusive	Exclusive	Exclusive	Exclusive

Tabs alternatives

Web portals include information that is divided in several categories, and usually presented in separate tabs. Another tab style "Tab without graphics" is proposed here, in addition to the classic tab style view that accommodates screen readers' use for blind, low vision users or users who prefer a more linearized view of the portal (see Figure 167 and Table 42).

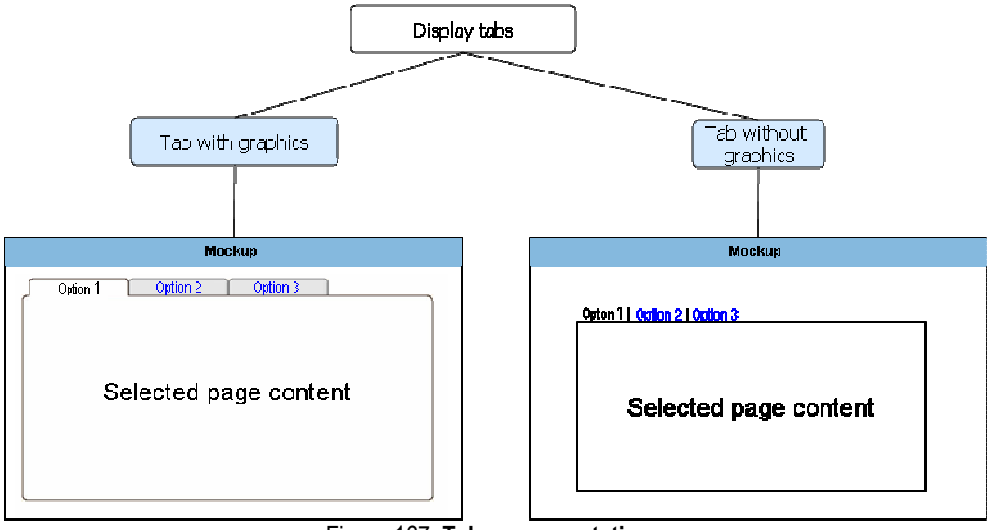


Figure 167: **Tabs representations**

Table 42: **Design rationale of tab alternative styles**

Task: Display tabs		
Style:	Tab with graphics	Tab without graphics
Targets:	Usability, flexibility	Accessibility, usability, flexibility
Parameters:	User preferences	User (Blind, Low vision)
Properties:	-	-
Relationships:	Exclusive	Exclusive

Fieldset alternatives

Another style that is used in web portals in order to group and categorize information is named "fieldset". Two alternative styles for "fieldset" were designed; a simple representation that uses only a title and a separate line, and another representation that uses graphics (see Figure 168, Table 43). The simple style may be used by blind or low vision users to simplify the graphical representation.

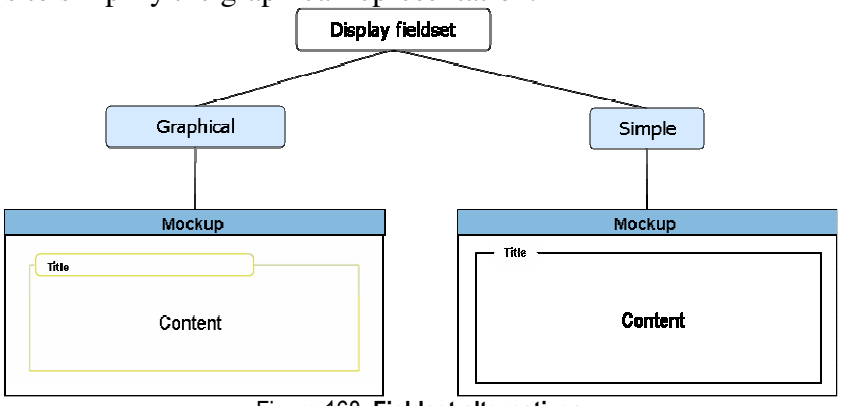


Figure 168: **Fieldset alternatives**

Table 43: **Design rationale of fieldset styles**

Task: Display fieldset		
Style:	Graphical	Simple
Targets:	Usability, flexibility	Accessibility, usability, flexibility

Parameters:	User preferences	User (Blind, Low vision)
Properties:	-	-
Relationships:	Exclusive	Exclusive

Tables alternatives

Tables are used are used frequently in a web portal to structure data, but some of them cause problems for screen reading software (used by users who are blind or low vision), since screen readers tend to read across the screen in a way that runs all of the text on a line together. If an entry in a cell occupies more that one line, the first line of each cell would be read, then the second, etc. Solutions since today create a text-only version of the page that does not offer an equivalent representation of the table to people with disabilities.

(Vanderheiden, 1996) proposes that solution strategies have to contain two attributes that will help people using screen readers access information in tables. These are the AXES and AXIS attributes that would be associated with each cell in the table. Thus, if the browser makes these attributes available for each entry in the table, the row and column information as well as the cell's data contents would be available to the screen reader.

The alternative designs for tables produced based on the proposed solution are presented in Figure 169 to Figure 175. Table 44 presents the design rationale of the alternative styles of the table.

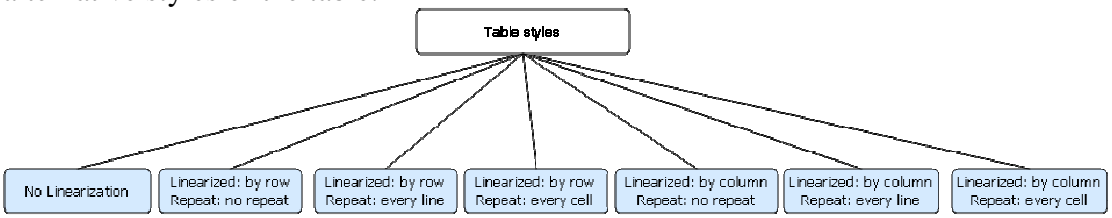


Figure 169: **Table alternatives**

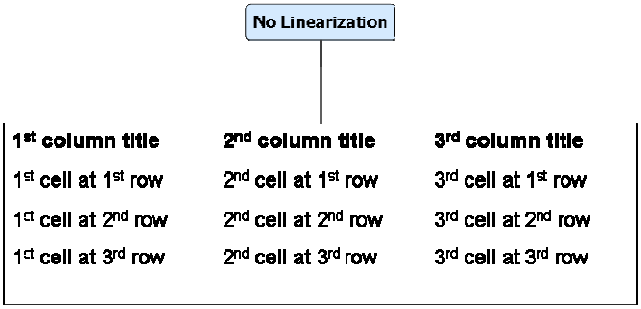


Figure 170: **No table linearization**

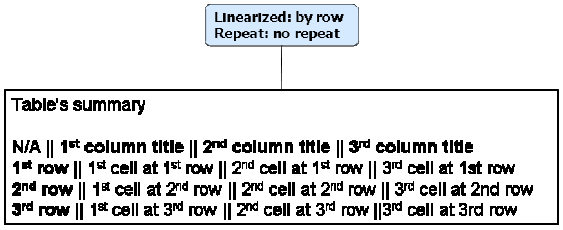


Figure 171: **Table linearization by row (no headings repeat)**

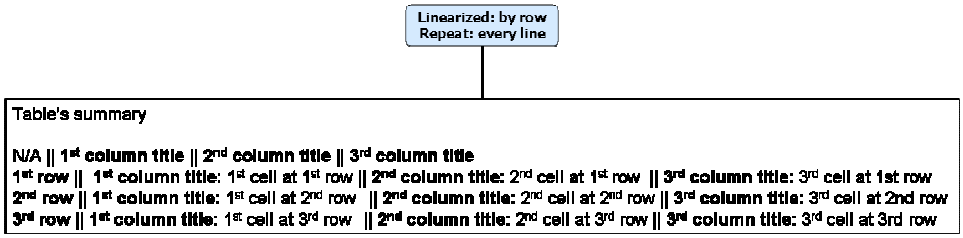


Figure 172: **Table linearization by row (repeat headings every line)**

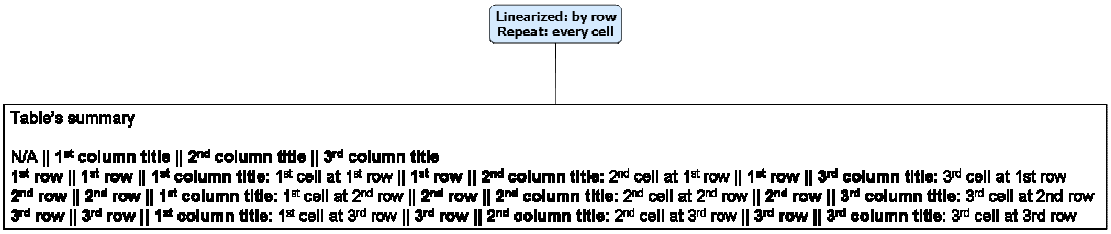


Figure 173: **Table linearization by row (repeat headings every cell)**

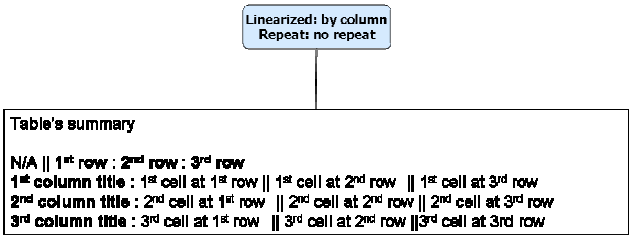


Figure 174: **Table linearization by column (no headings repeat)**

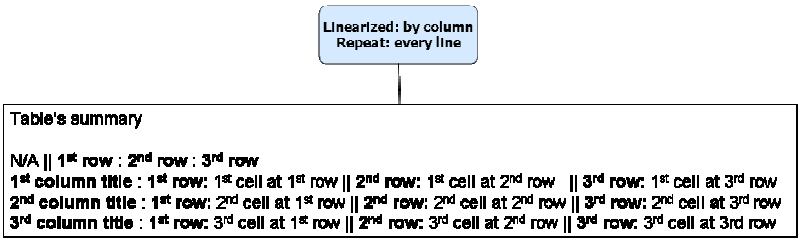


Figure 175: **Table linearization by column (repeat headings every line)**

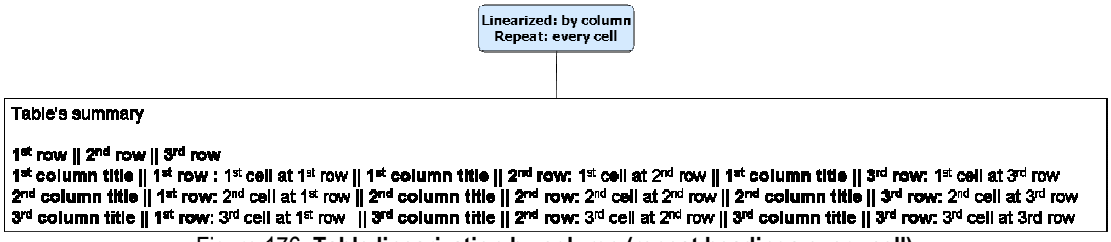


Figure 176: **Table linearization by column (repeat headings every cell)**

Table 44: **Design rationale of table styles**

Task: Table alternatives		
Style:	No linearization	Other styles
Targets:	Usability, flexibility	Accessibility, usability, flexibility
Parameters:	User preferences	User (Blind, Low vision), Preferences
Properties:	-	-
Relationships:	Exclusive	Exclusive

Charts alternatives

Most Web portals today include statistics that are presented frequently in charts. Five alternative artefacts were designed for charts in order to support different types of users. As shown in Figure 177 and Table 45, the normal view of a chart is rendered using a standard palette. On the other hand, considering that a measurable number of users face colour vision impairments, three alternative charts with the applicable colour palettes for each impairment were designed. Finally, for low vision or blind users the chart is represented as a table of values that will be easily read by a screen reader.

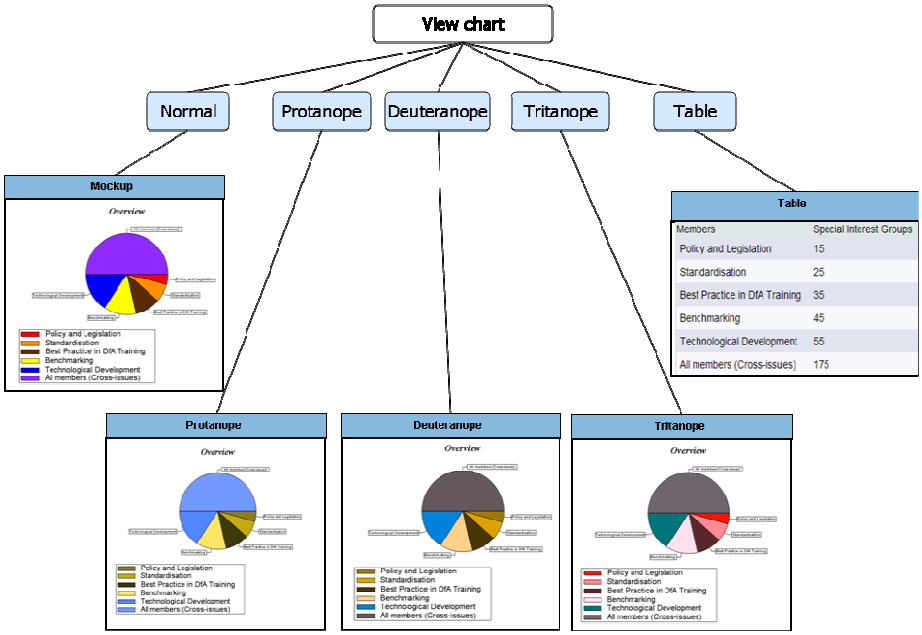


Figure 177: **Charts alternative representations**

Table 45: **Design rationale of chart representations**

Task: View charts					
Style:	Normal	Protanope	Deuteranope	Tritanope	Table
Targets:	Usability, flexibility	Accessibility, usability, flexibility	Accessibility, usability, flexibility	Accessibility, usability, flexibility	Accessibility, usability, flexibility
Parameters:	User preferences	User (protanope impairment)	User (deuteranope impairment)	User (tritanope impairment)	User (Low vision)
Properties:	-	-	-	-	-
Relationships:	Exclusive	Exclusive	Exclusive	Exclusive	Exclusive

Module options alternatives

Web portal provide several modules to users in order to achieve their goals through the portal. These powerful tools contain a number of options that can be presented alternatively as designed in Figure 178 to Figure 180.

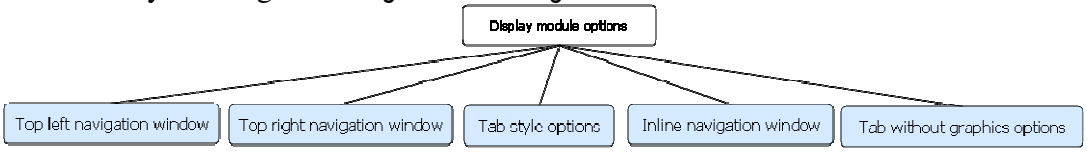


Figure 178: **Module options alternative styles**

"Navigation window" representations group options vertically in a window which is positioned on the top left or top right of the module's content, respectively the "Top left navigation window" or "Top right navigation window" style.

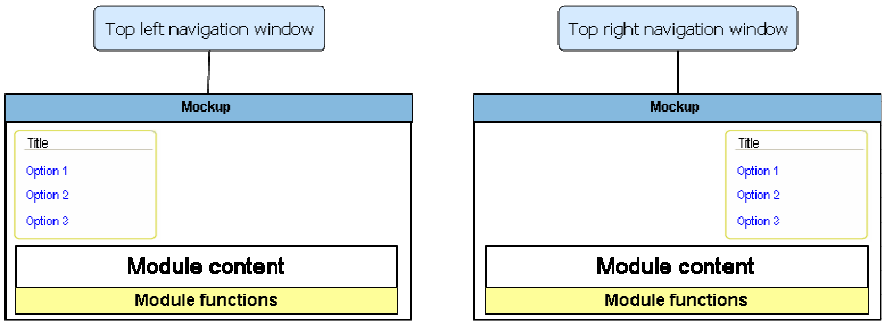


Figure 179: **Top left navigation window – Top right navigation window**

In the "Inline navigation window", options are grouped and positioned at the left of the module content, whereas in the "Tab style" form they are grouped horizontally at the top of the module content and are presented as tabs. All the alternative styles are used in the manner of usability and flexibility, as each user prefers (see Table 46) apart from the "Tab without graphics" style that is used in order to accommodate screen readers and as a result improves usability, flexibility for blind or low vision users.

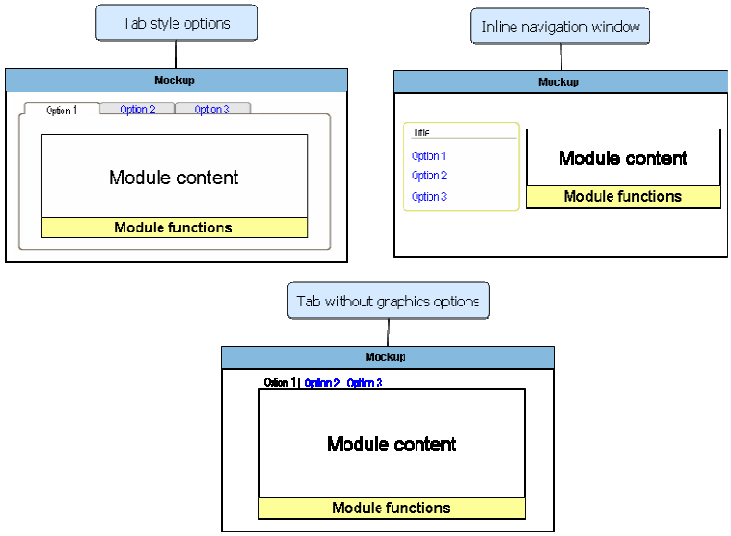


Figure 180: **Tab styles options (Inline navigation window vs. tabs without graphics)**

Table 46: **Design rationale of module options alternative styles**

Task: Display module options					
Style:	Top right navigation window	Tab style options	Top left navigation window	Inline navigation window	Tab without graphics options
Targets:	Usability, flexibility	Usability, flexibility	Usability, flexibility	Usability, flexibility	Accessibility, usability, flexibility, accommodate screen reader
Parameters:	User preferences	User preferences	User preferences	User preferences	User (Blind, Low vision)
Properties:	-	-	-	-	-
Relationships:	Exclusive	Exclusive	Exclusive	Exclusive	Exclusive

Functions alternatives

Each page of a web portal usually contains a number of functions. According to user web expertise, three alternative function artefacts were designed and are presented in Figure 181, along with their design rational in Table 47. The first alternative artefact presents only the function elements without any further guidance, addressing expert users.

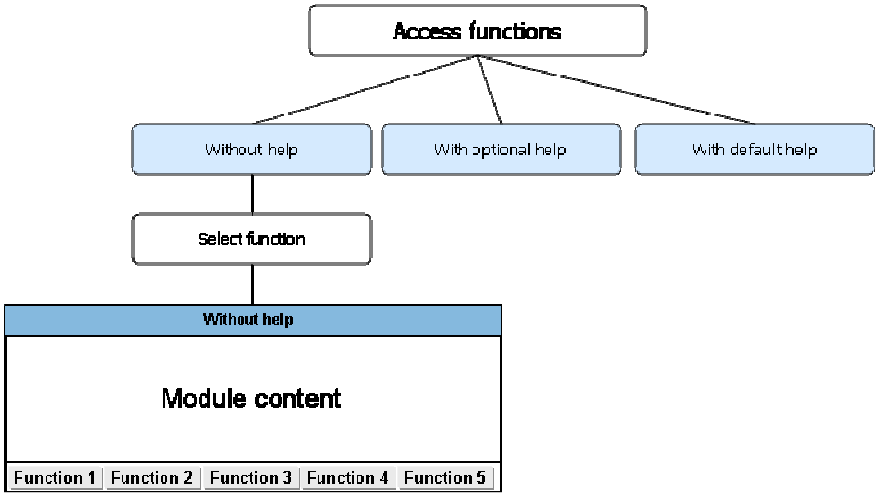


Figure 181: **Functions alternatives**

Figure 182 shows the alternative artifact for the novice user, where a systematically description is offered next to each function element to guide the user.

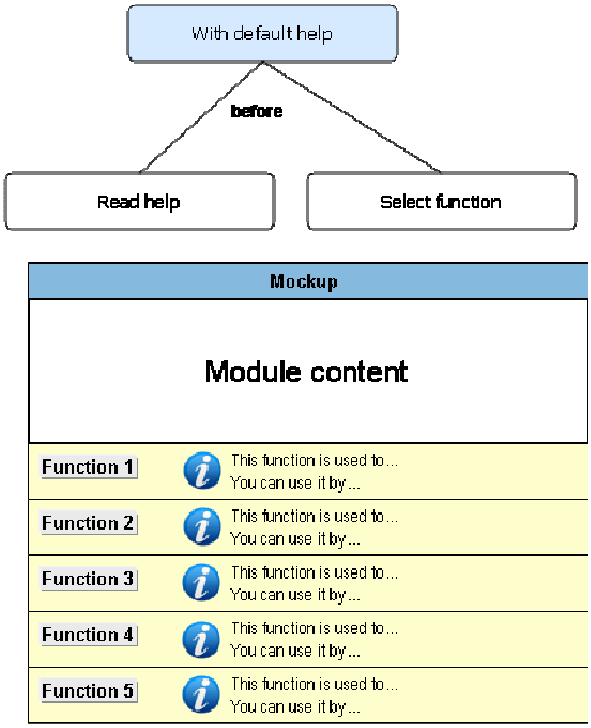


Figure 182: **Functions with default help alternative**

As shown in Figure 183, for moderate user an intermediate solution was designed, where the user may optionally select to view guidance details for all the functions.

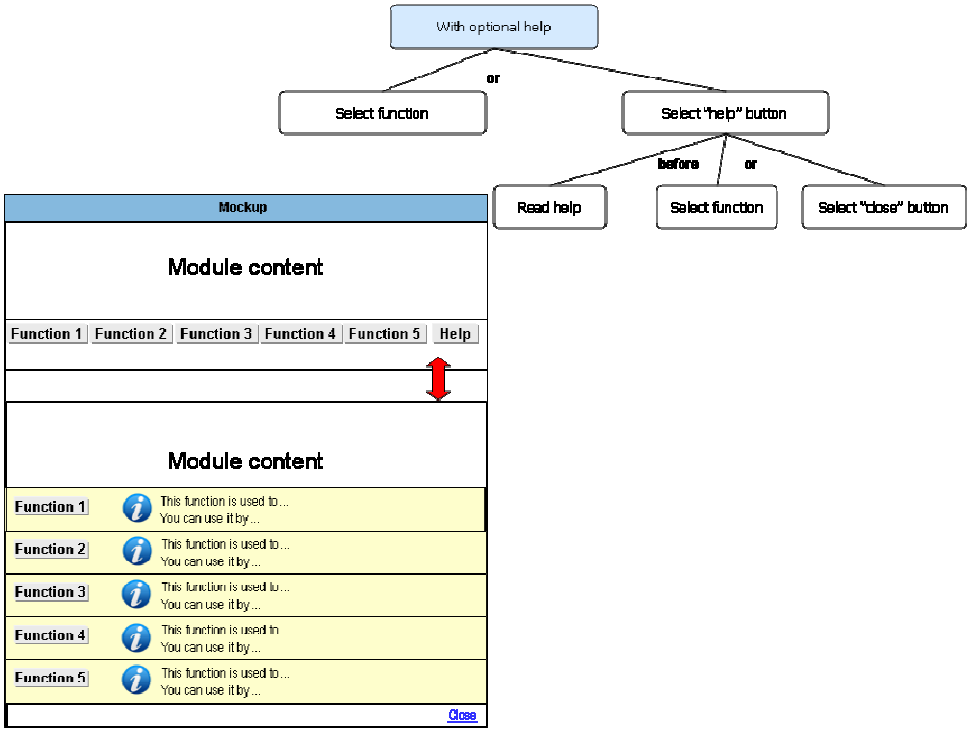


Figure 183: **Function with optional help alternative**

Table 47: **Design rational of functions alternatives**

Task: Access functions			
Style:	Without help	With optional help	With default help
Targets:	Speed, usability	Optional guidance, usability	Default guidance, usability, efficiency
Parameters:	User (expert)	User (moderate)	User (novice)
Properties:	Use function	Navigate to help optionally, use function next	Read help first, use function next
Relationships:	Exclusive	Exclusive	Exclusive

Paging representations - Single items representations

Results in a Web portal arise from a search query or by browsing a category constituted by multiple subcategories, and are presented usually in lists. Paging mechanisms are applied in order to break long lists of results into a number of single page sub-lists. More specifically, paging separates results in groups of specific numbered "pages", and offers to users a mechanism in order to navigate through different "pages". Alternative paging representations were designed and are presented in Figure 184.

The "Link button" paging gives multiple results in a page; it offers two links leading to the first and the last page of the results respectively, a list of numbered links leading to the specific number of the page, and finally a label that defines the total number of the available results' pages.

The "Drop down paging" alternatives with or without automatic paging redirection present multiple results; it offers four link buttons that lead to the first, last previous and next page respectively. To navigate to a specific page using the "drop down paging with automatic redirection", the user has to select only the specific page

from the drop down when using the "drop down paging with automatic redirection", but when using the "drop down paging with manual redirection" the button display is pressed too. In the drop down's text the total number of pages is also displayed.

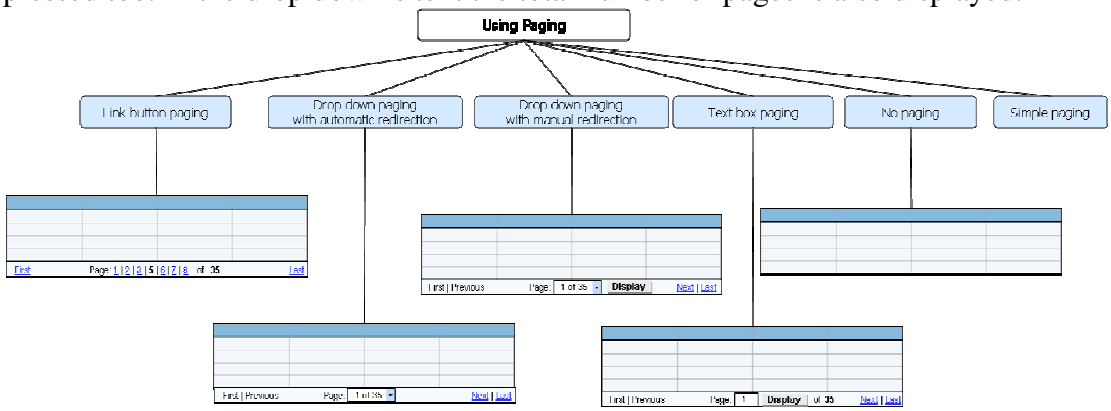


Figure 184: **Paging alternatives**

"Textbox paging" looks like the "drop down paging with manual redirection", with the difference that when the user wishes to navigate to a specific page, the number of the page in the textbox is typed instead of being selected from the dropdown, and then the button "display" is pressed. The "no paging" alternative simply presents all the results in a page, and consequently there is no need for a navigation mechanism through the pages. The last two paging options are based on the idea that a user may wish to navigate through the results one by one, thus there are only two links that lead to the previous and next result. In the "graphical navigation" option, the user interacts with the left and right arrows, but in the "no graphics navigation", links named "previous" and "next" are used.

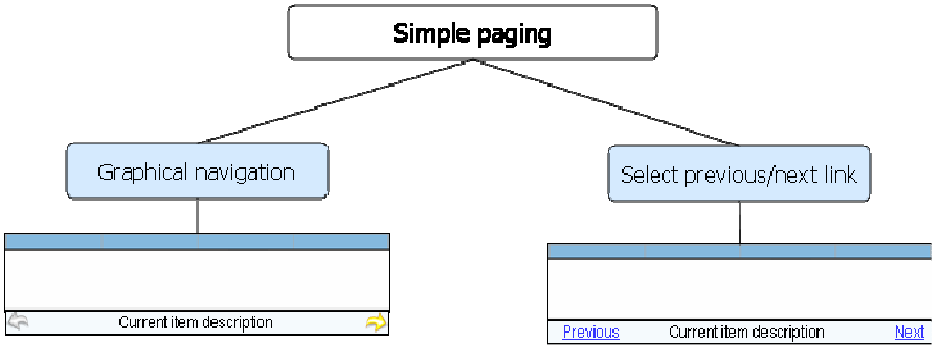


Figure 185: **Simple paging alternatives**

The design rationale of the alternative paging designs is recorded in Table 48 and Table 49. All the alternatives can be selected by each user, but some of them have been designed for specific user categories. The "link button paging" is targeted to motor impaired users, as it includes only links that can be scanned easily and entail only link selection (does not entail text entry). The "drop down paging with automatic redirection" is intended to be used by expert users that will have java script activated, and with a dropdown selection will be redirected to the desired page. The "drop down paging with manual redirection" and "textbox paging" are designed to be used by blind or low vision users, because they include minimum scan reading information and are simple in use. The "no paging" and "simple paging" with and without graphics are designed in order to serve individual user preferences.

Table 48: **Design rational of paging alternatives 1/2**

Task: Using paging (1/2)				
Style:	Link button paging	Drop down paging with automatic redirection	Drop down paging with manual redirection	Textbox paging
Targets:	Facilitate scanning techniques, effectiveness	Speed, efficiency, usability	Limited screen reading time, easiness	Limited screen reading time, easiness
Parameters:	User (Motor impaired), Default paging	User(Expert)	User (Blind or Low vision)	User (Blind or Low vision)
Properties:	Use 'first', 'last' links and numbered links to navigate through pages	Use 'first', 'previous', 'next', 'last' links and drop down to navigate through pages	Use 'first', 'previous', 'next', 'last' links and drop down in combination with the 'display' button to navigate through pages	Use 'first', 'previous', 'next', 'last' links and type into textbox in combination with the 'display' button to navigate through pages
Relationships:	Exclusive	Exclusive	Exclusive	Exclusive

Table 49: **Design rational of paging alternatives 2/2**

Task: Using paging (2/2)			
Style:	No paging	Simple paging without graphics	Simple paging with graphics
Targets:	Quick view of all the results	One by one results presentation	One by one results presentation and nice aesthetic result
Parameters:	User preferences	User preferences	User preferences
Properties:	-	Use 'previous, 'next' links to navigate through results	Use 'previous, 'next' arrow icons to navigate through results
Relationships:	Exclusive	Exclusive	Exclusive

Text entry alternatives

Text entry in a web portal typically includes the use of a text area and a keyboard. In this section alternative designs for typing text using a virtual keyboard are presented, motivated by scanning techniques for users with limited motor functionality of upper limbs, as well as contexts of use in which the keyboard is absent (e.g., tablet pc). Web virtual keyboard alternative designs which are presented in Figure 186 to Figure 189 are based on the AUK virtual keyboard presented in (Mourouzis et al., 2007), which is similar to a multi-tier 3x3 menu system. Each 3x3 grid has eight virtual *character keys* and a *menu key* for entering or exiting alternative menus. Four *basic menus* can be interchanged sequentially. Additionally, there are five *secondary menus*: *numerals* (4.2), *special characters* (4.4), *brackets* (4.3.1 – 4.3.2), *formatting options* (4.9), *selection options* (4.6), and *numeric operators* (4.8) that are presented in Figure 186.

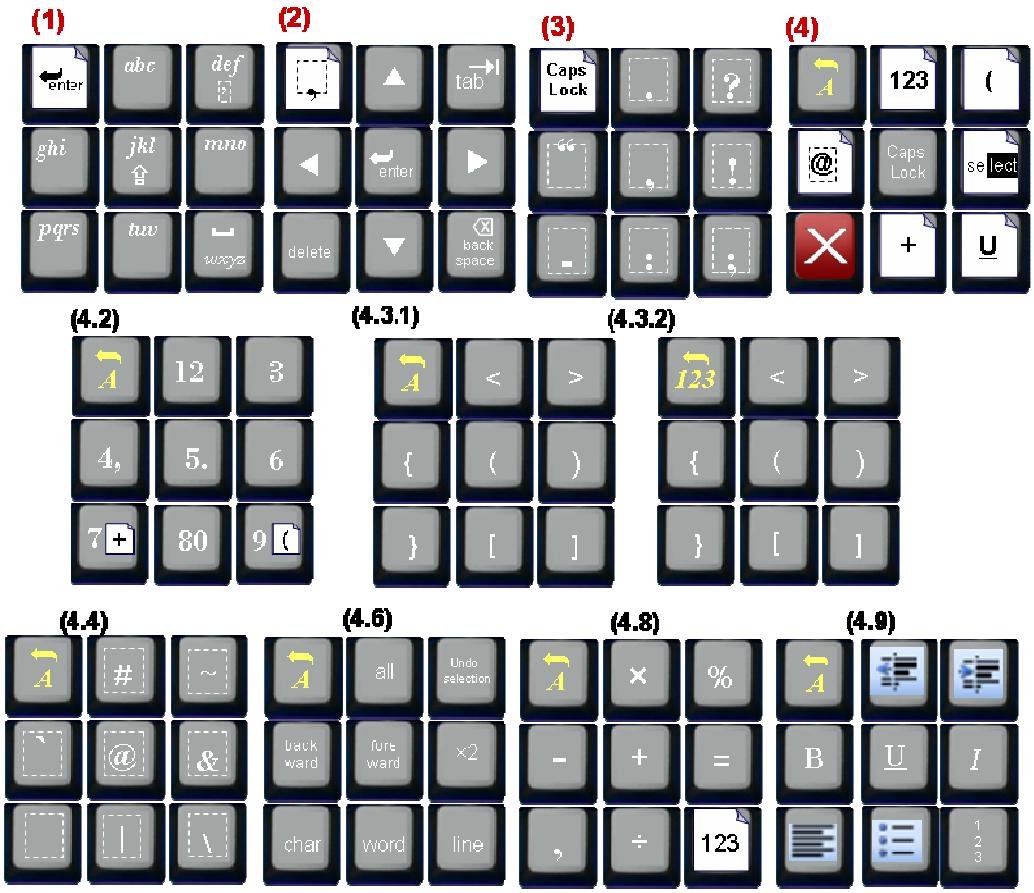


Figure 186: **Web-based Virtual Keyboard (overview)**

Entry modes and menu layouts

– The presented menu structure can be employed with various input methods. In particular, the Web-based version of AUK supports text entry for: 9-keys (e.g., phone keypads), *2-keys* (e.g., foot pedals); *3-kesy* (e.g., three switches); *5-keys* (e.g., gamepad). Basically, in this category, a *menu cursor* travels, automatically or manually, across each 3x3 menu, always starting from a *home position*. Once the desired input element is reached, a *select button* is used for "tapping" the onscreen key.

As suggested in (Mourouzis et al., 2007), different A-Z letters layouts have been produced for novice, moderate and expert users, including a

- A standard phone-like layout (see Figure 187). This is a familiar and easy to remember layout.

Figure 187: **Standard phone-like layout**

- A slightly modified phone-like layout (Figure 188), in which letters within each key are rearranged according to their frequency, in order to offer higher performance rates.

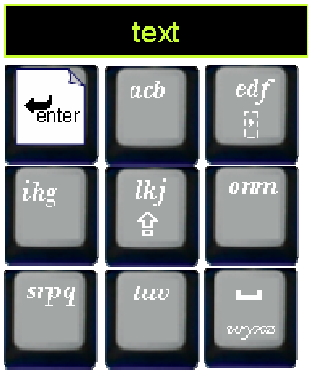


Figure 188: **Frequency based phone-like layout**

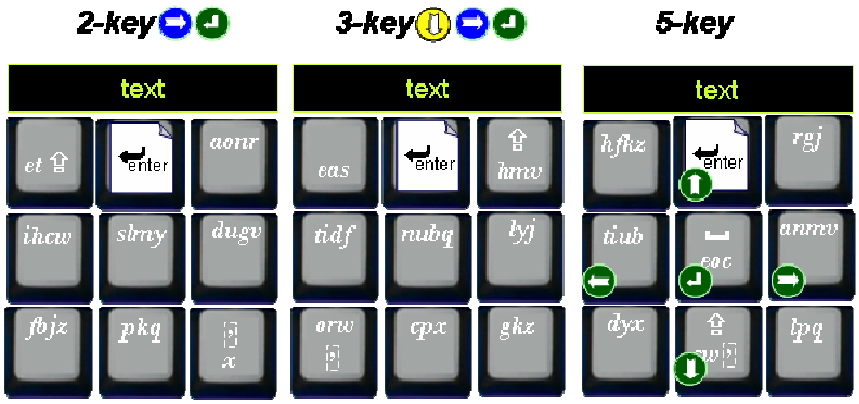


Figure 189: **Optimised layouts for 2-, 3- and 5- key text entry**

Style:	Standard phone-like layout	Frequency based phone-like layout
Targets:	Accessibility, usability	Accessibility, usability
Parameters:	User (motor impaired) or Context of use (Tablet pc), Low or moderate expertise	User (motor impaired) or Context of use (Tablet pc), Low or moderate expertise
Properties:	-	-
Relationships:	Exclusive	Exclusive

Table 50: **Design rationale of text entry alternatives**

Formatted text entry

In order for a web user to type a formatted text, usually a web editor is offered. Three customizations of the editor were designed in order to serve different web users' expertise. The first design, targeted to expert users, includes full functionality, as shown in Figure 190. The functions that are supported by this design artifact are: font customizations, alignment functions, mark text functions, copy-paste and undo-redo functions, date - time insertions and table creation. These tools offer many capabilities to expert users in order to construct a well structured and formatted text. The representation for the moderate user contains a subset of the functions of the design for expert user. Similarly, the design for novice users constitutes of a subnet of the moderate design functions, including simply fonts formatting and alignment functions.

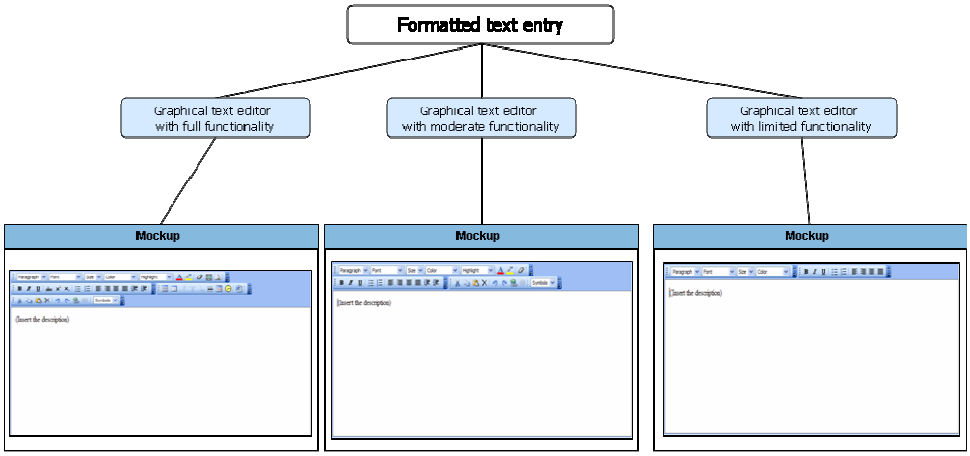


Figure 190: **Editor Alternatives**

Table 51: **Design rational of the editor alternatives**

Task: Formatted text entry			
Style:	Graphical text editor with full functionality	Graphical text editor with moderate functionality	Graphical editor with limited functionality
Targets:	Effectiveness, usability	Effectiveness, usability	Simplicity, effectiveness, usability
Parameters:	User (expert)	User (moderate)	User (novice)
Properties:	Use available functions	Use available functions	Use available functions
Relationships:	Exclusive	Exclusive	Exclusive

Date entry alternatives

Date entry constitutes a frequent task for web users in case of registering an event to the web portal or in case of searching for data based on dates. Five alternatives were designed for date entry and are presented in Figure 191 and Figure 192. As it is recorded in

Table 26, the first design includes three drop downs, where the user has to select three items year, month, day using drop downs. These dropdowns are constructed in such a way that when a user selects a specific year and month, the appropriate calculations are made based on leap years and number of days that each month includes, so that only the valid days are placed in the days' dropdown. This option is targeted to novice

users, because of its simplicity and error prevention mechanism implemented in Java Script.

The second design looks like the first, with the difference that the validation of the user input is made manually. The user after the selection of the appropriate values has to push the button 'ok' in order to validate the selected date. In case of a mistake, an error message appears in red, guiding the user to correct the mistakes. This design artefact focuses on user with visual impairments, since it does not require the use of Java Script, and therefore is, for example, compatible with screen readers.

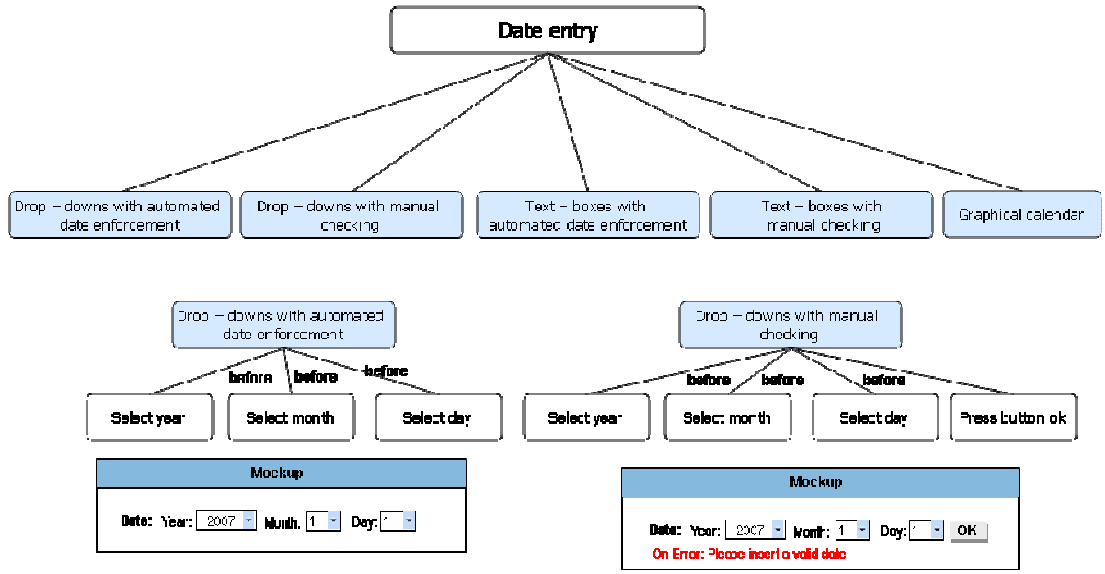


Figure 191: **Date entry alternatives (1/2)**

As shown in Figure 192, in textboxes design with automated date enforcement design the user has to type year, month and day in the textboxes. The validation of the date takes place when the user tries to complete the particular action. This date input alternative is designed for expert web users to improve speed and effectiveness. Another similar design intends to be used by users with visual impairments who are web experts. Simply, in this design the user has to push the button 'ok' to validate the date inserted. For moderate and motor impaired users, another design was produced that offers a virtual calendar from where the user has monthly previews, may navigate through years and months using two dropdowns, and can select a date by clinking on it when the appropriate date is met. This design is considered as suitable for motor impaired users because it does not necessitate much user input, and is suitable for moderate users too because it offers a graphical representation of dates that is more familiar users.

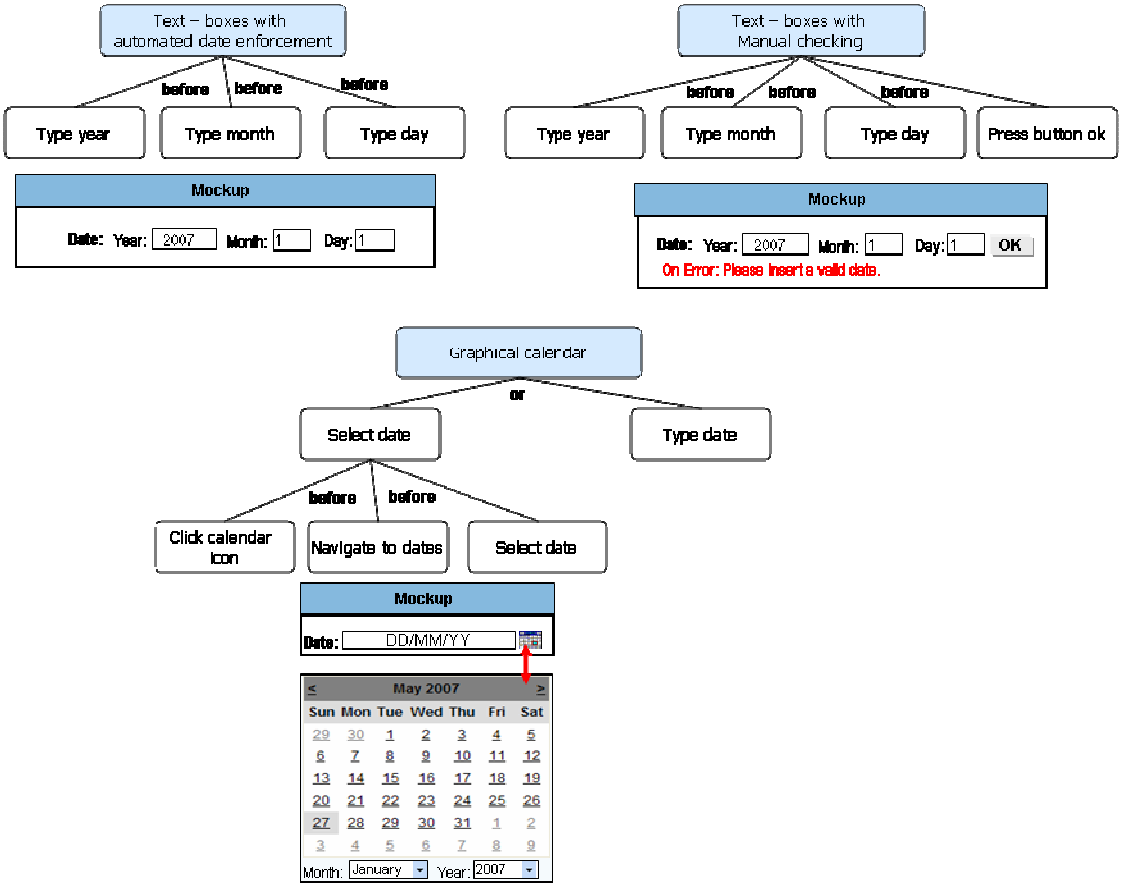


Figure 192: **Date entry alternatives (2/2)**

Table 52: **Design rationale of the date entry alternatives**

Task: Date entry					
Style:	Drop downs with automated date enforcement	Drop down with manual checking	Text boxes with automated date enforcement	Text boxes with manual checking	Graphical calendar
Targets:	Simplicity, error prevention easiness	Facilitate screen reader, effectiveness	Speed, effectiveness	Facilitate screen reader, effectiveness	Limited necessity of user input, simplicity
Parameters:	User (novice)	User (Blind or Low vision)	User (expert)	User ((Blind or Low vision) and expert)	User (Moderate, motor impaired)
Properties:	Select year first, month next and day at last	Select year first, month next, day next and press button 'ok' at last	Type year first, month next and day at last	Type year first, month next, day next and press button 'ok' at last	Type day first, '/' next, month next, '/' next, year at last or click on 'calendar' icon and select date on the virtual calendar
Relationships:	Exclusive	Exclusive	Exclusive	Exclusive	Exclusive

Upload files

Uploading files constitutes a frequently used function for web users. As shown in Figure 193 and Table 51, three alternative designs were produced towards expert, moderate and novice user for uploading and deleting files.

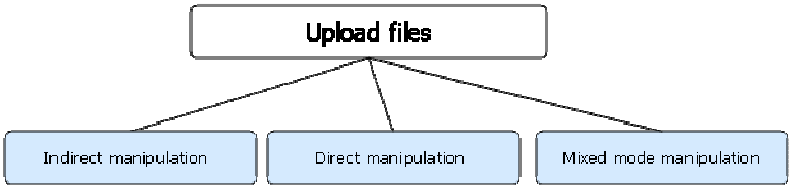


Figure 193: **Upload files alternatives**

As in Figure 194, a novice user in order to upload a file has to complete several simple steps; firstly, the user has to press the button ‘add new’, then the interface changes and the user has to complete three simple steps, browse and select a file, type a title and push the button ‘upload’. Next, a progress bar of the file upload time appears. To delete an uploaded file, the user has to press the button ‘delete’ and then to check the files to be deleted and press again the ‘delete’ button. The described design is targeted to novice users because it contains simple and detailed steps.

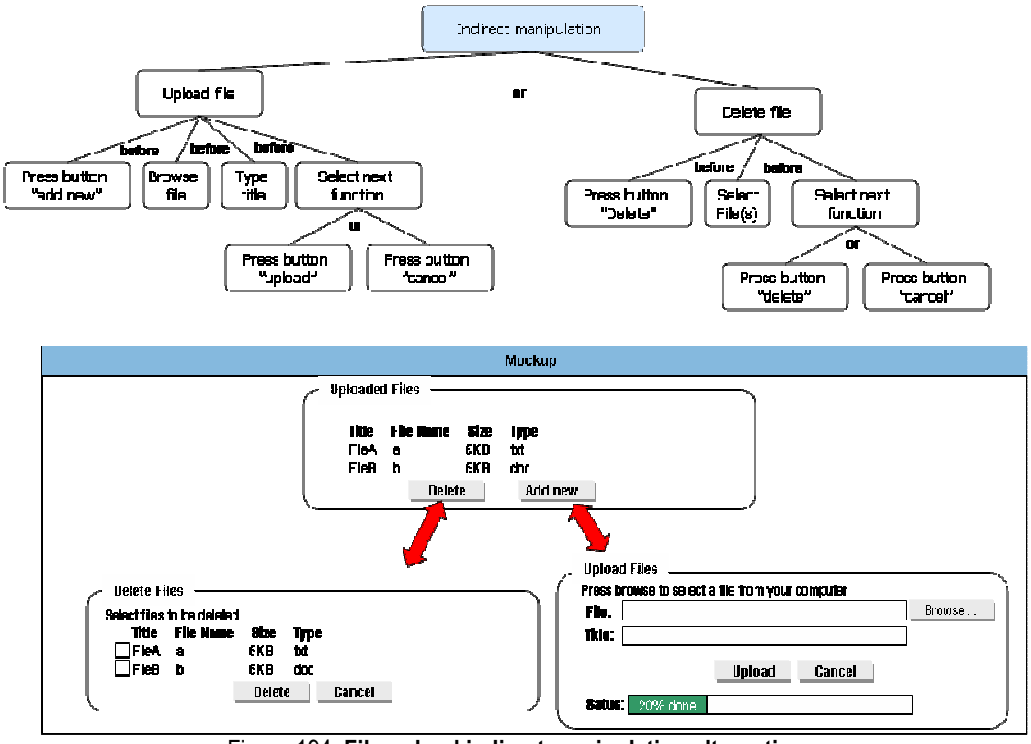
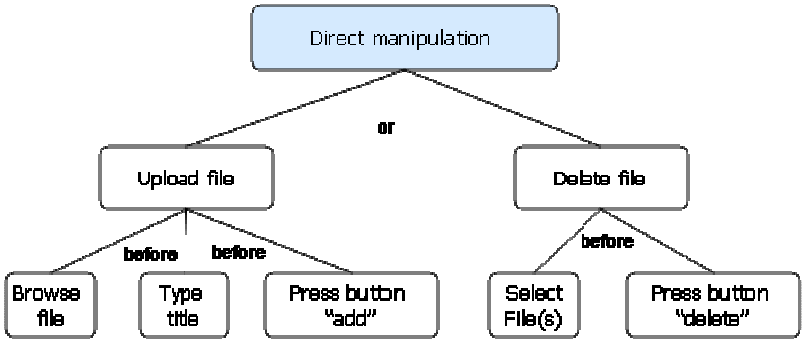


Figure 194: **File upload indirect manipulation alternative**

The direct manipulation style is designed for expert users. All the functions are lying in a single interface in order to be accessed by the user quickly and effectively. The user has only to browse and select a file, type a title and press the button ’add’ in order to upload a file. On the other hand, in order to delete a file, the user has only to select the file or files and press the button ‘delete’.

For moderate users, an intermediate design between the novice and expert user designs is provided that includes two interfaces: one to upload files and another to view uploaded files and delete files. The moderate user in order to upload a file has to

press the button ‘add’, then automatically a second interface appears where the user browses and selects a file, types a title and finally presses the button ‘upload’. A moderated user in order to delete a file has only to select the file or files to be deleted and then press the button ‘Delete’.



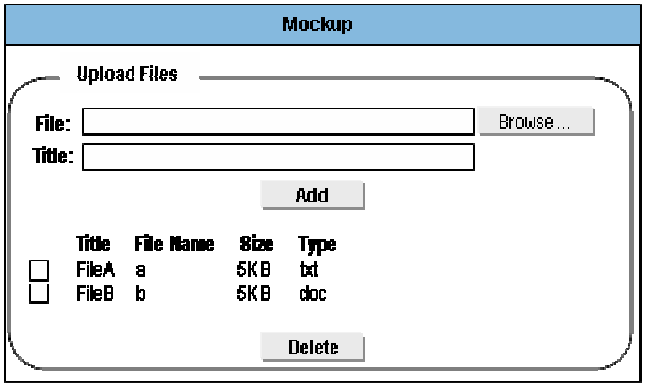


Figure 195: **File upload direct manipulation alternative**

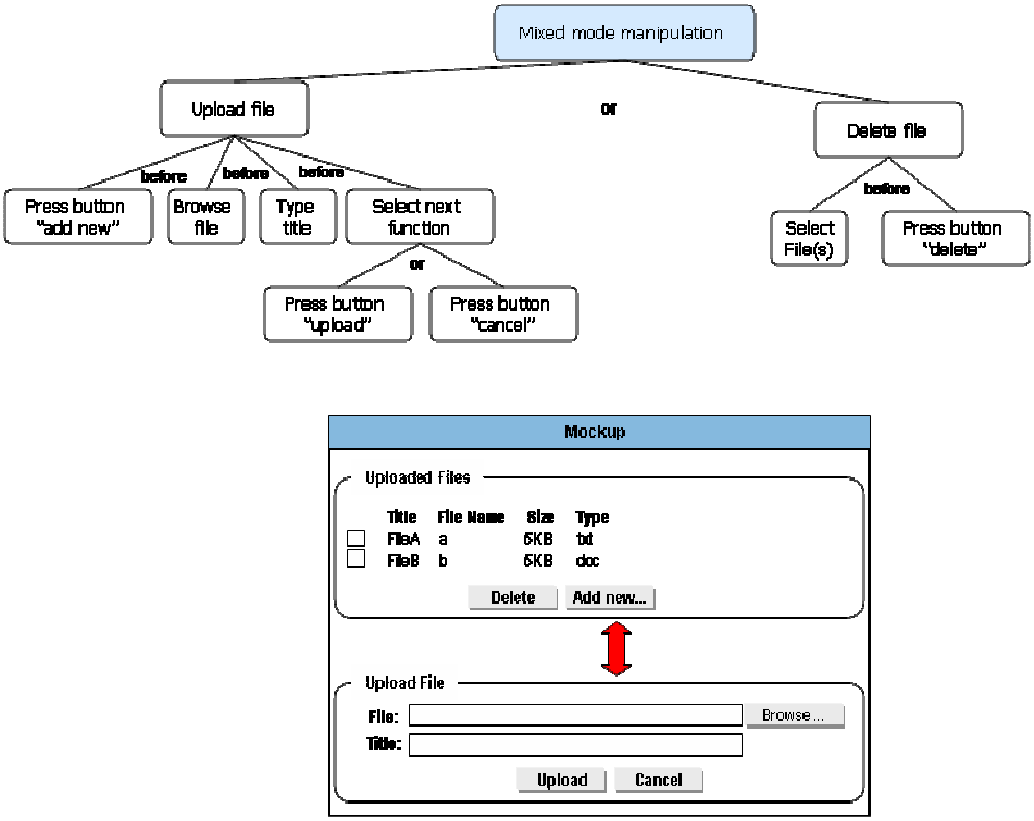


Figure 196: **File upload mixed mode manipulation alternative**

Table 53: **Design rationale of the upload files alternatives**

Task: Upload files			
Style:	Indirect manipulation	Direct manipulation	Mixed mode manipulation
Targets:	Simplicity, guided steps	Speed, effectiveness	Guides steps, effectiveness, usability
Parameters:	User (novice)	User (expert)	User (moderate)
Properties:	Upload file: Press button 'add new' first, browse file next, type title next, press button 'upload' Delete file: Press button 'delete' first, select file(s) next, press button 'delete'	Upload file: Browse file first, type title next, press button 'add Delete file: Select file(s) first, press button 'delete' next	Upload file: Press button 'add new' first, browse file next, type title next, press button 'upload' Delete file: Select file(s) first, press button 'delete' next
Relationships:	Exclusive	Exclusive	Exclusive

Upload images

Image uploading is another useful function that is similar to file uploading. Three image uploading alternatives were designed with indirect manipulation, direct manipulation and mixed mode manipulation, as shown in Figure 197.

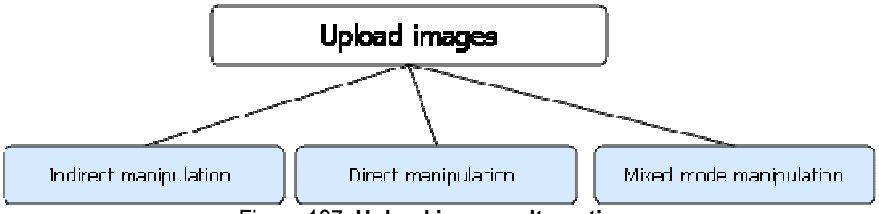


Figure 197: **Upload images alternatives**

As it appears in Figure 198 and Table 54 a user in order to upload an image has to complete the following steps:

- Press the button 'add new'
- Browse and select the desired image to upload
- Type an alternative text for the selected image
- Press the button 'upload'.

Next, the progress bar of the uploaded file is displayed.

After the completion of the above steps the user may view a thumbnail of the uploaded image in a list where other uploaded images are placed. In order to delete an image, the following steps have to be completed:

- Press the button 'delete'
- Select the images to delete
- Press the button 'delete'

The indirect manipulation style is targeted to novice user because it involves simple and guided steps.

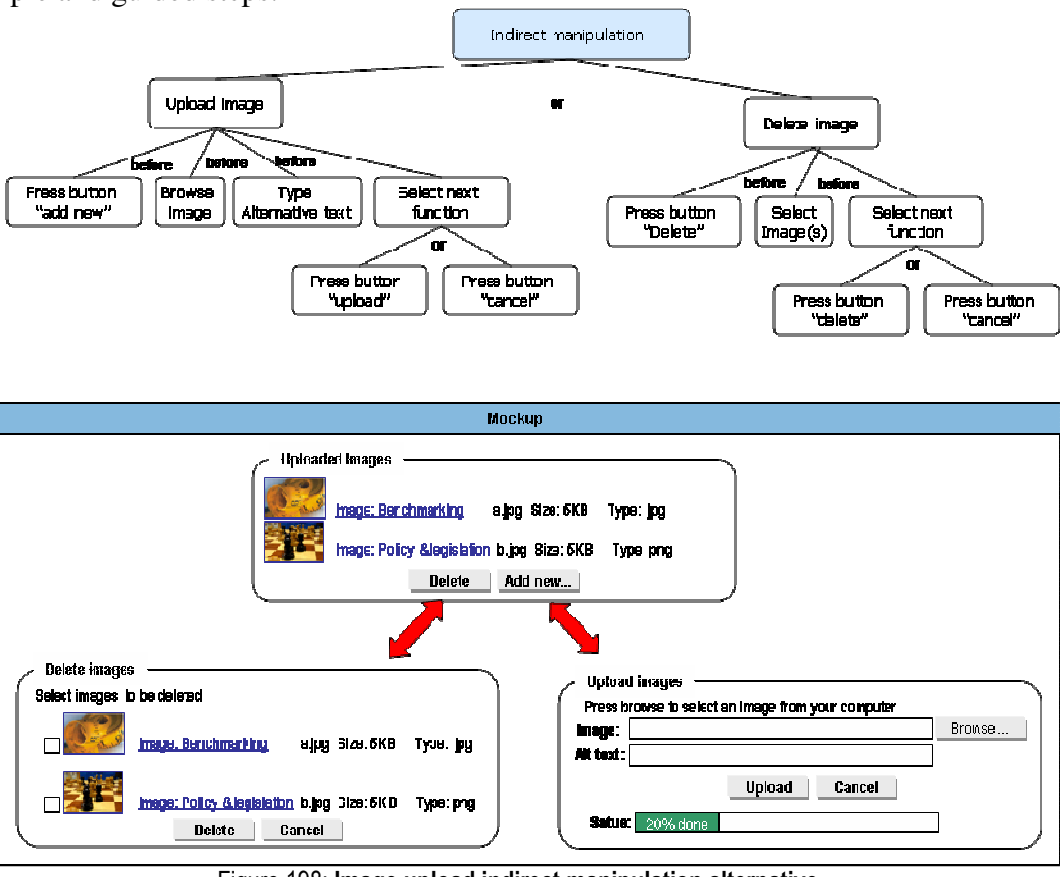
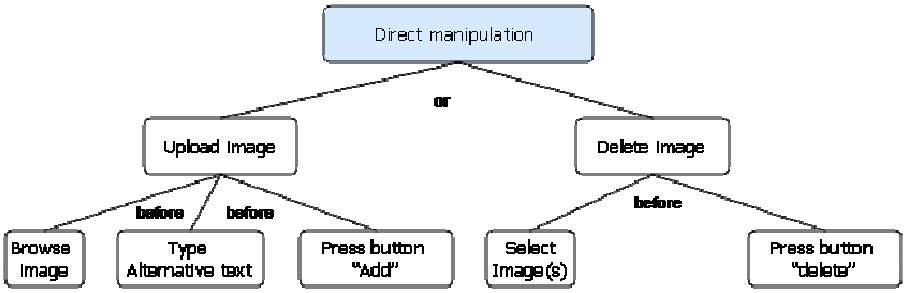


Figure 198: **Image upload indirect manipulation alternative**

On the other hand the direct manipulation style (Figure 199) was designed for expert users and includes quick and simple steps. An expert user, in order to upload an image:

- Browses and selects the desired image
- Types an alternative text for the image
- Presses the button 'Add'

To delete an uploaded image or images, the expert user simply selects from the list of images the desired image or images and presses the button 'delete'.



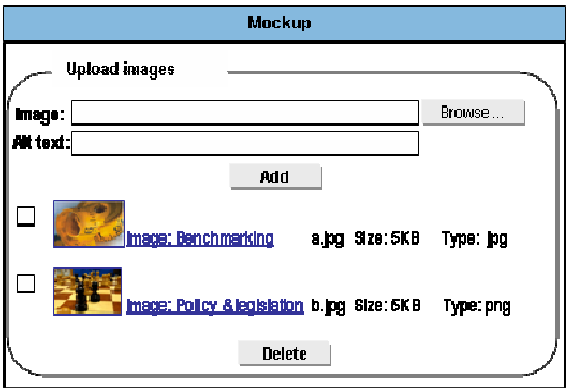


Figure 199: **Image upload direct manipulation alternative**

An intermediate solution between direct an indirect manipulation was designed for moderate users, as shown in Figure 200. A moderate user in order to upload an image has to complete the following steps:

- Press the button 'add new'
- Browse and select the desired image to upload
- Type an alternative text for the selected image
- Press the button 'upload'.

Next, the progress bar of the uploaded file is displayed.

In order to delete an image, only the following steps need to be completed:

- Select from the list the desired images to be deleted
- Press the button 'delete'.

In Table 54, the design rationale of all the alternatives is presented.

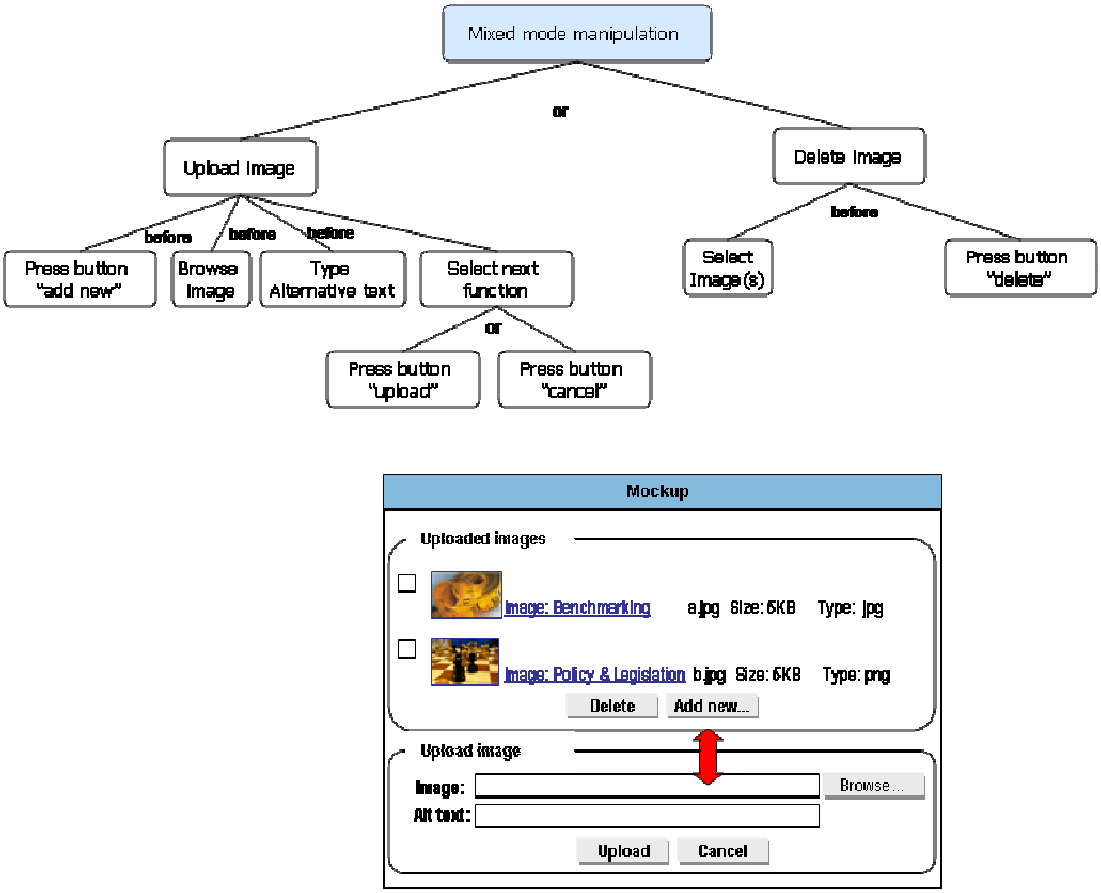


Figure 200: **Image upload mixed mode manipulation alternative**

Table 54: **Design rationale of the upload image alternatives**

Task: Upload images			
Style:	Indirect manipulation	Direct manipulation	Mixed mode manipulation
Targets:	Simplicity, guided steps	Speed, effectiveness	Guides steps, effectiveness, usability
Parameters:	User (novice)	User (expert)	User (moderate)
Properties:	Upload image: Press button 'add new' first, browse image next, type alterative text next, press button 'upload' Delete image: Press button 'delete' first, select image(s) next, press button 'delete'	Upload image: Browse image first, type alternative text next, press button 'add Delete image: Select image(s) first, press button 'delete' next	Upload image: Press button 'add new' first, browse image next, type alternative text next, press button 'upload' Delete image: Select image(s) first, press button 'delete' next
Relationships:	Exclusive	Exclusive	Exclusive

Files alternatives

Web portal usually include lists of downloadable files. As presented in Figure 201 three alternatives artefacts were designed according to user web expertise (Table 55). For novice users, the link to download a file is presented along with an estimation of

the download time, whereas for moderate users the link is accompanied with the file size. Finally, the files representation for expert users consists of the link to download the file along with the file name, the file size and the file type.

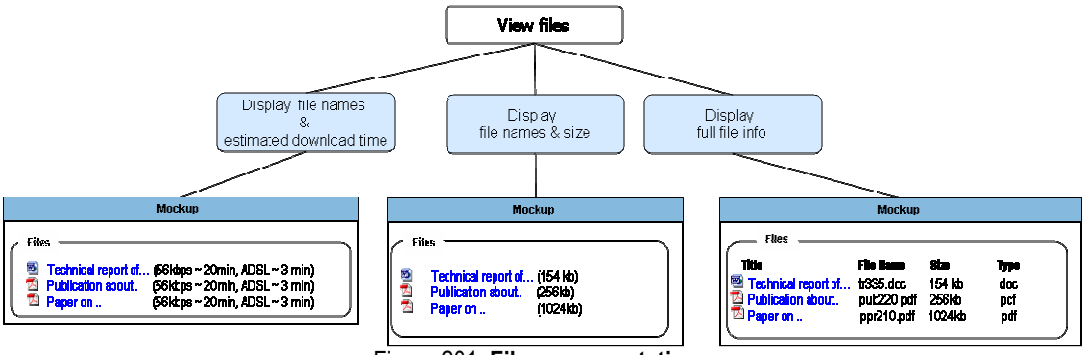


Figure 201: **Files representations**

Table 55: **Design rationale of files representations**

Task: View files			
Style:	Display file names & estimated download time	Display file names & size	Display full file info
Targets:	Usability, flexibility	Usability, flexibility	Usability, flexibility
Parameters:	User (novice)	User (moderate)	User (expert)
Properties:	-	-	-
Relationships:	Exclusive	Exclusive	Exclusive

Images alternatives

Web portals include image lists too. The alternative artefacts that were designed for images are similar to the designs for file downloading presented previously, but include two additional alternative representations in which the images are presented as a slide show (see from Figure 202 to Figure 205).

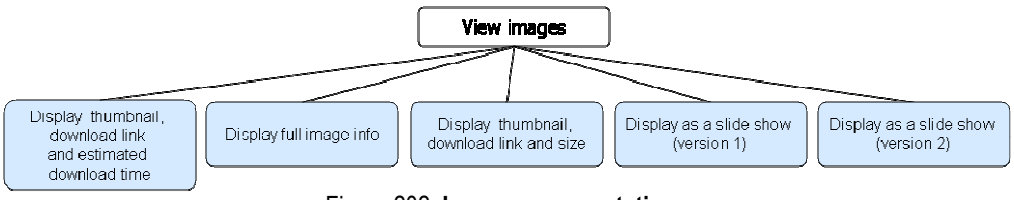


Figure 202: **Images representations**

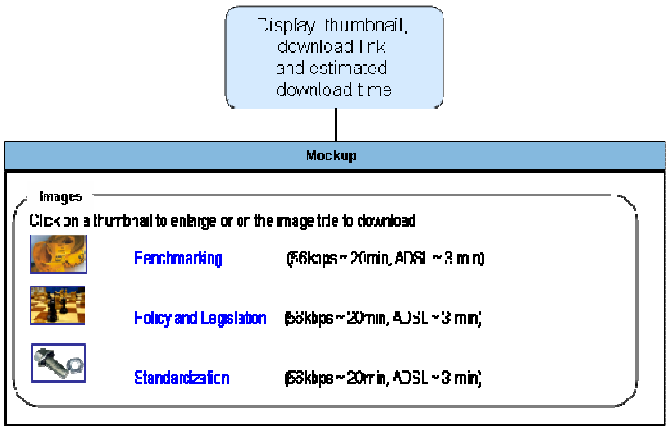


Figure 203: **Display thumbnail, download link and estimated download time**

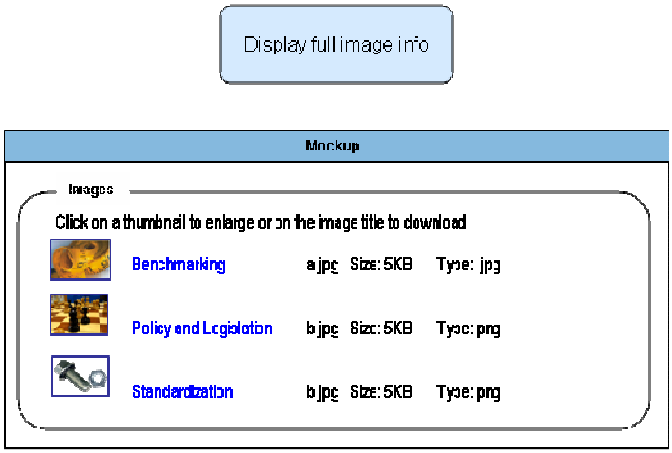


Figure 204: **Display full image info**

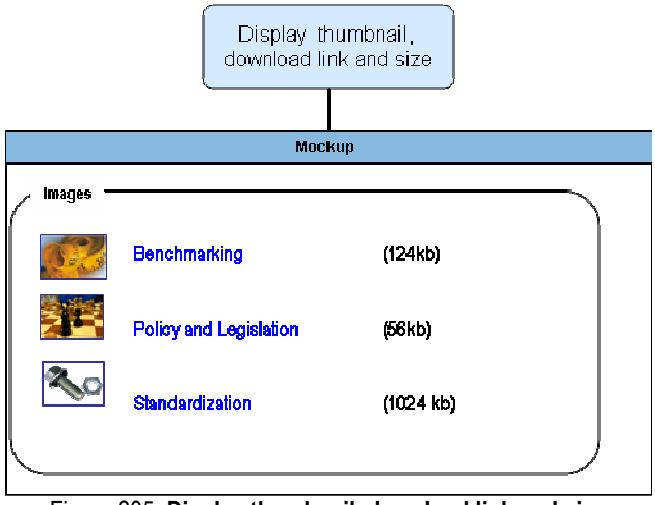


Figure 205: **Display thumbnail, download link and size**

In the three previous alternatives, the images are presented as thumbnails, along with information varying according to user web expertise. In the options of Figure 206 and Figure 207, the user may view images in a greater size and navigate among them by clicking on image buttons or on links respectively.

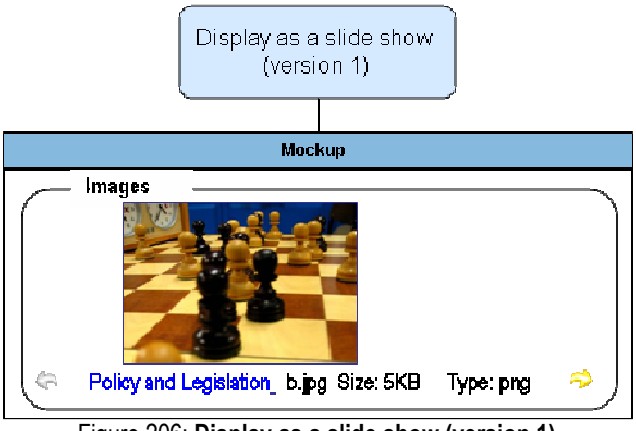


Figure 206: **Display as a slide show (version 1)**

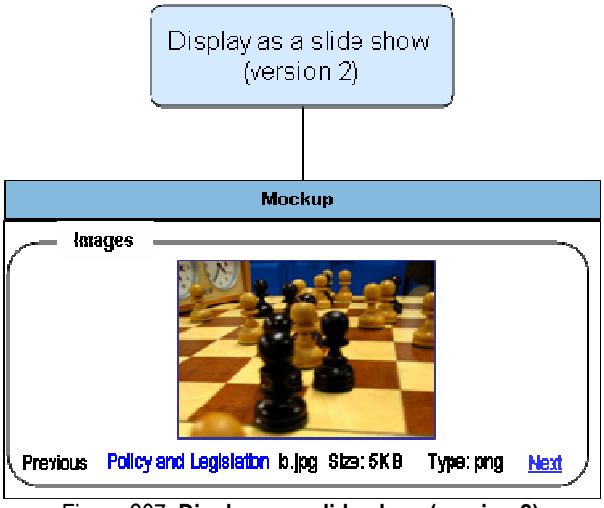


Figure 207: **Display as a slide show (version 2)**

Table 56: **Design rational of images representations**

Task: View images					
Style:	Display thumbnail, download link and estimated download time	Display full image info	Display thumbnail, download link and size	Display as a slide show (version 1)	Display as a slide show (version 2)
Targets:	Usability, flexibility	Usability, flexibility	Usability, flexibility	Usability, flexibility	Usability, flexibility
Parameters:	User (novice)	User (expert)	User (moderate)	User preference s	User preferences
Properties:	-	-	-	-	-
Relationships:	Exclusive	Exclusive	Exclusive	Exclusive	Exclusive

Search alternatives

Search functions provide a powerful tool for portal users helping them to localize information quickly, using some search keywords and parameters. The simple search design artefact includes limited capabilities for the novice user, in order to avoid confusion. As shown in Table 57, simple search includes only keywords typing, and the selection of the well-known ‘all words’ or ‘any words’ search option. The simple designed artefact with advanced switch was designed for moderate users, and includes in addition to the search keywords, optional search criteria. These criteria are used only when the user selects the link ‘more search criteria’, and also selects the appropriate dates to limit the search. The advanced search design includes by default the dates field, thus expert users have the possibility to select them quickly. Each of the three different design artefacts focuses on the individual user expertise, and offers more or less criteria options based on expertise.

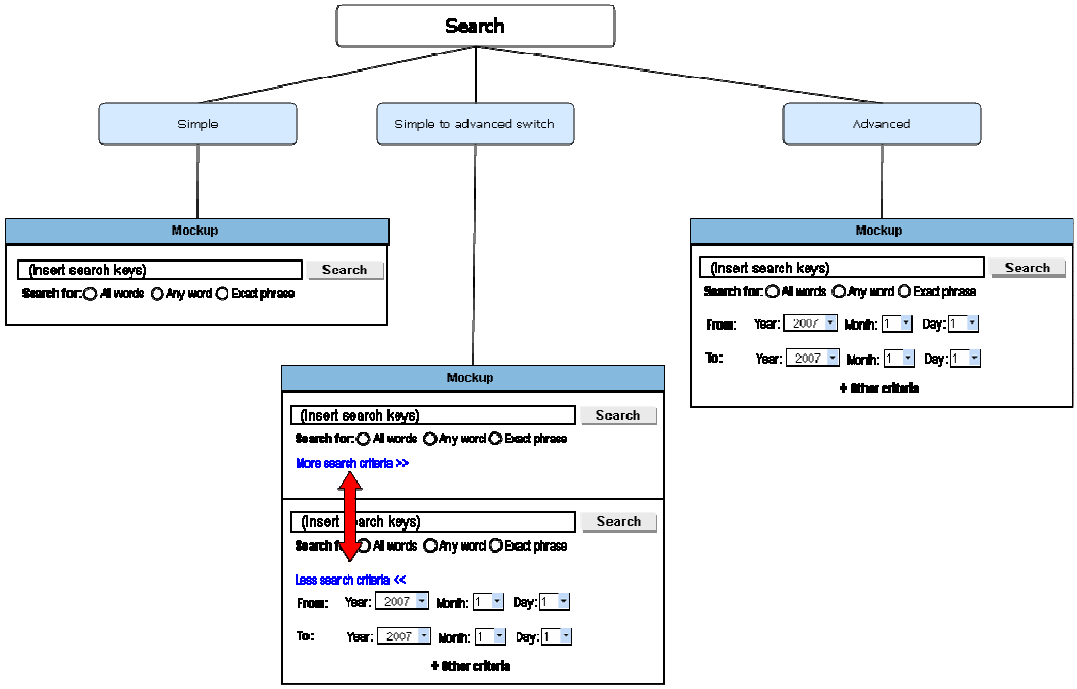


Figure 208: **Search function alternatives**

Table 57: Design **rational of the search functions alternatives**

Task: Search			
Style:	Simple	Simple to advanced switch	Advanced
Targets:	Simplicity, limited capabilities	Simplicity, with optional advanced criteria settings	Speed, effectiveness, default advanced criteria settings
Parameters:	User (novice)	User (moderate)	User (expert)
Properties:	Search term: Type search keywords, Select to search for all the words entered or for any word or for the exact phrase	Search term: Type search keywords, Select to search for all the words entered or for any word or for the exact phrase. Select link more search criteria and then select the date period of the search term	Search term: Type search keywords, Select to search for all the words entered or for any word or for the exact phrase. Select the date period of the search term
Relationships:	Exclusive	Exclusive	Exclusive

APPENDIX B: The EAGER toolkit (in detail)

EAGER builds on a number of primitive interaction elements to be subsequently used to form higher level interaction elements. The primitive interaction element presented in these sections are directly derived from their .Net framework counterparts, and enriched in order to support variations in style according to user and context specific parameters, and in output for rendering for example speech enabled output.

The following sections provide an overview of the framework developed in the context of this work for enabling the development of Unified Web Applications. Towards this direction several aspects of the framework are presented, focusing on class diagrams. The conventions employed for class diagrams description are presented in Table 58.

Table 58: **Conventions followed by the Microsoft Visual Studio 2005 class designer**

Class	Abstract class	Interface	Enumeration
			
Struct	**Delegate**	**Inheritance**	**Association**
			

In a modern web application, the ability to transform the presentation characteristics in term of UI positioning is of particular importance for providing alternative interaction schemes on request. In order for this to be generically applied through a web application, a specific UI layout and positioning mechanism was developed. The EAGER framework encapsulates the functionality for rendering a wide number of variant page templates. The development of the appropriate facilities was focused on providing alternatives for:

- Master pages: Control the overall positioning scheme to be applied on a page request
- Headers: Control the overall layout and positioning of header elements
- Footers: Control the overall layout and positioning of footer elements
- UI widgets: Control the presentation of content

With the help of the facilities presented above, the portal has the ability to render itself in a vast number of variations in terms of UI positioning and layout before the actual adaptation of functionality takes place. The following sections provide a more detailed insight of the specific implementation characteristics of each of the presented facilities, outlining their scope and function in the context of the EAGER.

Templates

Templates affect the basic characteristics of each page controlling the positioning to be applied on page content, the available containers of page content and the presentation characteristics used for rendering page content. To achieve this, a hierarchy of different template variations has been developed as presented in Figure 31.

Figure 209: **The hierarchy of the potential template variations**

The icsMasterPageBase acts as the base class for all different template variations available, exposing all the functionality to be overridden by its descendors for providing a specific implementation. More specifically, as presented in Table 14, icsMasterPageBase provides access to a number of placeholder for rendering each specific template contents and additional parameters for controlling template styles and functionality for rendering a template in variant ways (e.g., accessible presentation, design view presentation, etc.).

Header Templates

For providing variations of headers, a number of hierarchies were developed for controlling the positioning and layout of header contents. More specifically, the header containers hierarchy controls the higher level of positioning schemes providing the option of serial or encapsulated template – sub template layout. Each of these levels of abstraction is followed by hierarchies providing specific controls for alternative template or sub template schemes. Finally, the specific controls to be positioned in the containers exposed by header templates or the header sub templates are members of two other hierarchies. The following sections provide a step by step presentation of these hierarchies, followed by a description of the way all these are put together to form a specific header instantiation.

Header Container Templates

The hierarchy of header containers provides the first level of positioning scheme for each header. More specifically, the basic class of the hierarchy, namely icsHeaderContainerBase, provides the required properties for accessing styles and functions to be developed by it descendors for rendering according to the UI state (e.g., accessible, design, preview mode, etc.).

At runtime, each icsHeaderContainerTemplateBase instance is instantiated according to the user preferences as a specific member of the hierarchy, such as icsHeaderContainerTemplateEncaptulated or icsHeaderContainerTemplateSerial. A summative description of the control hierarchy is presented in Figure 210, which presents the class view of the hierarchy.

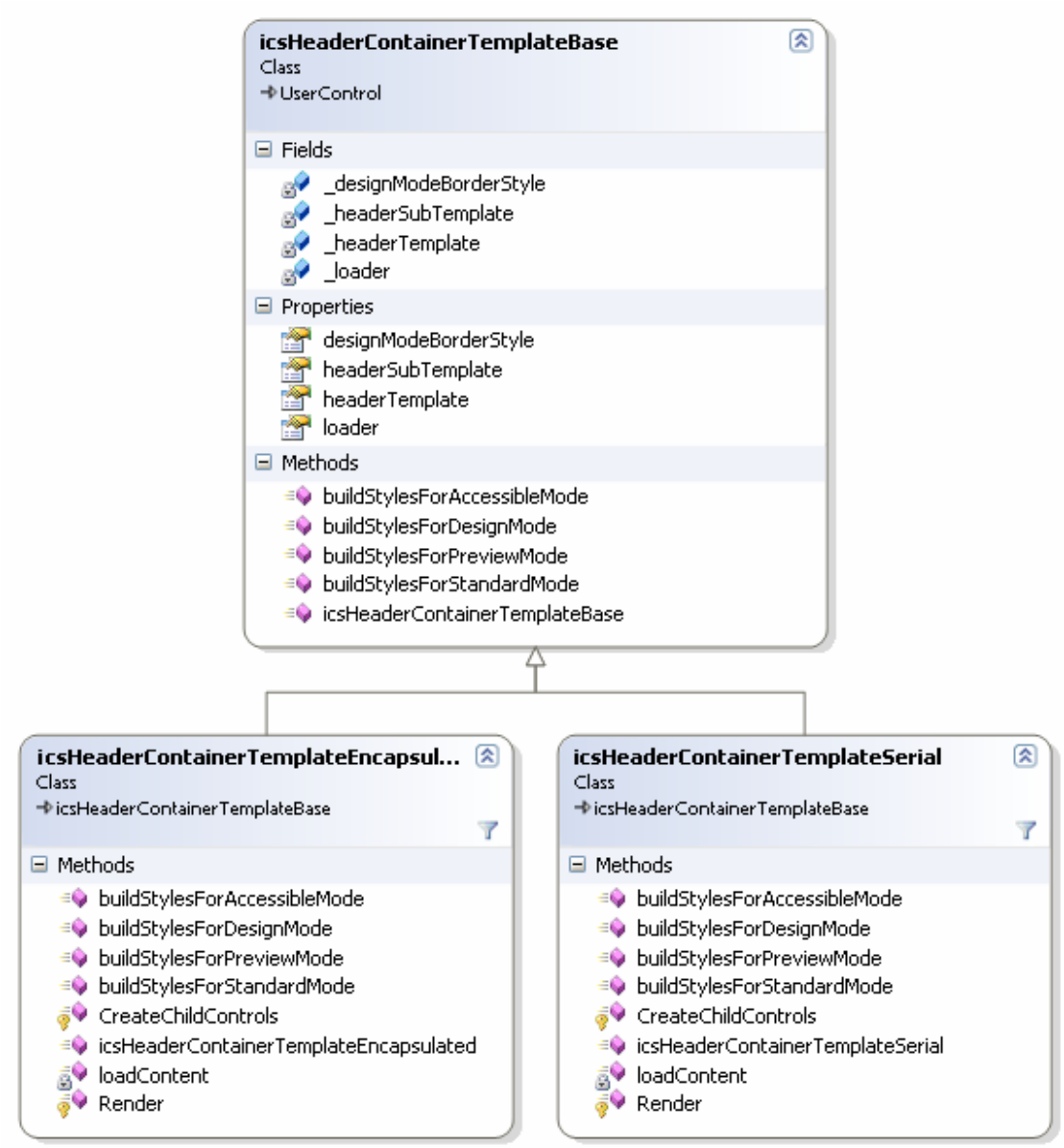


Figure 210: **Header container diagram**

Header Templates

As presented in the previous section, each header container template provides the abstract positioning scheme for two or more concrete objects: the header template, and the header sub template. These controls in turn encapsulate the specific header contents. The icsHeaderTemplateBase class, acting as a parent for all templates, exposes a number of functions for presenting header templates according to the UI state and additional options for controlling the presentation scheme to be used.

At runtime, each icsHeaderTemplateBase instance can be instantiated as any of the header template hierarchy members, each of which provides different positioning schemes. For example, the icsHeader3ColumsStatic template provides three content positioning placeholders of equal length. The available different instantiations are presented graphically in Figure 211.

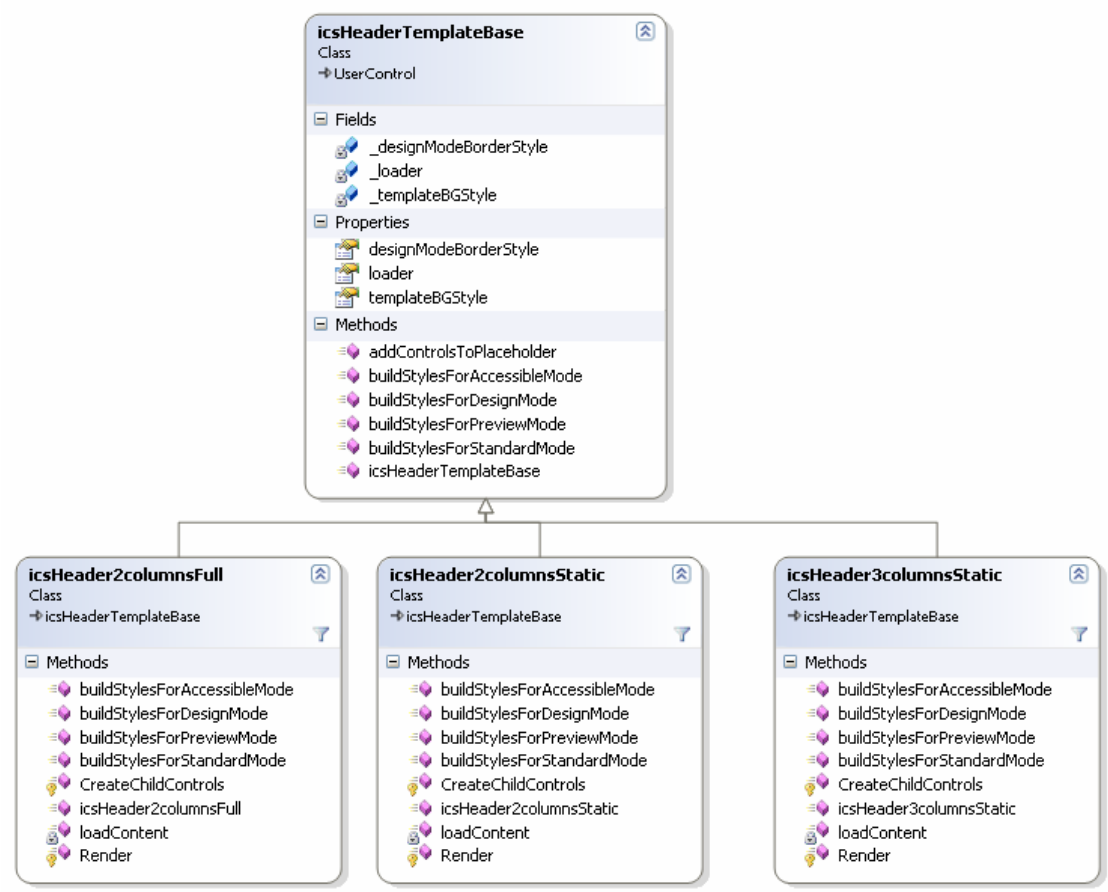


Figure 211: **Header template diagrams**

The specific controls to be placed on the exposed placeholders of each member of the aforementioned hierarchy are members of the template controls hierarchy presented in Figure 212. The icsHeaderTemplateControlBase is the parent of all control members. In that way, each header template handles a collection of template controls to be positioned at runtime in the templates exposed placeholders.

Figure 212: **Header template control diagram**

Header Sub - Templates

In the same manner as the header templates, the header sub templates hierarchy provides a number of alternative positioning schemes for the sub region of each template. The icsHeaderSubTemplateBase class, acting as a parent for all sub templates, exposes a number of functions for presenting header sub templates according to the UI state, and additional options for controlling the presentation scheme to be used.

The different runtime instantiations of each icsHeaderSubTemplateBase instance is presented graphically in Figure 213.

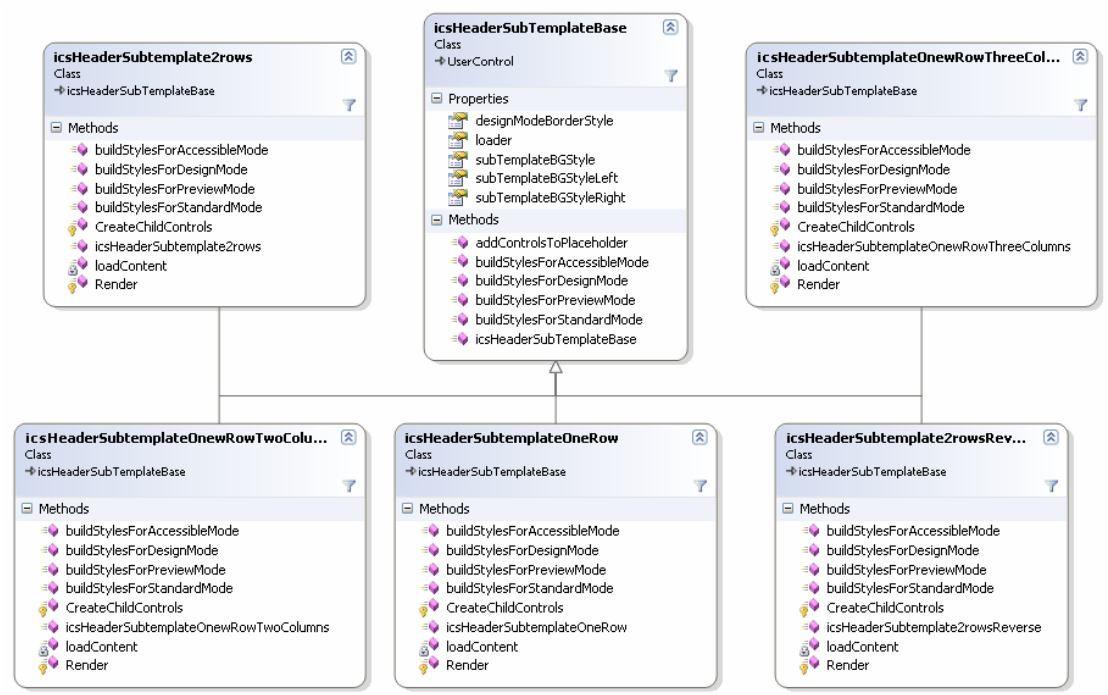


Figure 213: **Header sub templates diagram**

The specific controls to be placed in the exposed placeholders of each member of the aforementioned sub templates hierarchy are members of the sub template controls hierarchy presented in Figure 214. The icsHeaderSubTemplateControlBase is the parent of all control members. In this way, each header sub template handles a collection of sub template controls to be positioned at runtime in the sub templates exposed placeholders.

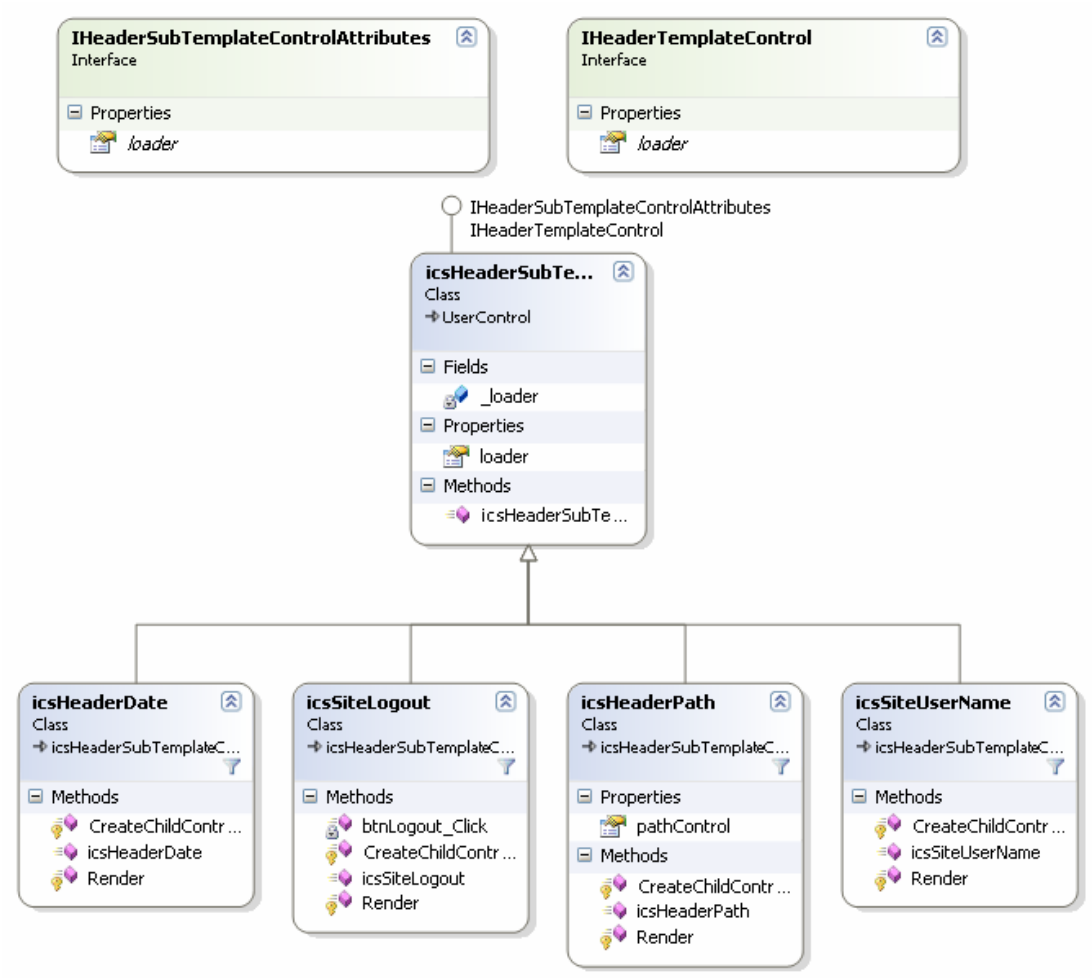


Figure 214: **Header sub templates controls**

Footer Templates

The footer templates hierarchy was developed for providing alternative footer instantiations in terms of presentation and positioning. The base class for each of the hierarchy members, called icsFooterTemplateBase exposes properties for controlling the footer layout and methods for rendering the footer according to the selected UI state (e.g., accessible, design state).

The different runtime instantiations of each icsFooterTemplateBase instance are presented graphically in Figure 215.

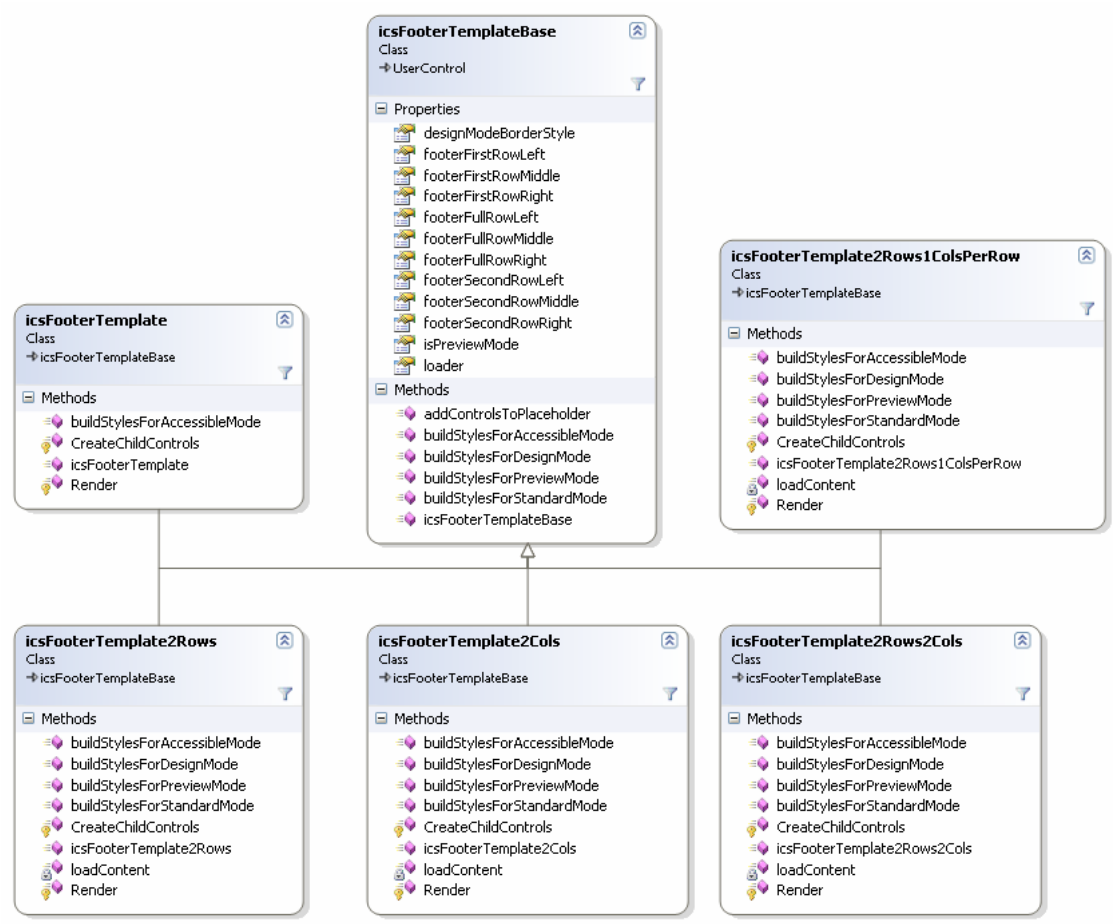


Figure 215: **Footer template diagram**

The specific controls to be placed on the exposed placeholders by each member of the aforementioned footer templates hierarchy are members of the footer controls hierarchy presented in Figure 216. The icsFooterControlBase is the parent of all footer controls. In this way, each footer template handles a collection of footer controls to be positioned at runtime in the exposed placeholders.

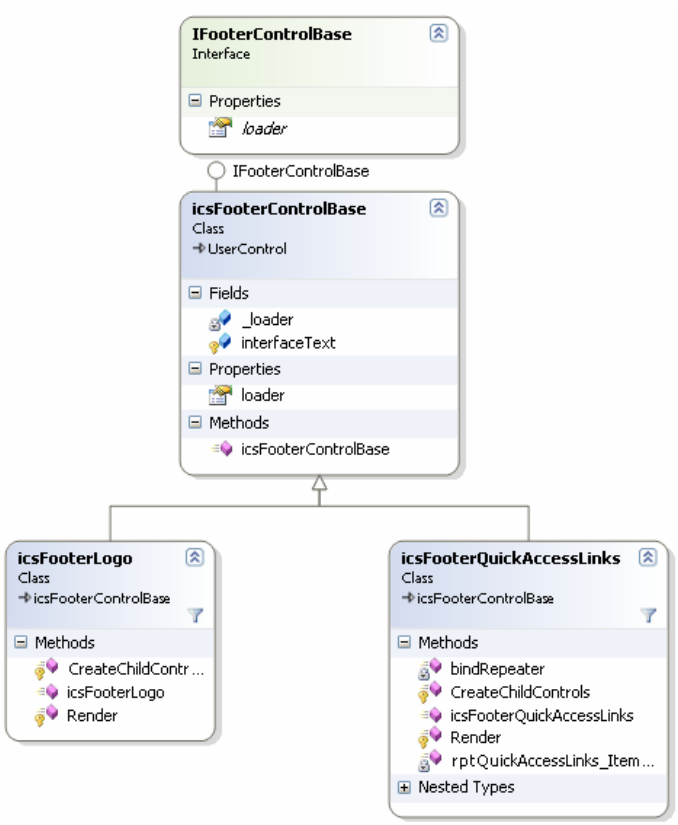


Figure 216: **Footer template control**

Windows

In the context of EAGER, windows act as containers of content providing aesthetic variations and a window-like look and feel. In order to support a wide number of different window styles, the widgets hierarchy was developed. The icsWidjectComponent acts as the base class of each widget, encapsulating all the required properties for rendering alternative representations, events for user actions and functions to be overridden by its descendors. The base class provides access to the required window data and the derived class is responsible for rendering the alternative representations.

The icsWidjectComponentRounded uses the data stemming from the icsWidjetComponent for appliying the selected skin to it self and for transmitting data to the loaded module. An overview of the widgets UI components hierarchy is presented in Figure 217.

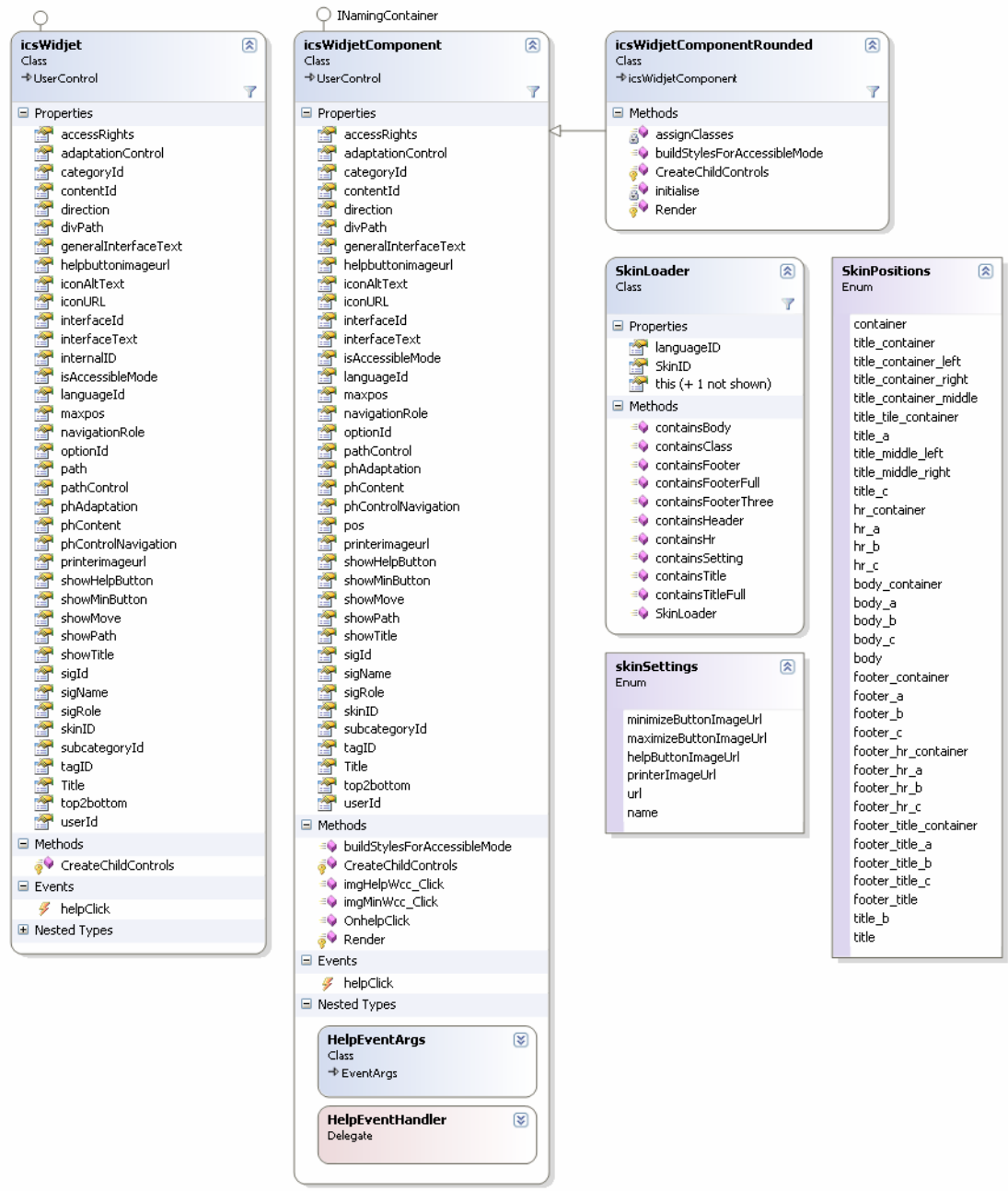


Figure 217: **Widget component diagram**

Navigation

EAGER supports various navigation schemes. The main navigation alternatives are those typically included in most web applications, namely the top, bottom and main navigation bars presented in the top, bottom and middle section of the application accordingly. Additionally, to address the need for unified navigation for different user characteristics, full navigation is presented incorporating all the aforementioned navigation elements in a navigation bar positioned at the top of the portal. Finally, the favourites' navigation facility is introduced for offering user favourite navigation options.

Top navigation

The top navigation hierarchy consists of the icsTopNavigationBase, which is the basic class for all alternative top navigation styles, and provides the related basic functionality. The complete top navigation hierarchy is presented in Figure 218.

Figure 218: **Top navigation diagram**

Bottom navigation

The bottom navigation hierarchy consists of the icsBottomNavigationBase, which is the basic class for all alternative bottom navigation styles, and provides the related basic functionality. At runtime, an instance of the alternative navigation styles is rendered based on the specific user characteristics. The complete bottom navigation hierarchy is presented in Figure 219.

Figure 219: **Bottom navigation diagram**

Main navigation

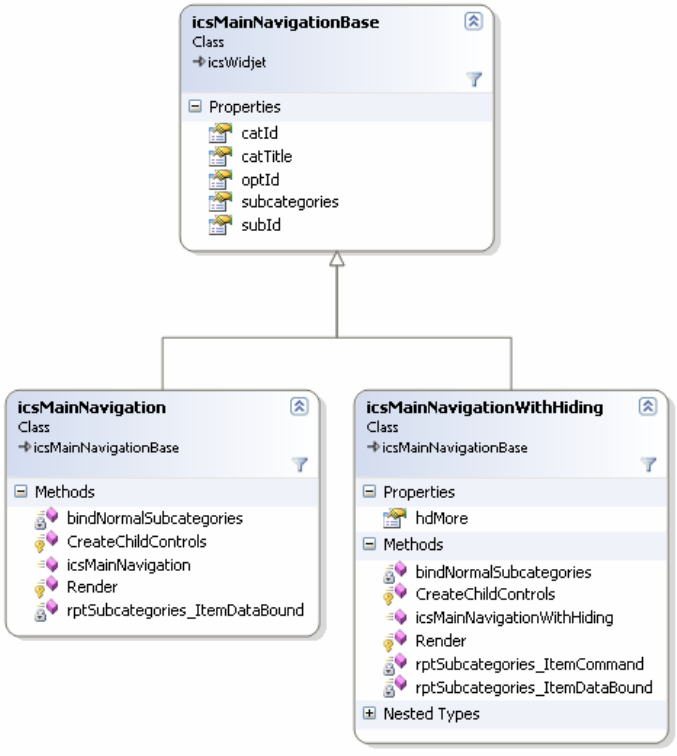


Figure 220: **Main navigation diagram**

Full navigation

In many occasions, the existence of several navigation schemes at different screen positions may influence the usability of a web application, especially when the scanning of a complete page for each selection is required. In this case, an alternative way to navigate through pages is required. The full navigation scheme was implemented to provide a navigation alternative that would be available at a specific screen location. The full navigation hierarchy is presented in Figure 221.

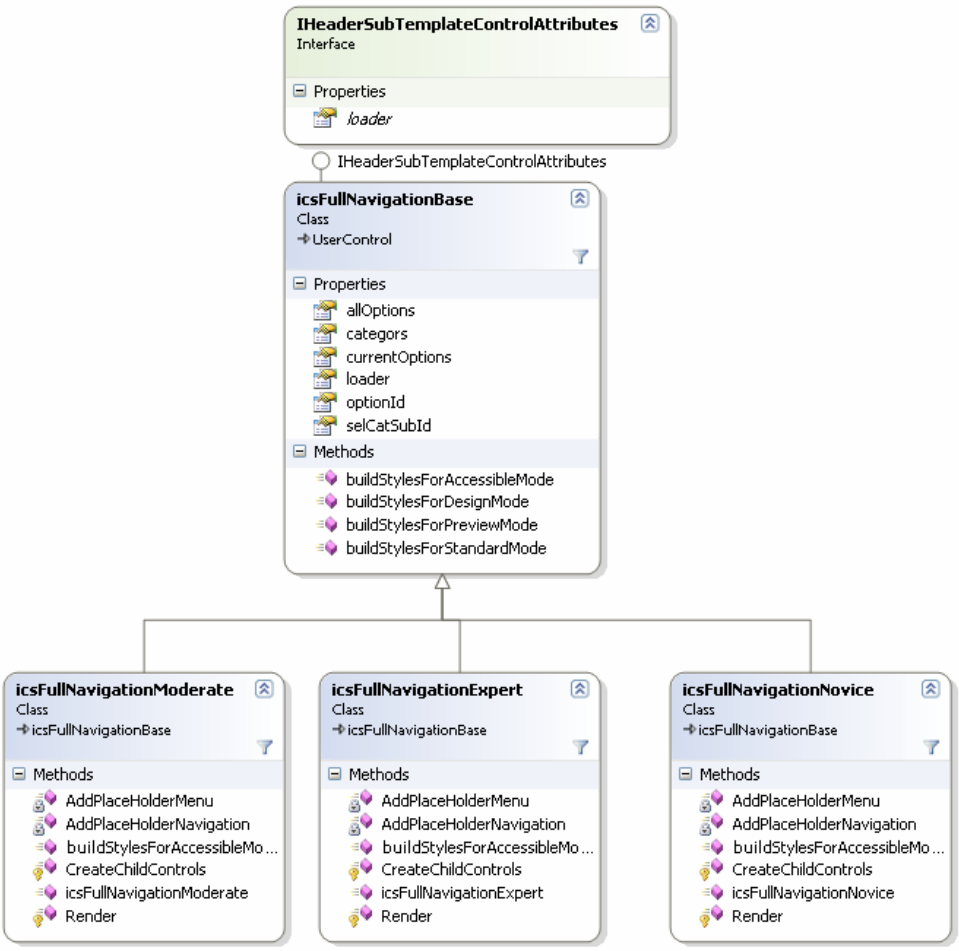


Figure 221: **Full navigation sub template control attributes**

The full navigation scheme is based on the composition of three alternative controls representing the navigation categories, subcategories and options. Each of the aforementioned controls has a unique UI components hierarchy:

- The icsCategories navigation hierarchy (see Figure 222): This hierarchy contains alternative categories representations according to the expertise of the user currently accessing an interface
- The icsSubCategories navigation hierarchy (see Figure 223): This hierarchy contains alternative sub - categories representations according to the expertise of the user currently accessing an interface
- The icsOption navigation hierarchy (see Figure 224 and Figure 225): This hierarchy contains alternative options representations according to the expertise of the user currently accessing an interface

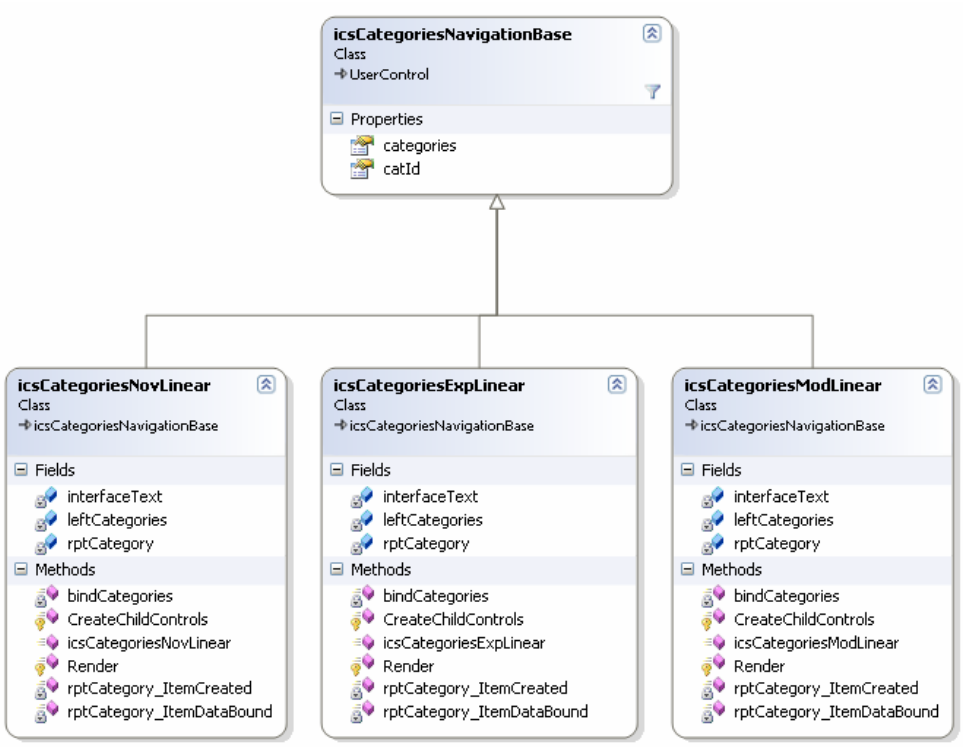


Figure 222: **Categories navigation diagram**

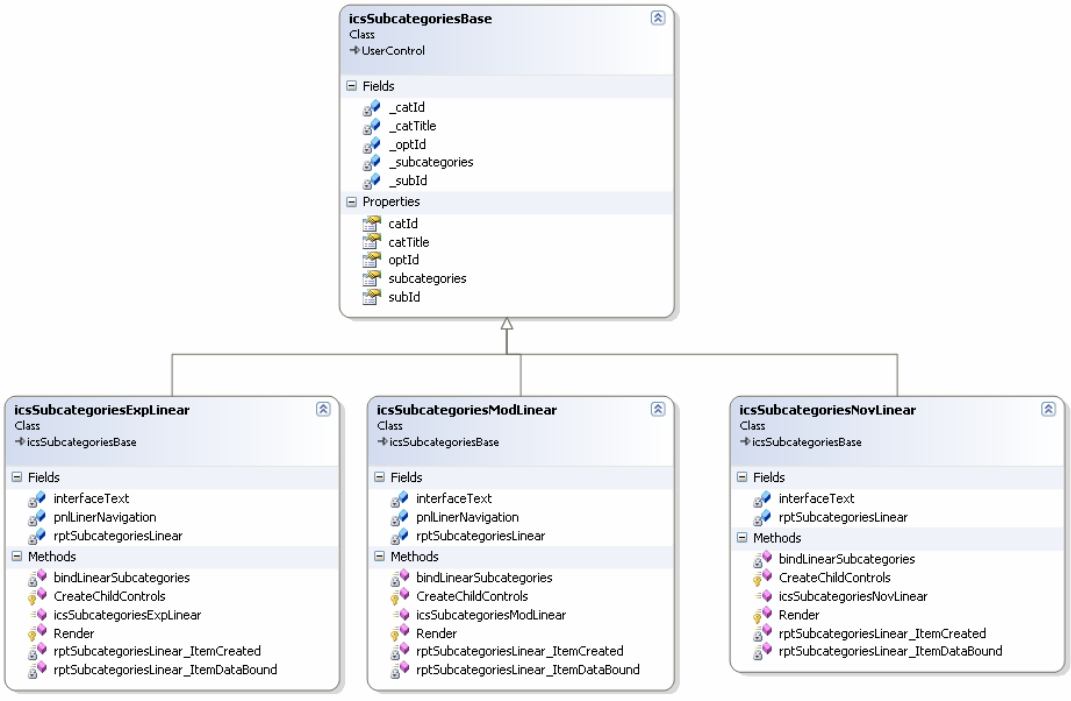


Figure 223: **Sub categories navigation diagram**

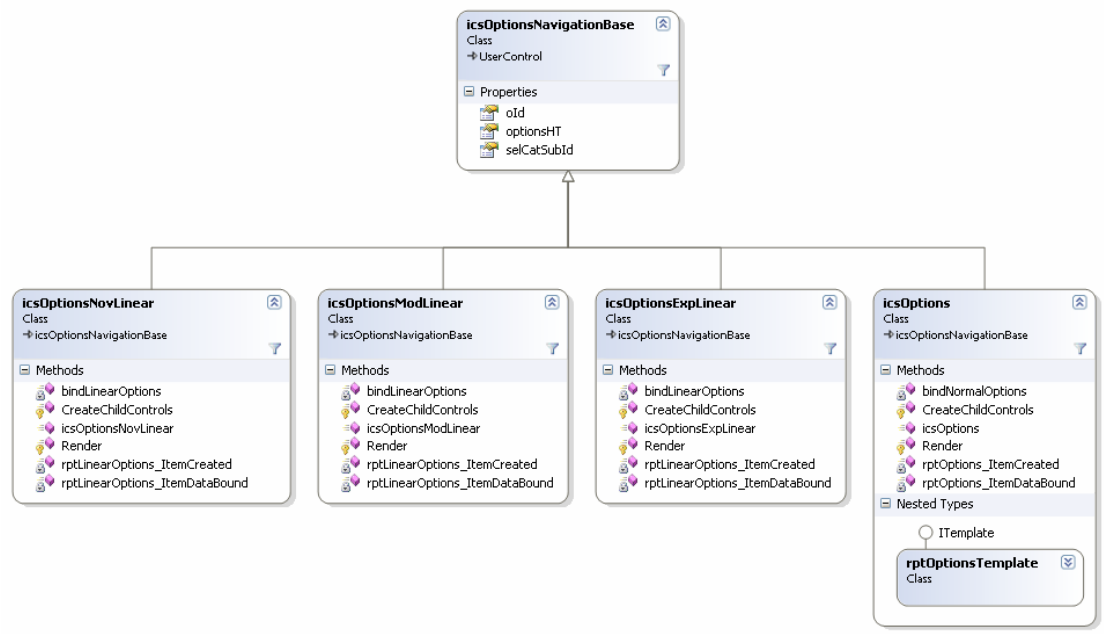


Figure 224: **Option navigation diagram**

Favourites navigation options

The favourite navigation options represent the most commonly used options from the navigation hierarchy. These options can be represented either as a separate navigation widget or be marked in place with a specific icon. Figure 225 presents the class for rendering the favourites widget.

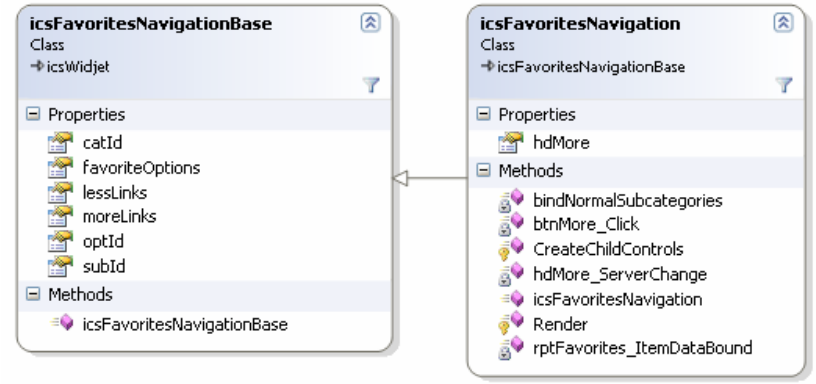


Figure 225: **Favourites navigation options**

Links

The links category contains all the framework members used for linking to other documents or used as function links. The members of this category are:

- icsLinkButton
- icsHyperlink
- icsAncor

The icsLinkButton control is extended from the basic asp:Button control, but its representation characteristics are inherited from the link control. The

icsLinkButton control can act as a standard button with the default post back behaviour, but is represented as a link. The icsHyperlink control is the basic object to be used when linking to another location on the Web. The control extends the basic asp:Hyperlink control providing additional functionality from the server side. The icsAncor control is yet another link representation inherited from the standard htmlAncor control. This control is to be used when the standard html functionality is required, but with the extended presentation behaviour provided by EAGER.

The aforementioned controls can be rendered through various representations according to specific user parameters, such as:

- Links with variant foreground-background styles.
- Buttons with variant foreground-background styles.
- Speech enabled links or buttons

The class representation of these link type controls is presented in Figure 226.

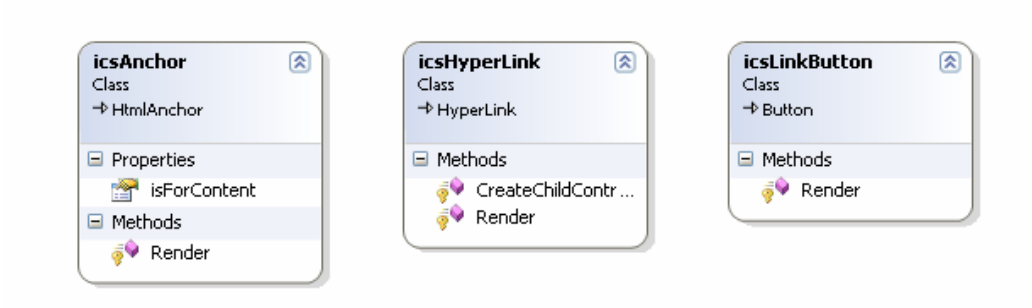


Figure 226: **The alternative link controls**

Buttons

The icsButton control extends the basic asp:Button control in order to provide the additional presentation characteristics of the framework. The icsButton control can be rendered through a number of variations:

- Link with variant foreground-background styles.
- Button with variant foreground-background styles.
- Speech enabled

The class representation of the icsButton control is presented in Figure 227.

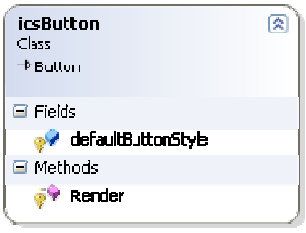


Figure 227: **The icsButton controls**

Single selection controls

The single selection controls are used for selection among Boolean values. The overridden framework controls that support these facilities are:

- icsRadioButton
- icsCheckBox

The icsRadioButton and the icsCheckBox controls were extended to provide additional functionality when rendered inside tables, and to support speech output. The class representation of these controls is presented in Figure 228.

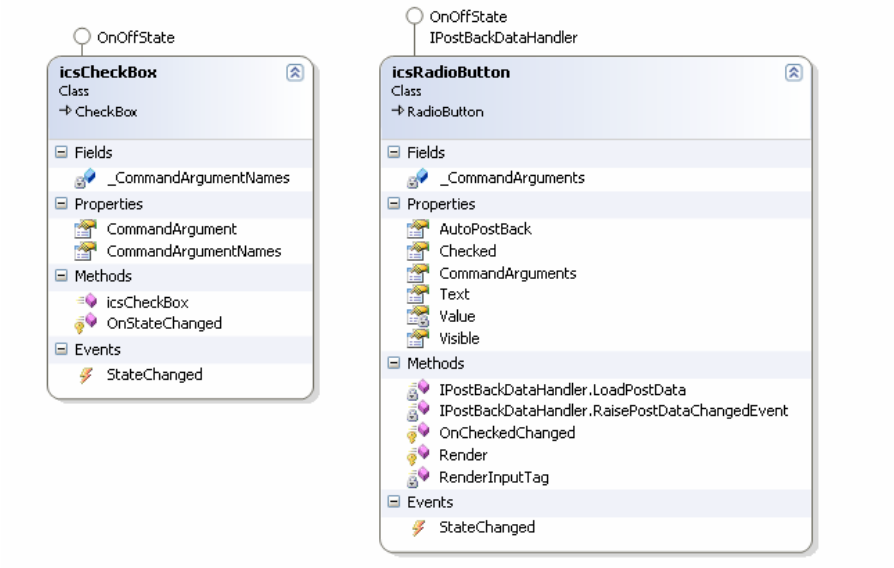


Figure 228: **The class diagram representing the alternative selection controls**

Labels

The labels controls hierarchy contains all the controls that represent labels. This category contains all the alternative variations according to the desired functionality attributes or presentation characteristics. Mores specifically, the following label controls are available:

- icsLabel
- icsHTMLLabel
- icsErrorLabel

The icsLabel control is directly extended from the standard asp:Label control in order to provide the additional functionality required by EAGER. The icsHtmlLabel control is derived from the html label control and is used to combine standard html label functionality with the inherited framework characteristics. The icsErrorLabel control was created to support different presentation schemes for representing error messages according to the user characteristics, such as, for example, colour blindness.

The aforementioned controls can alter their presentation scheme according to specific user characteristics to support variations in the background – foreground

colour, or alter completely their presentation to support Text to Speech output. The class representations of these controls are presented in Figure 229.

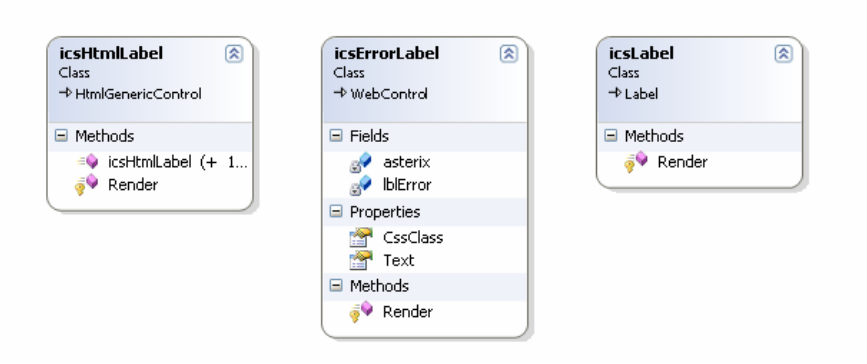


Figure 229: **The alternative label controls**

Multiple selection elements

The icsDropDown control was directly extended by the standard asp:DropDown control to provide the different presentation schemes to be supported by EAGER. The added functionality includes the altering of font style and size and the rendering of different foreground-background variations. The class diagram representation of the icsDropDown control is shown in Figure 230.

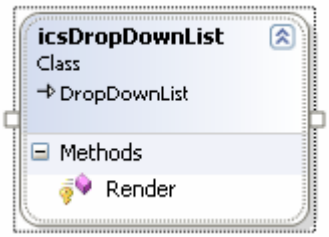


Figure 230: **The icsDropDown control**

Images

Image presentation in a web application can be altered according to the specific user characteristics and presentation requirements. The different image displaying schemes presented in Appendix A are facilitated by EAGER through the image UI components hierarchy presented in Figure 29. The hierarchy is constituted by the icsImageControl base, which acts as a common parent for all alternative image styles. This base class includes all the appropriate fields and methods in order to be extended and used by the child classes to achieve specialized configuration of an image control.

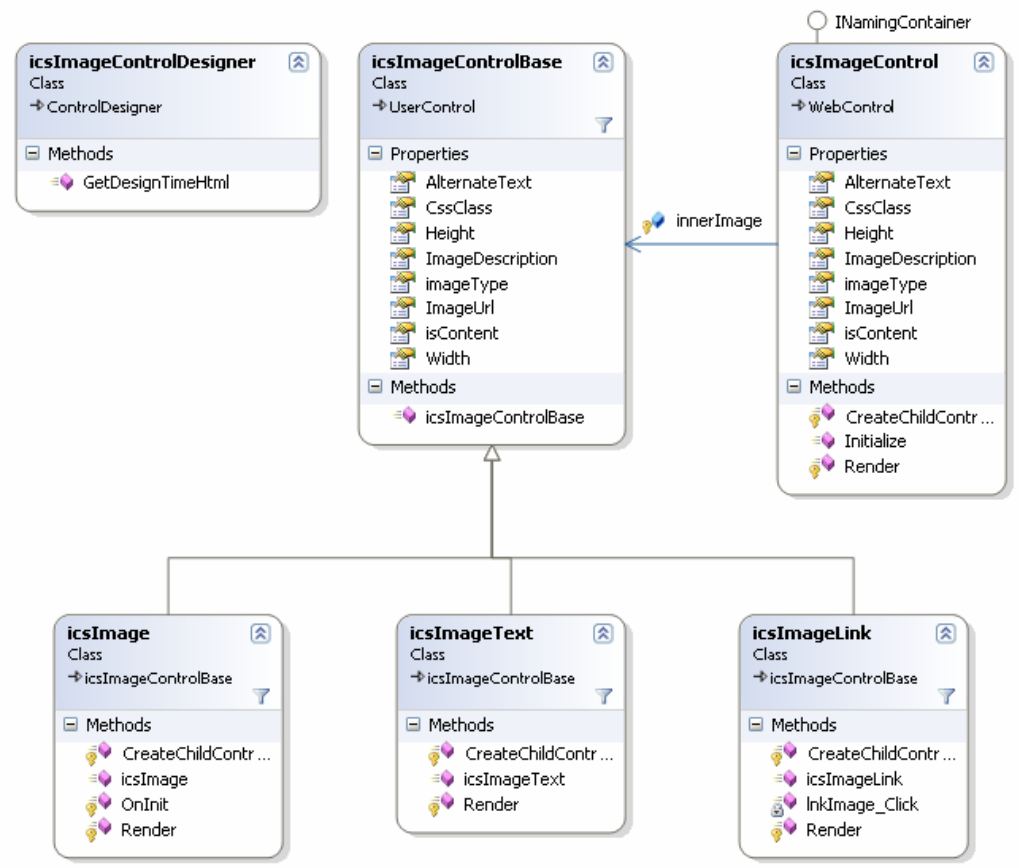


Figure 231: **Images hierarchy**

The 'icsImageControlBase' child classes are presented in Table 11 along with a summative description. 'icsImageControl' (Figure 29) encapsulates an 'icsImageControlBase' object that at runtime is instantiated in an object of the child classes providing end user with a configurable image control, depending on the decision making component commands. 'icsImageControl' is placed on the Visual Studio toolbox from where the developer drags and drops it in the Visual Studio designer. In order for this control to be represented, the 'icsImageControlDesigner' class is used, providing html code for rendering graphically the control in the designer.

Tabs

Web portals include information that is divided in several categories and usually presented in separate tabs. For supporting the different tab styles identified in Appendix A, the tab styles control hierarchy was developed as shown in Figure 232.

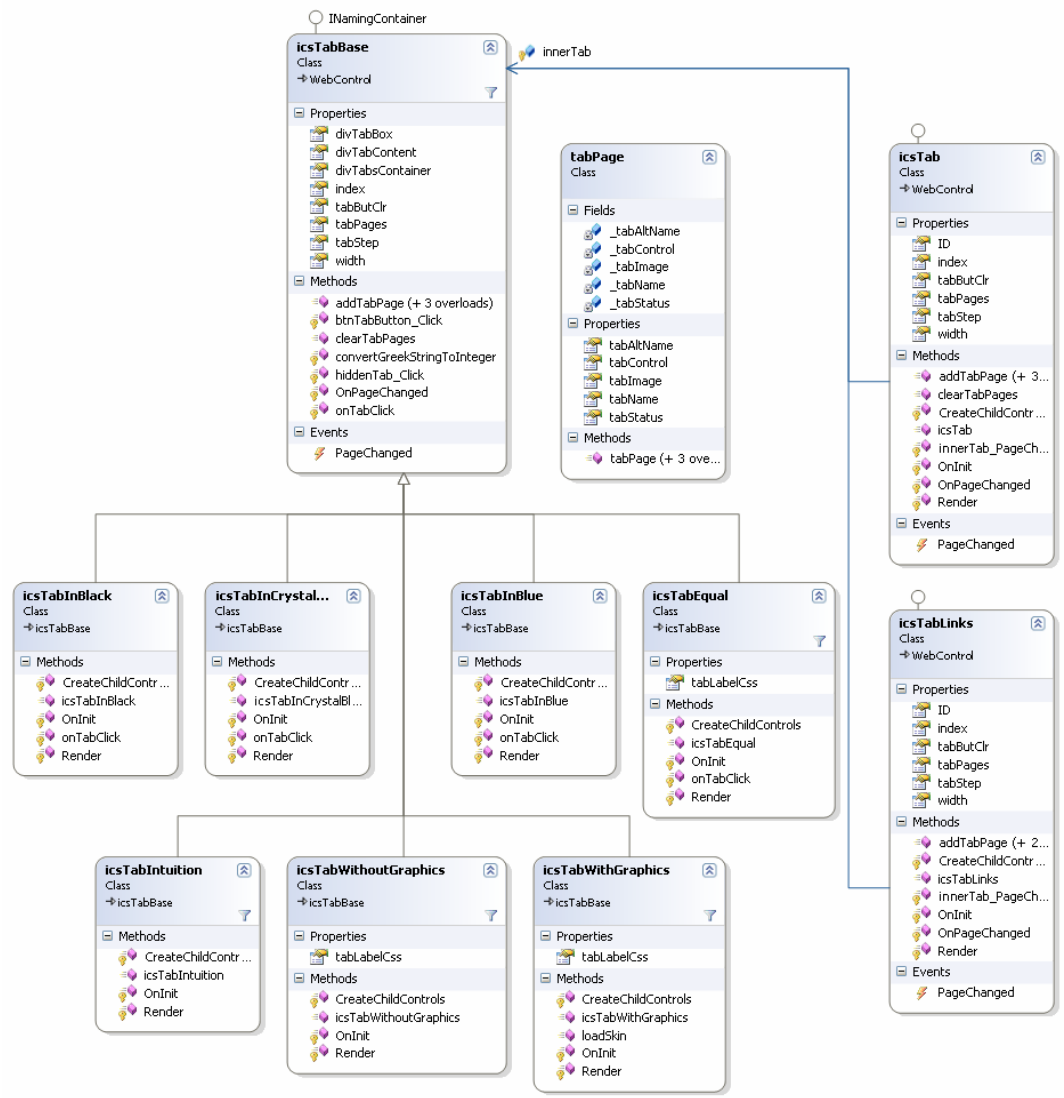


Figure 232: **Tabs diagram**

In this hierarchy, the'icsTabBase' class acts as a common parent for all tabs, exposing the basic functionality for all the tab UI elements.

As shown in Figure 232, 'icsTabBase' has a number of child classes (icsTabEqual, icsTabIntuition, icsTabWithGraphics, icsTabWithoutGraphics), each of which represents a different tab style. The 'icsTabDesigner' contained in the tab control library is intended to be automatically used by Microsoft Visual Studio for providing design time support. End users are provided with a generic 'icsTabControl' that encapsulates a base instance of the tab hierarchy ('innerTab'). This internal instance is instantiated at runtime in any of the control hierarchy members.

Field Sets

In the context of web applications, field sets are used for grouping and categorizing information. As identified in Appendix A, field sets can vary in style for supporting light-weight interaction together with screen readers and more rich graphical representation for all other occasions. This is facilitated by EAGER through the field sets UI hierarchy, presented in Figure 233.

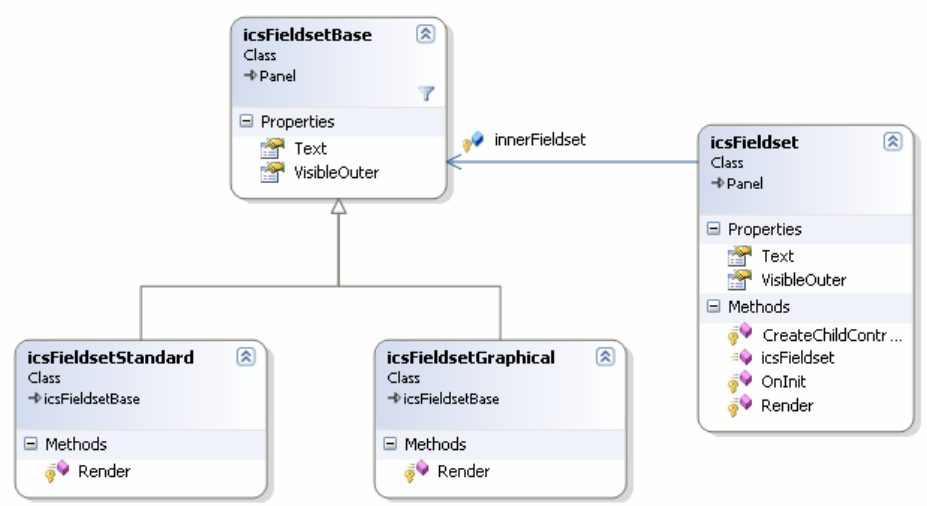


Figure 233: **Field set diagram**

The UI Hierarchy consists of the 'icsFieldsetBase' that provides the basic properties and functionality for the fieldset UI element and two derived classes ('icsFieldsetStandard', and 'icsFieldsetGraphical') for the alternative UI representations.

The 'icsFieldsetDesigner' class contained in the field sets namespace incorporastes the appropriate functionality for representing the icsfieldset control in the Visual Studio designer. Developers are provided with an 'icsFieldset' control. This control encapsulates an instance of the 'icsFieldsetBase' class which is instantiated at runtime as a concrete derived field set control.

Tables

Tables are used frequently by web applications, but some of them cause problems for screen reading software, since screen readers tend to read across the screen in a way that runs all of the text on a line together. Toward addressing this issue specific linearization schemes were proposed in Appendix A. These schemes are facilitated by EAGER through the creation of a generic table control. This control uses specific header and item templates to determine its contents and the linearization scheme to be applied. End users of the framework can create tables without the need to write specific code in order for the alternative linearization schemes to be displayed. The UI controls hierarchy for representing the alternative templates for headers – items is presented in Figure 234.

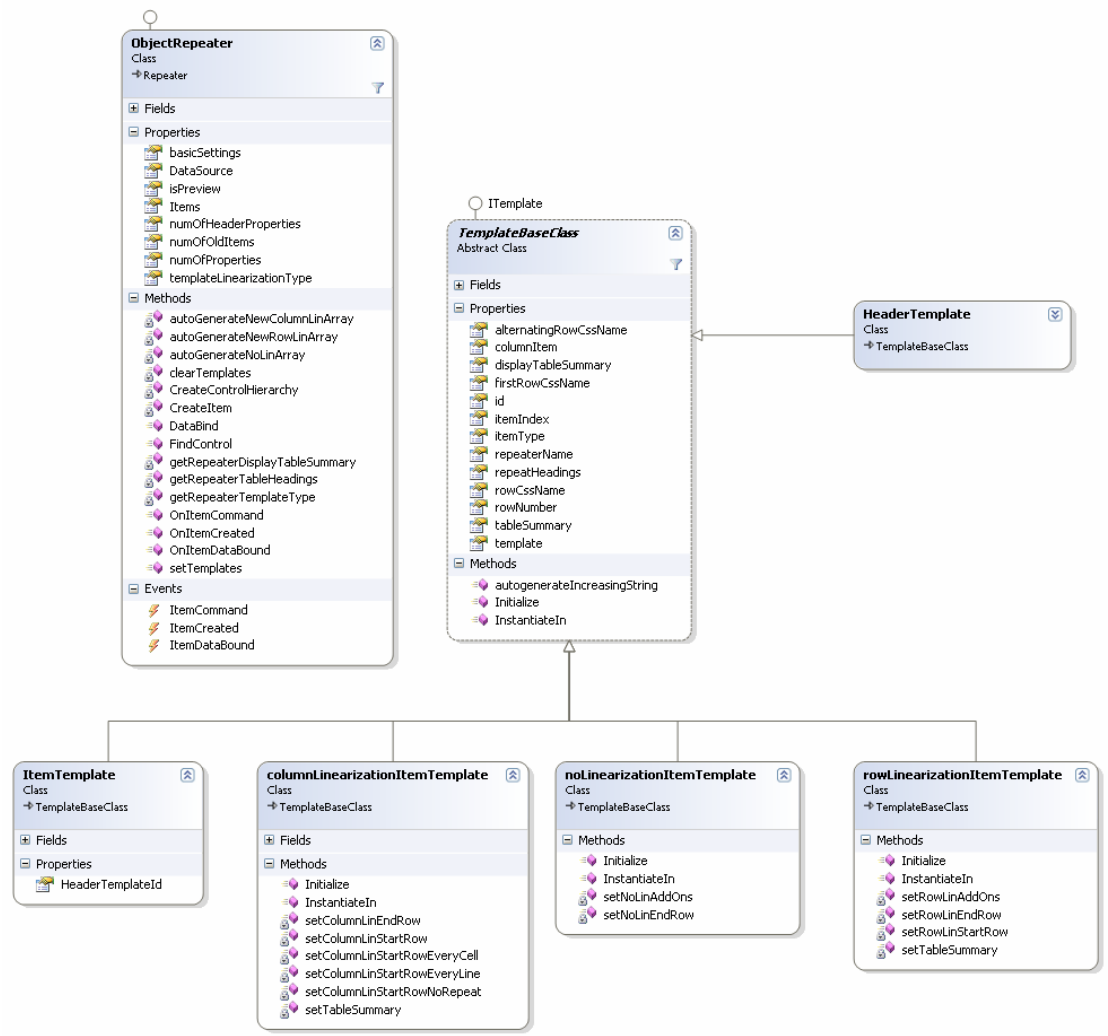


Figure 234: **Tables diagram**

Charts

Presenting charts in the context of a universally accessible web portal is not a trivial task. Charts may vary in the way that information is presented, but also in the colour scheme selected for displaying chart graphics. EAGER incorporates different hierarchies for presenting different kind of charts, and a help hierarchy for providing the colour scheme to be used for rendering charts.

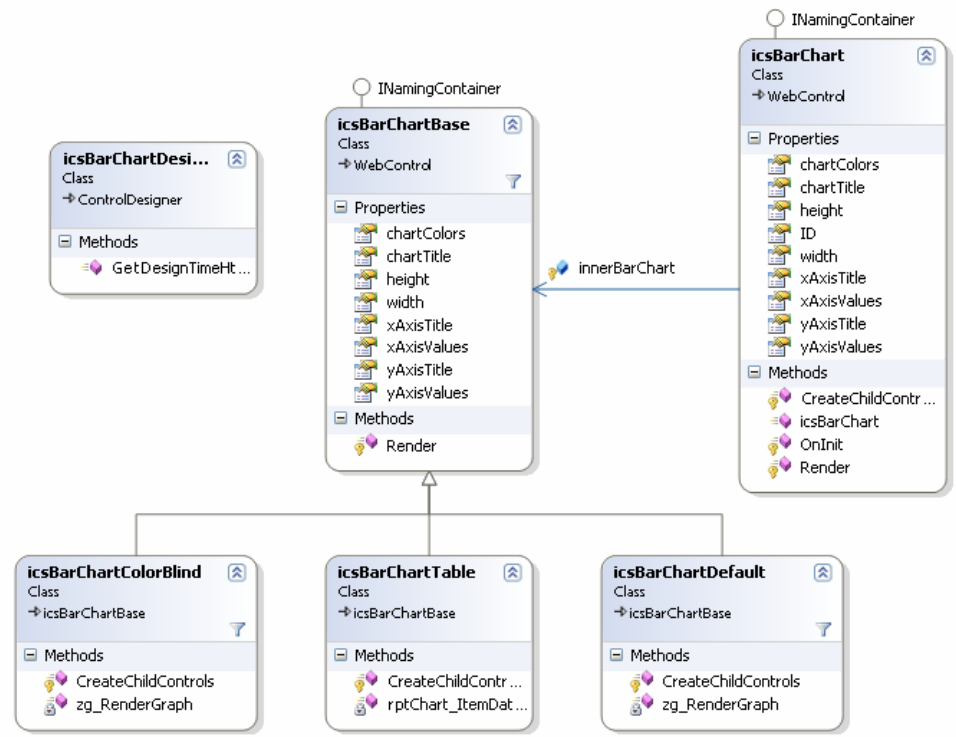


Figure 235: **BarChart diagram**

The bar chart hierarchy is shown as part of the Figure 235, presenting the base class of the hierarchy 'icsBarChartBase' along with the classes deriving from it. The 'icsBarChartBase' class includes properties and methods in order to be extended by the derived classes.

The 'icsBarChartDesigner' class is used by the Visual Studio designer in order to represent a bar chart object at design time. The 'icsBarChart' is a web control that includes a 'barChartBase' object named 'innerBarchart', exposing all the appropriate functionality of a barChart, and is instantiated in any of the derived class objects 'icsBarChartColourBlind', 'icsBarChartDefault', or 'icsBarChartTable', depending on user profile options and the context of use.

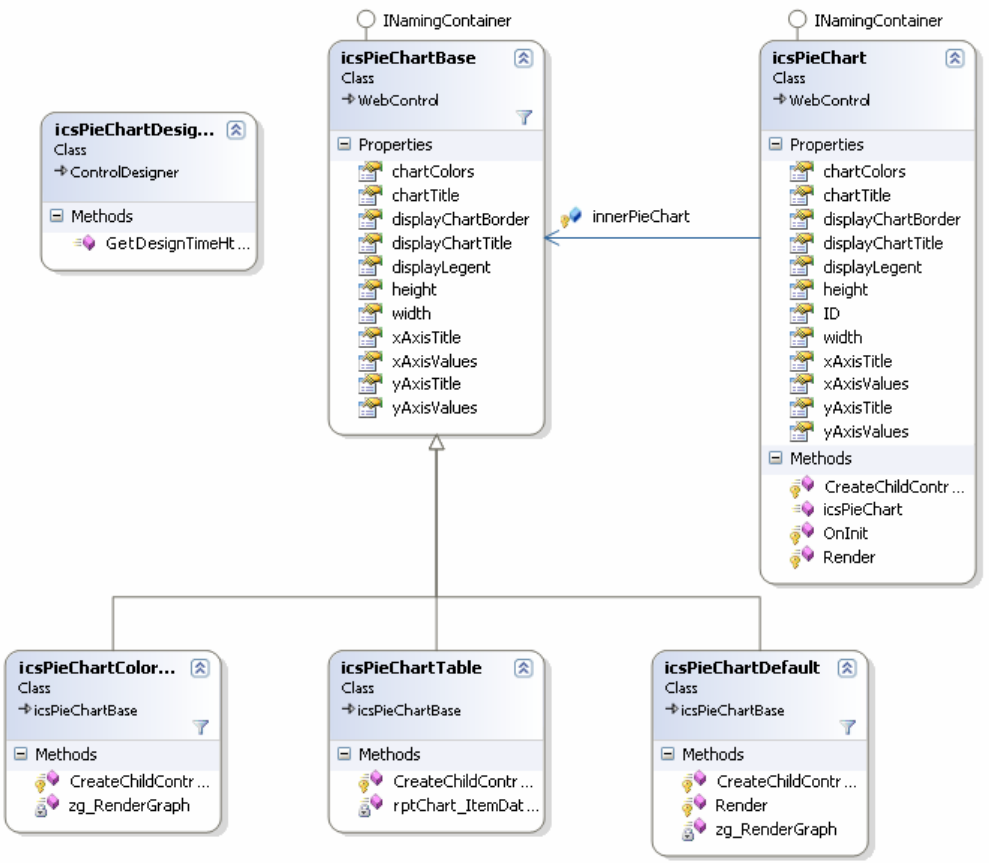


Figure 236: **Pie chart diagram**

Figure 236 represents the class hierarchy for the pie chart category. The pie charts hierarchy consists of the base class 'icsPieChartBase' class, including all the appropriate functionality to be exposed by the derived classes 'icsPieChartColourBlind', 'icsPieChartDefault', and 'icsPieChartTable'.
'icsPieChart' is web control that includes an 'icsPieChartBase' object named 'innerPieChart', exposing the functionality of the pie chart and transforming itself to the appropriate derived class object. This control is offered as a Visual Studio toolbox object that at design time can be dragged and dropped by the developer in the Visual Studio designer in order to use a pie chart in a web page. The 'icsPieChartDesigner' class defines the way that the pie chart will be rendered in the Visual Studio designer.

To address colour blindness, the necessity of developing a mechanism to produce applicable colour collections for charts arises. The 'icsBarChartBase' and 'icsPieChartBase' classes include an object named 'chartColours'. This object defines a collection of known colours that will be used in order to render different values on charts. The 'chartColour' object is of type 'icsChartColourBase' and is instantiated at runtime in any of its child classes: 'icsChartColoursDefault', 'icsChartColoursProtanope', 'icsChartColoursDeuteranope', and 'icsChartColoursTritanope', depending on the colour vision impairment (none, protanope, deuteranope and trinanope correspondingly, see Figure 237). Each one of these classes includes a method 'getColours' derived from the base class that produces the appropriate colour combinations for each impairment.

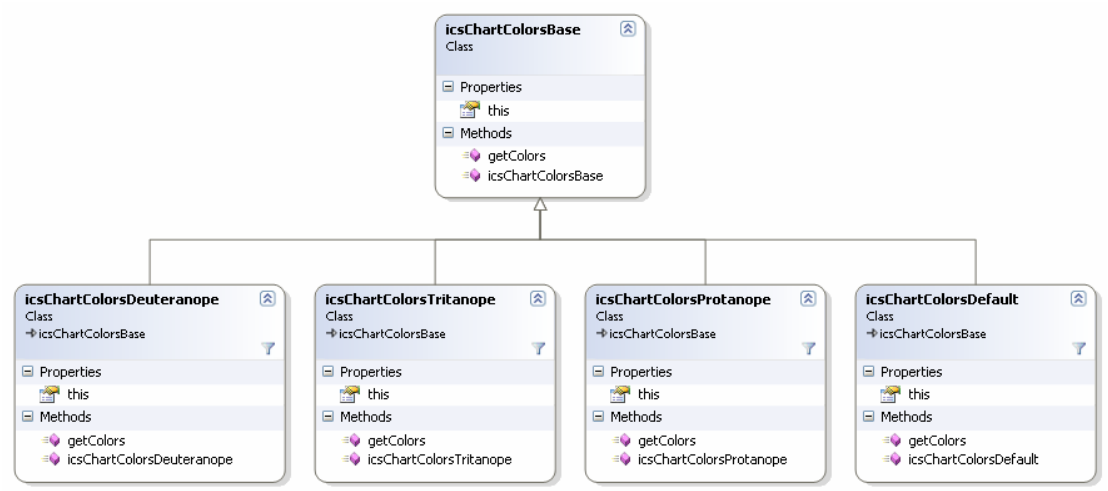


Figure 237: **Colour chart diagram**

Module options

Module options encompass the need for navigation support in the context of a module for accessing the available navigation options. In order for EAGER to support variations in the presentation of module options, the module options hierarchy was developed as presented in Figure 238.

Figure 238: **Module options diagram**

According to the specific user settings, each 'icsModuleOptionsBase' object is instantiated at runtime as a concrete 'icsModuleOptionsTabStyle', 'icsModuleOptionsWithoutGraphics' or icsModuleOptionsTopRightWindow' instance.

The end user of the control library is provided with an icsModuleOption web control which can be used through the designer. A concrete designer class is provided, 'icsModuleOptionsDesigner', responsible for providing design time support for the control (rendering the html representation of the control to the control designer). The 'icsModuleOptions' control adapts to the module option hierarchy containing a generic base class instance and exposes the provided functionality. This internal instance can be polymorphosed at runtime to meet individual user and context attributes.

Functions

Functions are usually incorporated by web portal in order to submit data on a server or perform some meaningful operations. As discussed in Appendix A, the way that these functions are presented can enhance the visibility and understandability of the available operations. The module functions class hierarchy was developed for supporting alternative function styles as presented in Figure 239. The

'icsFunctionsBase' class is a base class including methods and properties, and providing the appropriate functionality to be inherited by the derived classes. The derived classes include the desired functionality in order to build an alternative functions' style.

Figure 239: **Functions diagram**

The 'icsFunctions' constitutes a web control that includes an 'icsFunctionsBase' object named 'innerFunctions' that is altered according to portal settings in an object of the derived classes, and encloses all the appropriate functionality of the 'portal functions'. The 'icsFunctions' control is placed in a Microsoft Visual studio toolbox, from where the developer drags and drops it on Microsoft Visual Studio designer; at runtime this control is instantiated in an object of the 'icsFunctionsBase' class children presented in Figure 239.

Paging- Single items

In the context of web portals, paging mechanisms are applied on results stemming from a search query or a browsing operation in order for the users to have sequential access to them. The way that paging facilities are presented to users and the

functionality included may vary according to user characteristics, as identified during in Appendix A. The class hierarchy containing the identified alternative paging styles is presented in Figure 240. The base class of all paging styles named 'icsPagingBase' class incorporates methods and properties to be inherited and extended by the derived classes.

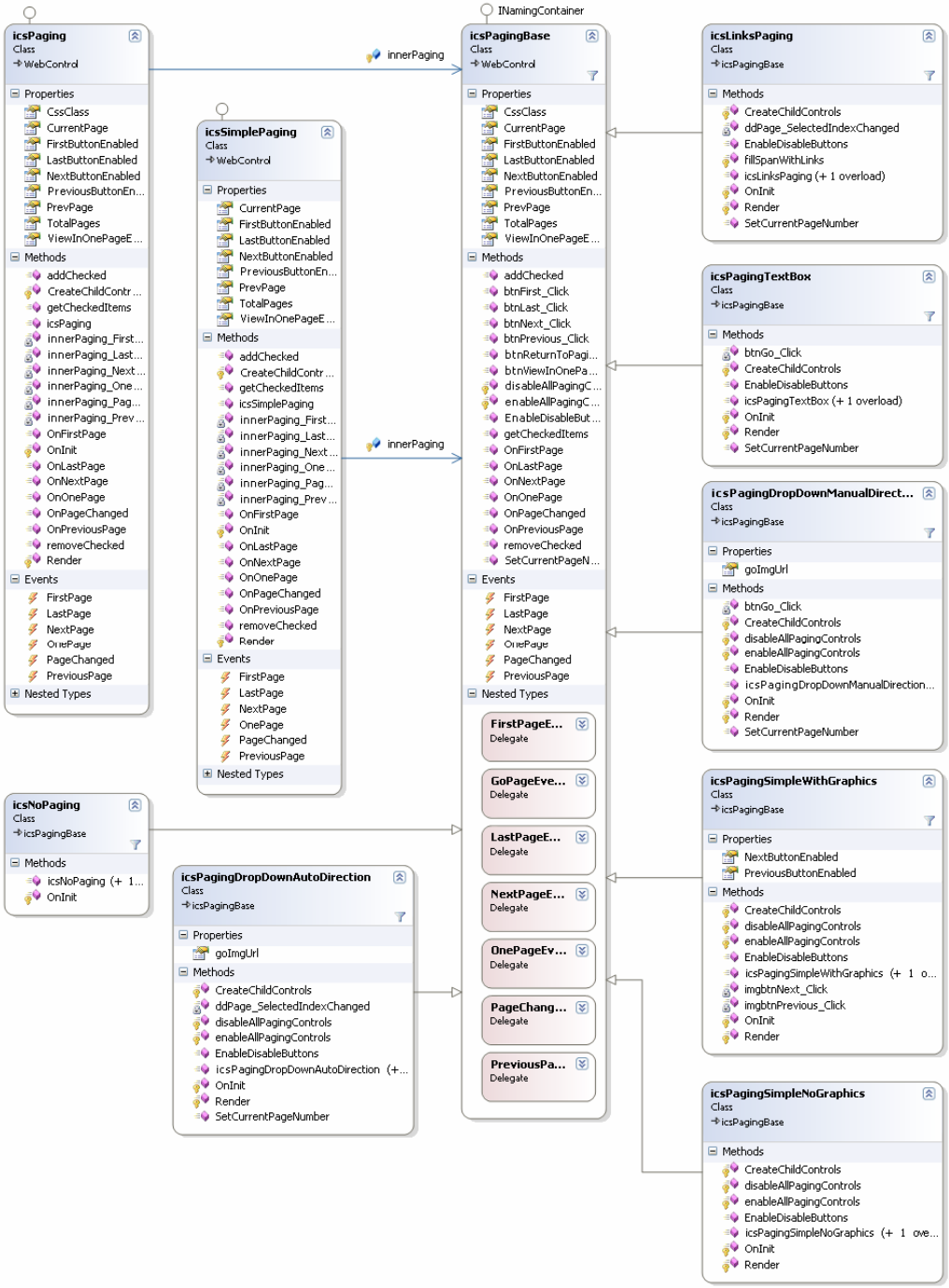


Figure 240: **Paging diagrams**

In order to construct the polymorphic hierarchy described above, a web control 'icsPaging' was developed that encapsulates an 'icsPagingBase' control instantiated at runtime in an object of the derived classes, depending on user and context attributes. The class 'icsPagingDesigner' includes the appropriate

functionality in order for the 'icsPaging' control to be rendered in the visual studio designer at design time.

Text entry

Text entry in a Web portal typically includes the use of a text area and a keyboard. In Appendix A, alternative scenarios for typing text using a virtual keyboard are identified, motivated by scanning techniques for users with limited motor functionality of upper limbs and contexts of use in which the keyboard is absent (e.g., tablet pc). These alternative text entry methods are facilitated by the text input UI hierarchy presented in Figure 241. The base class of the hierarchy, namely 'icsInputBase', includes basic properties and is derived from 'icsSimpleTextBox' and 'icsTextBoxWithVK'.

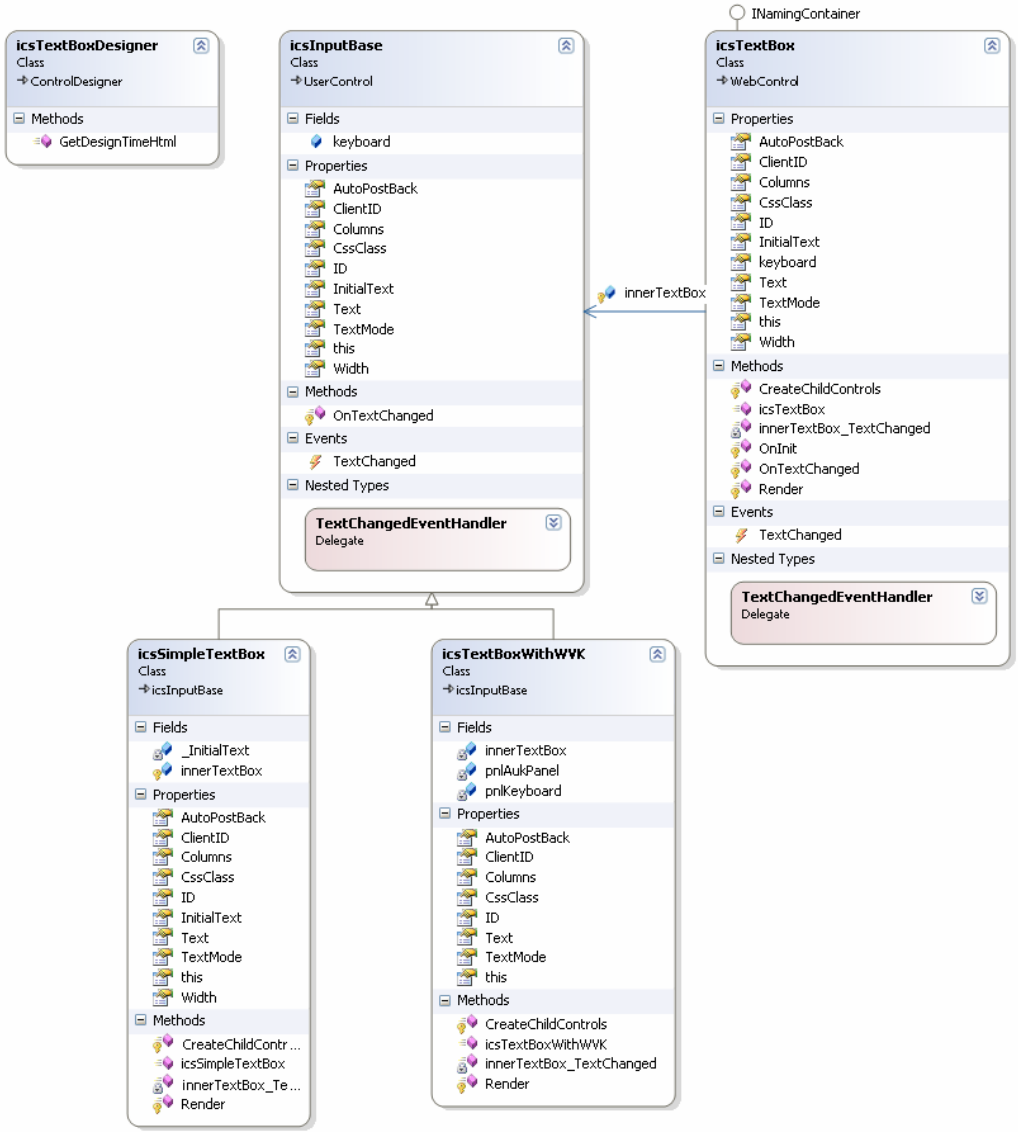


Figure 241: **Textbox diagram**

'icsTextBox' is a web control used by developers in order to place a textbox in a web page. It includes an 'icsInputBase' object that at runtime is instantiated in a simple textbox or in a textbox with a virtual keyboard. The 'icsTextBoxDesigner'

class provides the appropriate means for this base object to be represented graphically.

In the case that a textbox has to be presented along with a virtual keyboard, several web virtual keyboard styles are available. A hierarchy of classes supporting different styles was developed and is presented in Figure 242. The 'WVKTemplateBase' base class exposes the basic functionality of the web virtual keyboard to the derived classes: icsQuertyTemplate, icsWVKMinTap2Template, icsWVKMinTap3Template, icsWVKMinTap5Template, icsWVKRLessTapTemplate, and icsWVKRStandardTemplate. A 'WVKTemplateBase' object is included in the 'icsTextBoxtWVK' class, and is instantiaed at runtime in one of the derived class objects, thus enriching web virtual keyboard with adaptivity.

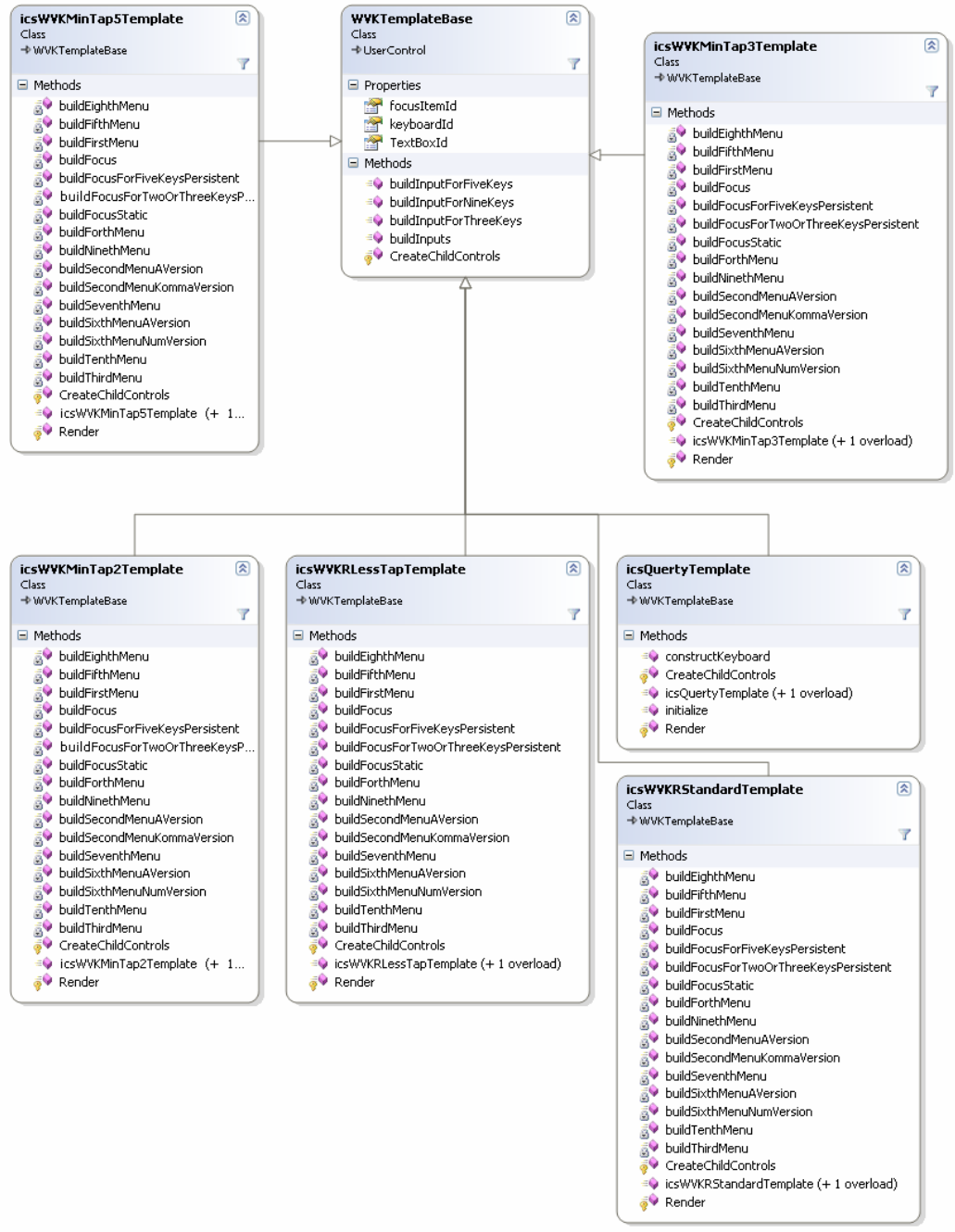


Figure 242: **Web virtual keyboard diagram**

Formatted text entry

Editing web content is a typical task in modern content management systems. This task is usually carried out with the help of an online web editor that simulates desktop word processing applications. For formatted text typing, Appendix A presents a number of variations facilitated by EAGER through a polymorphic hierarchy of text editors presented in Figure 243.

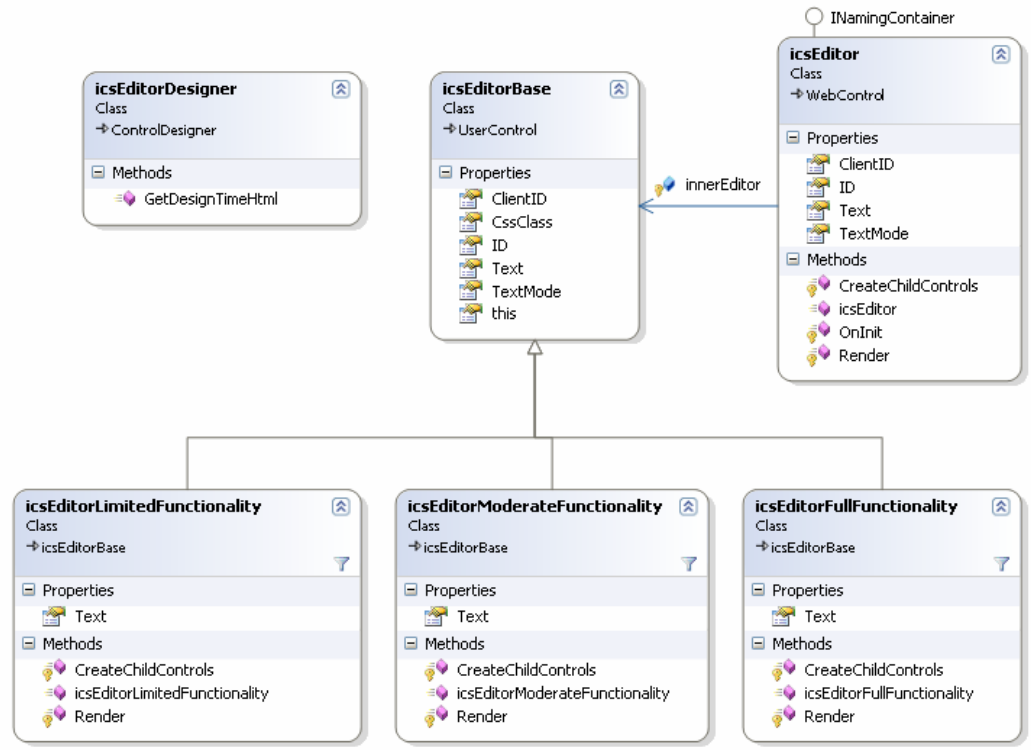


Figure 243: **Editor Diagram**

'icsEditorBase' includes properties and methods that are inherited by the derived classes. The 'icsEditor' web control encapsulates an 'icsEditorBase' object that at design time is represented in the Visual Studio designer with the help of the class 'icsEditorDesigner', and is instantiated at runtime in any of the derived classes object.

Date entry

Selecting dates is a frequent task carried out by users of web applications. As discussed in Appendix A, a number of variations in style can be facilitated according to specific user characteristics. In order for these variations to be supported in EAGER, the date pickers class hierarchy was developed as presented in Figure 37.

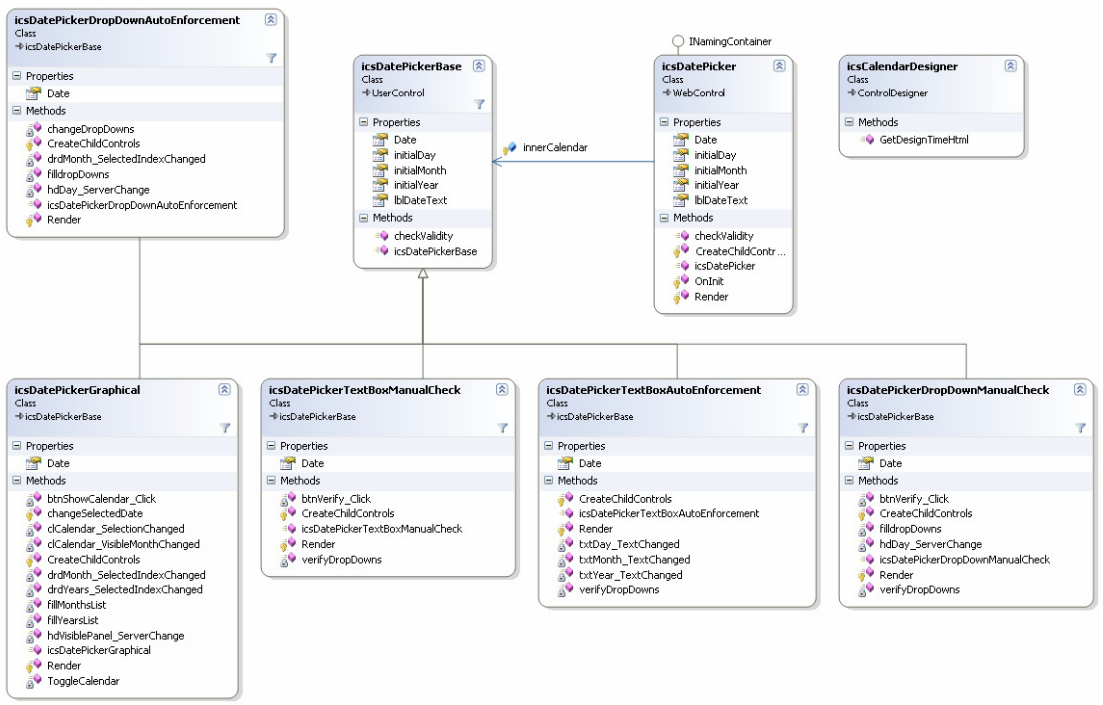


Figure 244: **Date picker diagram**

The 'icsDatePickerBase' constitutes the base class of the Date pickers hierarchy, including methods and properties appearing in Table 18, and being used and extended by the derived classes to meet individual user and contexts of use needs.

Upload files

Modern web applications typically introduce facilities, such file sharing downloads, that mainly facilitate the concept of a common file system for a wide number of users. In this context, the process of uploading files is of particular importance, especially for users with limited experience with using web applications. In EAGER, the different file uploaded styles identified are facilitated thought the file up loaders polymorphic hierarchy presented in Figure 36. The 'icsFileUploaderBase' class, acting as a parent of all file up loaders, incorporates all the appropriate methods and functionality (see Table 16) to be shared among the derived classes. Several derived classes inherit from the base class in order to match different user and contexts of use needs and are summarised in Table 17. 'icsFileUploader' is used by the developers in order to place a file uploader object in a web page. The procedure from the systems' perspective is that 'icsFileUploader' includes an 'icsFileUploaderBase' object that at design time is presented in a designer following the 'icsFileUploaderDesigner' guidelines, and at runtime is transformed in an object of the derived classes depending on user and context of use attributes.

Figure 245: **File up loader diagram**

Upload images

Image uploading, although similar to file uploading, should be facilitated separately due to the different presentation requirements of images. The styles for uploading images as identified in Appendix A were developed in the form of a polymorphic image uploaders hierarchy presented in Figure 246. The base class for all image uploaders, namely 'icsImageUploaderBase', includes methods and properties to be shared, altered and used among the derived classes. The 'icsImageUploader' web control includes an 'innerImageUploader' that exposes all the appropriate functionality of the 'icsImageUplaoderBase'.

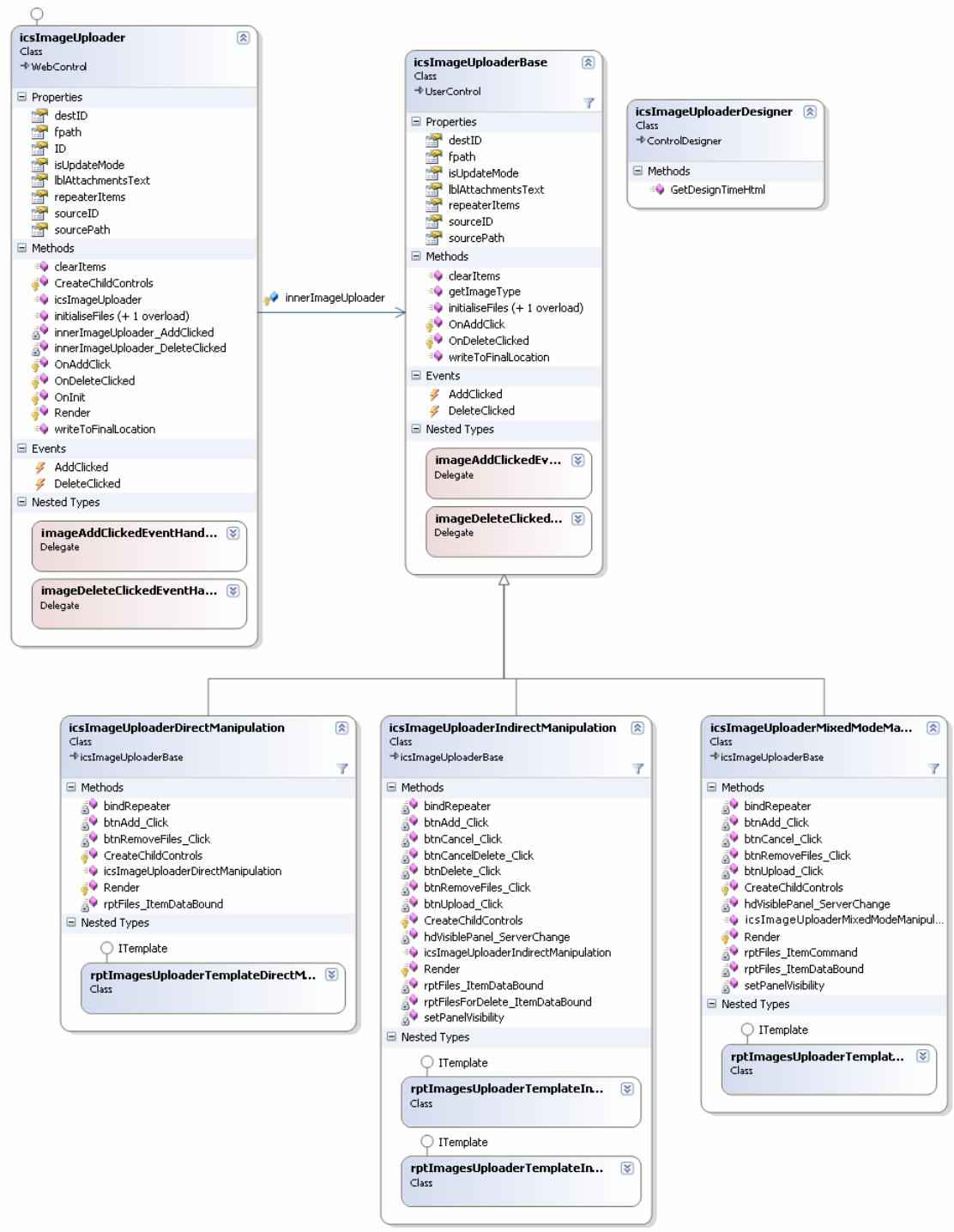


Figure 246: **Image uploader diagram**

File displayer

Web portal usually include lists of downloadable files. As discussed in Appendix I, the amount of information provided to end user varies according to their expertise. To overcome this problem, different presentation schemes for the file displayed were proposed. These schemes are facilitated through the file displayer UI hierarchy, presented in Figure 247.

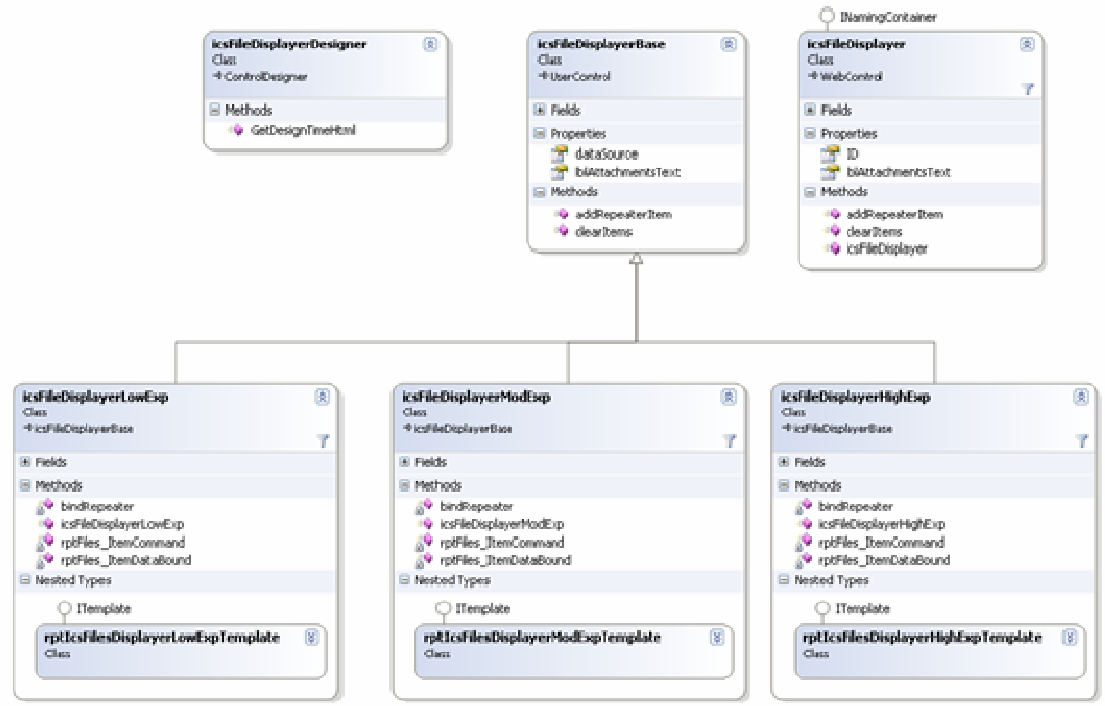


Figure 247: **File displayer diagram**

The file displayer hierarchy includes the base class along with the classes deriving from it. The basic functionality provided by 'icsFileDispalyerBase' class acts as a common application interface to be individualized by each derived class.

The 'icsFileDisplayerDesigner' that appears in Figure 247 encapsulates the appropriate functionality facilitating the 'icsFileDisplayerBase' instances to be presented to the Visual Studio designer. 'icsFileDisplayer' constitutes a web control that contains an instance of the 'icsFileDisplayerBase' class exposing all the required functionality for the file displayer, and is instantiated at runtime to the appropriate derived class object according to certain user and context parameters.

Image displayer

Web portals include image lists too. In Appendix A, alternative variations for presenting images are identified, based not only on the expertise of a given user, but also on specific interaction requirements. To facilitate these alternatives, the image displayers UI controls hierarchy was developed, as presented in Figure 248.

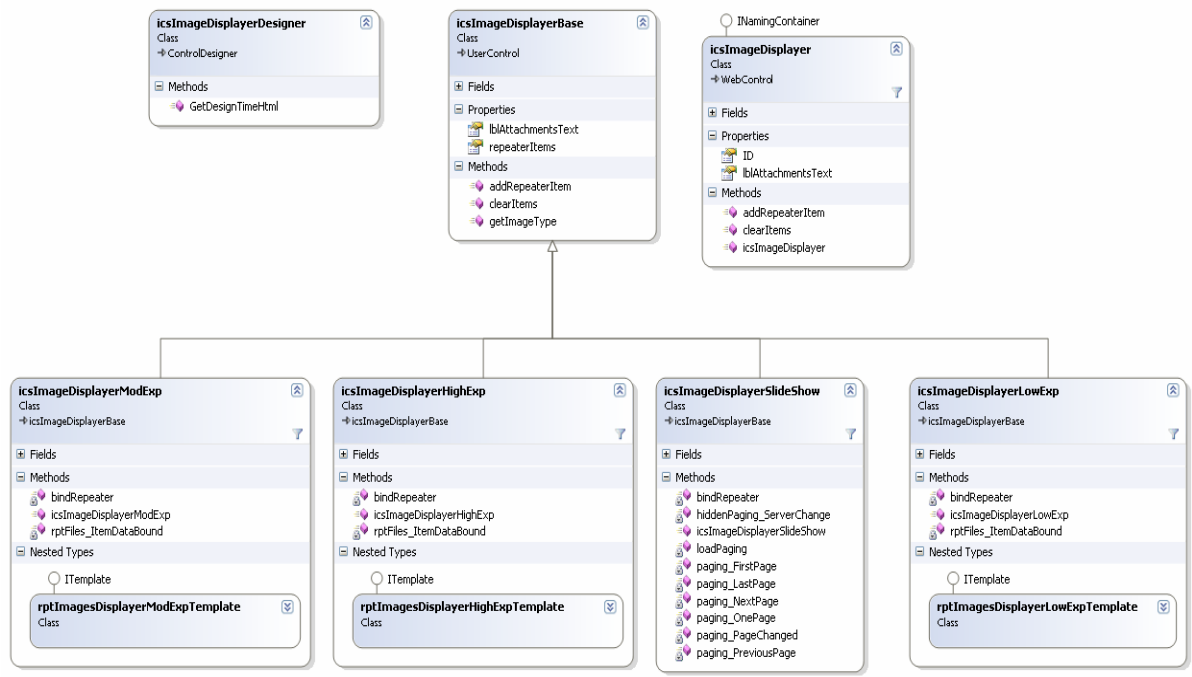


Figure 248: **Image displayer diagrams**

The 'icsImageDisplayerBase' class includes the basic properties and methods (see Table 12) that are implemented by the derived classes (see Table 13) in order to generate special image displayers, as shown in Figure 249.

The 'icsImageDisplayerDesigner' class renders in html the 'icsImageDisplayerBase' control for design time support. In order for a developer to add an image displayer to a web application, the 'icsImageDisplayer' control is offered from the Visual Studio toolbox area which encapsulates an 'icsImageDisplayerBase' object, and is instantiated to a distinct object of the derived classes as necessary in each case.

Search

The way that search is represented may vary depending on the specific user preferences. In some cases, users prefer to use a simple search facility, where in other cases they prefer to use an advanced version to narrow the results. To support these kinds of variations, a hierarchy for representing alternative search variations was developed and incorporated in EAGER. The basic class of the hierarchy, namely icsSearchBase, exposes the functionality to be implemented by its descendors. According to the user and context specific attributes an instance of the base class can be instantiated as any of the hierarchy members presented.

The class diagram of the search variations hierarchy is presented in Figure 249.

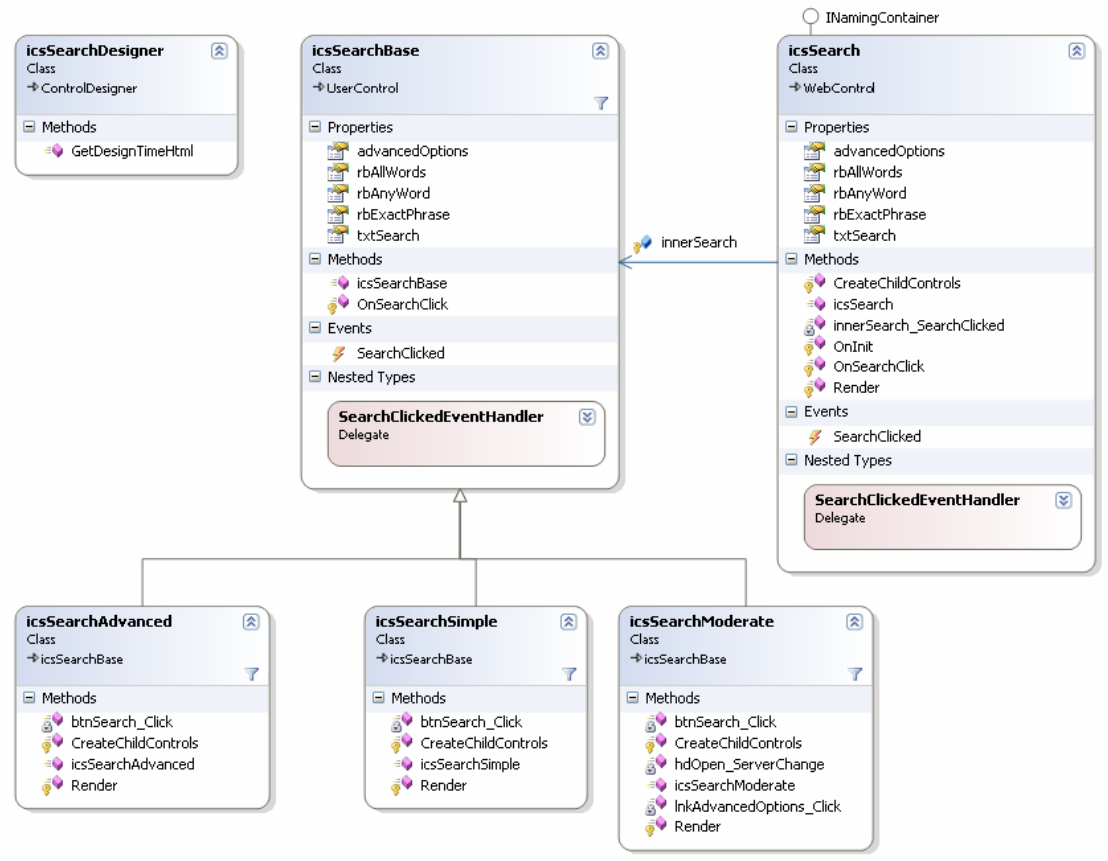


Figure 249: **Image uploader diagram**

Appendices References

Arditi, A., & Knoblauch, K. (1996) Effective colour contrast and low vision. In B. Rosenthal and R. Cole (Eds). Functional Assessment of low vision. St. Louis, Mosby, 129-135.

MacKenzie, I. S., & Soukoreff, R. W. (2002) Text Entry for Mobile Computing: Models and Methods, Theory and Practice. Human-Computer interaction, 17, pp. 147–198.

Mourouzis, A., Ntoa, S., Boutsakis, E., Kartakis, G., & Stephanidis, C. (2005) Expert-based assessment of the ARGO Web browser for people with disability. In Proc. of 8th European Conference for the Advancement of Assistive Technology in Europe (AAATE 2005), Lille - France, 6-9 September, pp. 767-771.

Mourouzis, A., Boutsakis, E., Ntoa, S., Antona, M. & Stephanidis, C. (2007) An Accessible and Usable Soft Keyboard. In Volume 6 of the Proceedings the 12th International Conference on Human-Computer Interaction (HCI International 2007), LNCS_4555, Beijing, China, 22-27 July, (ISBN: 978-3-540-73280-8)

Nielsen, J. (2006): Screen Resolution and Page Layout. Available electronically at: http://www.useit.com/alertbox/screen_resolution.html.

Norman, D. A., & Fisher, D. (1982). Why alphabetic keyboards are not easy to use: Keyboard layout doesn't much matter. Human Factors, 24(5), 509-519.

Pavlovych, A., & Stuerzlinger, W. (2003). Less-Tap: A Fast and Easy-to-learn Text Input Technique for Phones. In Proc. of Graphics Interfaces 2003, pp. 319-326.

Vanderheiden, G.C., Chisholm, W.A., & Ewers N. (1996) Design of HTML Pages to Increase their Accessibility to Users with Disabilities: Strategies for Today and Tomorrow. Available electronically at: http://trace.wisc.edu/archive/html_guidelines/version6.8.html.