UNIVERSITY OF CRETE

SCHOOL OF SCIENCES AND ENGINEERING

COMPUTER SCIENCE DEPARTMENT

A DEVELOPMENT LIBRARY FOR THE AUTOMATIC USER INTERFACE ADAPTATION OF WEB PORTALS ON PDA DEVICES

by
DIMAKIS ANDREAS

Master's Thesis

Heraklion, November 2007

UNIVERSITY OF CRETE
SCHOOL OF SCIENCES AND ENGINEERING
COMPUTER SCIENCE DEPARTMENT

A DEVELOPMENT LIBRARY FOR THE AUTOMATIC USER INTERFACE ADAPTATION OF WEB PORTALS ON PDA DEVICES

by
DIMAKIS ANDREAS

A thesis submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Author:

Dimakis Andreas, Computer Science Department

Supervisory Committee:

Constantine Stephanidis, Professor, Supervisor

Evangelos Markatos, Professor, Member

Christos Nikolaou, Professor, Member

Approved by:

Panagiotis Trahanias, Professor
Chairman of the Graduate Studies Committee

Heraklion, November 2007

A DEVELOPMENT LIBRARY FOR THE AUTOMATIC USER INTERFACE ADAPTATION OF WEB PORTALS ON PDA DEVICES

DIMAKIS ANDREAS

Master's Thesis

University of Crete
School of Sciences and Engineering
Computer Science Department

Abstract

In recent years, handheld devices are evolving from Personal Digital Assistants (PDAs) to direct competitors of laptop and desktop computers. The market for mobile platforms is growing two times faster than the market for traditional desktop computers. This diffusion of handheld devices allows users to access the Internet anytime and anywhere. However, most pages are designed with only a desktop audience in mind. This fact raises the need for automatic user interface adaptation mechanisms capable of providing mobile interfaces equally usable and functional with respect to their desktop versions.

This thesis proposes an automatic user interface adaptation mechanism that provides usable and accessible mobile interfaces. Two sets of parameters are taken into consideration during the adaptation process, the user characteristics (e.g., web familiarity, disability) and the context of use (e.g., device). The outcome is a consistent interface in which the final form of the elements of each page encapsulates interface design knowledge. The adaptation process is transparent to the end user, and provides graceful transformation of the navigation mechanism and the content of the adapted portal. Furthermore, the produced mobile interface preserves all the content and functionality of the desktop version.

As part of this thesis, a software library of adaptable interface artifacts with several alternative presentation styles was developed. These styles address different contexts of use, including, among others, accessible alternatives for blind and color blind users. A fully functional prototype portal was developed as a case study by means of this library and was evaluated. Two types of evaluation methods were used, a heuristic evaluation with design experts and a subjective evaluation with end users. The results

indicated that the overall user experience and the interface quality are quite satisfying for the end users.

Supervisor: Constantine Stephanidis, Professor

ΜΙΑ ΒΙΒΛΙΟΘΗΚΗ ΓΙΑ ΤΗΝ ΑΥΤΟΜΑΤΗ ΠΡΟΣΑΡΜΟΓΗ ΔΙΕΠΑΦΩΝ ΔΙΑΔΙΚΤΥΑΚΩΝ ΠΥΛΩΝ ΣΕ ΥΠΟΛΟΓΙΣΤΕΣ ΠΑΛΑΜΗΣ

ΔΗΜΑΚΗΣ ΑΝΔΡΕΑΣ

Μεταπτυχιακή Εργασία

Πανεπιστήμιο Κρήτης
Σχολή Θετικών Επιστημών
Τμήμα Επιστήμης Υπολογιστών

Περίληψη

Τα τελευταία χρόνια, οι φορητές υπολογιστικές συσκευές εξελίσσονται από Προσωπικοί Ψηφιακοί Βοηθοί (PDA) σε ανταγωνιστές των φορητών υπολογιστών και των υπολογιστών γραφείου. Η εμπορική κίνηση της αγοράς αυτών των συσκευών μεγαλώνει με διπλάσιους ρυθμούς από ότι αυτή των υπολογιστών γραφείου. Η εξάπλωση αυτή επιτρέπει στους χρήστες την πρόσβαση στο διαδύκτιο από οπουδήποτε και οποτεδήποτε. Παρόλα αυτά, οι περισσότερες διαδυκτιακές σελίδες κατασκευάζονται έχοντας υπόψη μόνο τις ανάγκες χρήσης σε περιβάλλον γραφείου. Το γεγονός αυτό δημιουργεί την ανάγκη για ανάπτυξη αυτόματων μηχανισμών προσαρμογής οι οποίοι θα παρέχουν διεπαφές για κινητές συσκευές εφάμιλλης ευχρηστίας και λειτουργικότητας με τις διεπαφές που προορίζονται για χρήση στους υπολογιστές γραφείου.

Η παρούσα εργασία προτείνει έναν αυτόματο μηχανισμό προσαρμογής διεπαφών ο οποίος παρέχει εύχρηστες (usable) και προσβάσιμες (accessible) διεπαφές για φορητές υπολογιστικές συσκευές. Κατά την διαδικασία της προσαρμογής, δύο ομάδες χαρακτηριστικών λαμβάνονται υπόψη: τα χαρακτηριστικά του χρήστη (π.χ., βαθμός εξοικείωσης με το διαδύκτιο, αναπηρία) και το περιβάλλον χρήσης (π.χ., φορητή συσκευή). Το αποτέλεσμα είναι μια διεπαφή τα δομικά στοιχεία της οποίας έχουν κατασκευαστεί με συγκεκριμένα σχεδιαστικά κριτήρια. Η συνολική διαδικασία της προσαρμογής είναι αόρατη στο χρήστη και παρέχει ομαλό μετασχηματισμό του μηχανισμού πλοήγησης και του περιεχομένου της διαδικτυακής πύλης. Επιπλέον, η παραγόμενη διαδυκτιακή διεπαφή διατηρεί πλήρως την λειτουργικότητα και το περιεχόμενο της αρχικής διεπαφής.

Στα πλαίσια της εργασίας αυτής αναπτύχθηκε μια βιβλιοθήκη προσαρμόσιμων τμημάτων διεπαφής με εναλλακτικές μορφές παρουσίασης για το κάθε ένα. Οι διαφορετικές αυτές μορφές παρουσίασης αναφέρονται σε διαφορετικά περιβάλλοντα χρήσης και περιλαμβάνουν, μεταξύ άλλων, προσβάσιμες εκδόσεις για τυφλούς και άτομα με προβλήματα όρασης. Για τις ανάγκες της αξιολόγησης του συστήματος, αναπτύχθηκε μια πλήρως λειτουργική πρωτότυπη πύλη με τη χρήση της βιβλιοθήκης αυτής. Χρησιμοποιήθηκαν δύο μορφές αξιολόγησης, μια Ευρετική (Heuristic) αξιολόγηση με ειδικούς στη σχεδίαση διεπαφών και μια Υποκειμενική (Subjective) αξιολόγηση με τελικούς χρήστες. Τα αποτελέσματα έδειξαν ότι η συνολική εντύπωση από την χρήση της διεπαφής, καθώς και η ποιότητά της, είναι πολύ ικανοποιητικές για τους τελικούς χρήστες.

Επόπτης: Κωνσταντίνος Στεφανίδης, Καθηγητής

Acknowledgements

I am grateful to my supervisor, Professor Constantine Stephanidis, for his advice and support. I would also like to thank the Institute of Computer Science, Foundation of Research and Technology (ICS-FORTH) and especially the Human Computer Interaction Laboratory for the graduate fellowship.

I am also very thankful to Dr. Margherita Antona, for her valuable comments in my thesis report, Mr. Alexandros Mourouzis, for his interesting suggestions regarding the interface design process. Ms. Constantina Doulgeraki and Mr. Nikolaos Partarakis for their assistance, collaborative spirit and support during the development and implementation of this research.

I would also like to thank my friends for sharing their creative comments and my worries.

Finally, I would like to thank my family for supporting me all these years.

Table of Contents

List of Figures .....
List of Tables .....
1. Introduction 1
1.1 Challenges 2
1.2 Motivation 3
1.3 Thesis statement 4
1.4 Thesis outline 4
2. Background & Related Work 7
2.1 Adaptation 7
2.1.1 Type of Adaptation 7
2.1.2 Stage of Implementation 9
2.1.3 Enhanced Characteristics 12
2.2 Classification of Page Layout Adaptation Mechanisms 13
2.2.1 Device Specific Authoring 14
2.2.2 Transcoding & Automatic Re-authoring 16
2.2.3 Extended Client-side Capabilities 24
2.3 Summary 27
3. Unified User Interfaces 29
3.1 UniMob 29
3.2 Unified User Interfaces Design Methodology 31
3.3 Unified User Interface Architecture 34
3.4 UniMob - EAGER Relation 37
3.4.1 Dialogue Patterns Component 37
3.4.2 User Information Server 37
3.4.3 Context Information Server 39
3.4.4 Decision Making Component 39
3.5 Summary 40
4. Unified Design of UniMob's Interactive Artifacts 41
4.1 Designing the Mobile Styles 41
4.1.1 Polymorphic Task Hierarchies 42
4.1.2 Content Access Styles 42
4.1.3 Navigation Styles 54
4.1.4 Design Rationale 57
4.2 Implemented Mobile Styles 63
4.3 Summary 86
5. Case Study: The EDeAN mobile portal 87
5.1 Introduction 87
5.2 EDeAN Mobile User Interface 88
5.3 Discussion 101
6. Evaluation & Results 103
6.1 Methodology 103
6.2 Heuristic Evaluation 104
6.3 Subjective Evaluation 106
6.4 W3C Recommendations Conformance 110
6.5 Summary 111
7. Conclusion & Future Work 113
7.1 Conclusion 113

7.2 Future Work 114
Appendix A: Heuristic Evaluation Checklist 117
Appendix B: Usability Evaluation Scenarios and IBM Questionnaires 119
Bibliography .....

List of Figures

Figure 2.1: The IMDB site as it appears in a desktop PC. 20
Figure 2.2: The IMDB site as it appears in the Opera mobile browser. 21
Figure 3.1: Polymorphic Task Hierarchy. 32
Figure 3.2: Polymorphic Task decomposition process. 33
Figure 3.3: Structural components of the Unified User Interface architecture. 36
Figure 3.4: Favorites control (adaptivity). 39
Figure 4.1: Diagrams used in polymorphic task hierarchies. 42
Figure 4.2: Polymorphic Task Hierarchy of "View Image" task. 43
Figure 4.3: Polymorphic Task Hierarchy of "View Table" task. 44
Figure 4.4: Polymorphic Task Hierarchy of "View Paged Content" task. 44
Figure 4.5: Polymorphic Task Hierarchy of "Select Date" task. 45
Figure 4.6: Polymorphic Task Hierarchy of "Select Time" task. 46
Figure 4.7: Polymorphic Task Hierarchy of "Upload File" task, "Direct" and 'Mixed" styles. 46
Figure 4.8: Polymorphic Task Hierarchy of "Upload File" task, "Indirect" style. 47
Figure 4.9: Polymorphic Task Hierarchy of "Upload Image" task, "Direct" and 'Mixed" styles. 47
Figure 4.10: Polymorphic Task Hierarchy of "Upload Image" task, "Indirect" style. 48
Figure 4.11: Polymorphic Task Hierarchy of "View Functions" task. 49
Figure 4.12: Polymorphic Task Hierarchy of "Display Files" task. 49
Figure 4.13: Polymorphic Task Hierarchy of "Display Images" task. 50
Figure 4.14: Polymorphic Task Hierarchy of "View Options" task. 50
Figure 4.15: Polymorphic Task Hierarchy of "View Fieldset" task. 51
Figure 4.16: Polymorphic Task Hierarchy of "View Tabs" task. 51
Figure 4.17: Polymorphic Task Hierarchy of "Display List" task. 51
Figure 4.18: Polymorphic Task Hierarchy of "Display Paged Text" task. 52
Figure 4.19: Polymorphic Task Hierarchy of "Search" task, "Simple" and "Advanced" styles. 53
Figure 4.20: Polymorphic Task Hierarchy of "Search" task, "Moderate" style. 53
Figure 4.21: Polymorphic Task Hierarchy of "View Pie/Bar Chart" task. 54
Figure 4.22: Polymorphic Task Hierarchy of "Display Path" task. 55
Figure 4.23: Polymorphic Task Hierarchy of "Display Categories" task. 55
Figure 4.24: Polymorphic Task Hierarchy of "Display Subcategories" task. 55
Figure 4.25: Polymorphic Task Hierarchy of "Display Options" task. 56
Figure 4.26: Polymorphic Task Hierarchy of "Display Top Menu" task. 56
Figure 4.27: Polymorphic Task Hierarchy of "Display Bottom Menu" task. 56
Figure 4.28: Class hierarchy of the "Select Date" styles implementation. 64
Figure 4.29: Alternative styles of the "View Image" task. 65
Figure 4.30: Alternative Adapted Image styles, "Pure Black-White", "Grayscale" and "Color". 66
Figure 4.31: Displaying modes of the "Thumbnail" style. 67
Figure 4.32: Alternative styles of the "View Table" task. 68
Figure 4.33: Alternative styles of the "View Paged Content" task. 69
Figure 4.34: Alternative styles of the "Select Date" task. 70
Figure 4.35: Alternative styles of the "Select Time" task. 71
Figure 4.36: Alternative styles of the "Upload Files" task. 72
Figure 4.37: Alternative styles of the "Upload Images" task. 73

Figure 4.38: Alternative styles of the “View Functions” task. 74
Figure 4.39: Alternative styles of the “Display Files” task. 74
Figure 4.40: Alternative styles of the “Display Images” task. 75
Figure 4.41: Alternative styles of the “View Options, Fieldset and Tabs” task. 76
Figure 4.42: Alternative styles of the “Display List” task. 77
Figure 4.43: Alternative styles of the “Display Paged Text” task. 79
Figure 4.44: Alternative styles of the “Search” task. 80
Figure 4.45: Image and Table styles of the “View Pie/Bar Chart” task. 81
Figure 4.46: Alternative Image styles of the “View Pie/Bar Chart” task. 81
Figure 4.47: The “Display Path” style. 82
Figure 4.48: Alternative styles of the “Display Categories” task. 83
Figure 4.49: Alternative styles of the “Display Top Menu” task. 84
Figure 4.50: The template being used in the mobile adaptation. 85
Figure 4.51: “Navigation Search” and “Favorites” components. 85
Figure 5.1: Navigation hierarchy of the EDeAN portal in PDA. 88
Figure 5.2: Navigation areas in the EDeAN portal. 89
Figure 5.3: Logged out home page in desktop EDeAN. 89
Figure 5.4: Logged out home page in mobile EDeAN. 90
Figure 5.5: Logged in home page in desktop EDeAN. 90
Figure 5.6: Logged in home page in mobile EDeAN. 91
Figure 5.7: Custom Accessibility area in desktop EDeAN. 92
Figure 5.8: Custom Accessibility area in mobile EDeAN. 92
Figure 5.9: Modify Quick Access Links settings in desktop EDeAN. 93
Figure 5.10: Modify Quick Access Links settings in mobile EDeAN. 93
Figure 5.11: Discussion topics of Standardization SIG in desktop EDeAN. 94
Figure 5.12: Discussion topics of Standardization SIG in mobile EDeAN. 95
Figure 5.13: Create new topic in Discussion topics of Standardization SIG in desktop EDeAN. 96
Figure 5.14: Create new topic in Discussion topics of Standardization SIG in mobile EDeAN. 96
Figure 5.15: Statistics of the Resource Center in desktop EDeAN. 97
Figure 5.16: Statistics of the Resource Center in mobile EDeAN. 98
Figure 5.17: General Settings area in desktop EDeAN with disability Blind enabled. 98
Figure 5.18: General Settings area in mobile EDeAN with disability Blind enabled. 99
Figure 5.19: General Settings area in desktop EDeAN with disability Color Blind - Deuteranope enabled. 100
Figure 5.20: General Settings area in mobile EDeAN with disability Color Blind - Deuteranope enabled. 100

List of Tables

Table 2.1: Adaptation parameters. 13
Table 2.2: Page Layout Adaptation Mechanisms. 27
Table 3.1: Communication messages of the Unified User Interface architecture. 36
Table 3.2: Basic characteristics of the user profile. 38
Table 3.3: Extended characteristics of the user profile. 38
Table 4.1: Task “View Image”. 57
Table 4.2: Task “View Table”. 58
Table 4.3: Task “View Paged Content”. 58
Table 4.4: Task “Select Date”. 58
Table 4.5: Task “Select Time”. 58
Table 4.6: Task “Upload File”. 59
Table 4.7: Task “Upload Image”. 59
Table 4.8: Task “View Functions”. 59
Table 4.9: Task “Display Files”. 59
Table 4.10: Task “Display Images”. 60
Table 4.11: Task “View Options”. 60
Table 4.12: Task “View Fieldset”. 60
Table 4.13: Task “View Tabs”. 60
Table 4.14: Task “Display List”. 60
Table 4.15: Task “Display Paged Text”. 61
Table 4.16: Task “Search”. 61
Table 4.17: Task “View Pie/Bar Chart”. 61
Table 4.18: Task “Display Path”. 62
Table 4.19: Task “Display Categories”. 62
Table 4.20: Task “Display Subcategories”. 62
Table 4.21: Task “Display Options”. 62
Table 4.22: Task “Display Top Menu”. 63
Table 4.23: Task “Display Bottom Menu”. 63
Table 6.1: End user characteristics. 108
Table 6.2: Summary of ASQ results. 109
Table 6.3: Summary of CSUQ results. 109
Table 6.4: Default Delivery Context. 111

Chapter 1

Introduction

Web browsing using small screen handheld devices is becoming more and more common. There has been a realization over the last years that handheld devices are evolving from Personal Digital Assistants (PDAs), as they were originally called, to direct competitors of laptop and desktop computers. The market for mobile platforms is growing two times faster than the market for traditional desktop computers. Two practical shortcomings prevented the greater adoption of these devices in the past. The first was limited network speed. The second problem was related to their very nature and special characteristics, such as small screens, limited color displays, different interaction techniques (virtual keyboard, stylus, touch screen, etc.), limited memory, small storage capacity, limited battery life, and reduced processing power. With the rapid and successful development of 2.5G and 3G wireless networks, the bandwidth factor is expected to be less constraining in the near future. Moreover, the remarkable improvement in the hardware capabilities of these devices provides satisfactory solutions to the limitations of processing power, storage capacity and battery life. At the same time, however, the limitations of display size and interaction techniques are likely to remain unchanged for a certain period of time.

With the demand for Internet access increasing for e-mail, shopping, e-commerce, online event schedules and quick up-to-date information, the need to provide interfaces to the PDA users that are equally usable compared to the desktop versions is becoming more and more crucial. Most pages are designed with only a desktop audience in mind. Multi-column layouts preformatted to certain page widths are used, fixed values, too many graphic elements, and more. On a PDA such pages can be hard to read. Usually, they require too much time to load and need scrolling in both axes. Users confined to viewing a small portion of each page lack a sense of the overall context, and are forced to remember the location of items in the page. These are two of the major problems a user experiences when browsing a web site with a PDA

device. Researchers have started to address such problems by developing adaptation mechanisms and procedures.

Adaptation means a process of selection, generation or modification of content to suit to the user's computing environment and usage context. Several adaptation mechanisms are reported in literature for web content manipulation. They can be classified according to the location of the adaptation process (server-side, client-side, proxy), the type of the adaptation (adaptivity, adaptability) and the characteristics of the device that are aimed to be enhanced by the adaptation process (resource specific adaptation, page layout adaptation). Current approaches include: device-specific authoring, transcoding, automatic re-authoring, extended client-side navigation capabilities (thumbnail view, zoom-in/-out, summarized versions, etc.), or special characteristics of recent browsers (e.g., MS Pocket Internet Explorer™ narrows down cells, Opera's Small-Screen Rendering™ reformats the page to fit inside the screen width). Each method has its own scope of application, produces different results and has different pros and cons.

1.1 Challenges

Internet is a very large and heterogeneous environment. This means that the "real" pages are too complex and varied to be successfully adapted in the general case. Many adaptation algorithms depend highly on structural analysis of the web by decomposition. Some success has been achieved in some areas, but generally it is a hard problem because of the nature of reverse engineering. Web pages that use "hidden semantics" - such as tables to control layout or text embedded within images - typically produce unusable results.

The goal of this thesis is to design an adaptation mechanism that does not depend on algorithms of structural analysis. This burden can be overcome, at design time, by separating navigation, content and presentation. In this way, each one of these elements can be separately adapted according to specific needs.

Methods that use extended client side navigation capabilities suffer from excessive scrolling and depend highly on the nature of the content. Summarization and zooming techniques require frequent interactions between the user and the device. The result pages may sometimes be unreadable, depending on the screen resolution of the device, and users tend to lose the overall context. The adaptation mechanism

proposed in this thesis minimizes scrolling and provides a consistent user interface with no redundant interaction steps required.

Early approaches to authoring multiple versions of a web page result in burden on content management and scalability. Despite all its limitations, this method can produce highly usable interfaces. The major disadvantage of most adaptation mechanisms is that the final presentation is usually unpredictable during the designing phase. The level of usability and consistency of the adapted page is not guaranteed. As a result, most pages lose their quality of layout, segmentation and flow. This thesis examines the adaptation process from an HCI point of view and sets as a major challenge to provide a highly usable adapted interface.

1.2 Motivation

The aim of this thesis is to build an adaptation system for portals or generally speaking for web pages organized in hierarchies of components (semantic blocks). Although this approach is mainly targeted to web portals, Chen et al [1] showed that 90% of web sites can be successfully partitioned into semantic blocks. However, algorithms for layout analysis, document modeling and content block detection are out of the scope of this thesis.

The proposed system will be able to provide web interfaces to the mobile users with a high level of usability. The outcome is a consistent interface in which the final form of the elements of each page encapsulates interface design knowledge. This knowledge originates from a usability expert that designed the presentation of the final adapted element. Thus, no crucial design decisions are taken by the software itself.

The adaptation mechanism used takes into consideration three basic parameters in order to build the final interface, a) user characteristics (e.g., level of web familiarity, disabilities), b) context of use, c) system characteristics (e.g., display capabilities such as resolution, color depth). It must be emphasized that the adaptation is not directed to a specific device type or browser, however the system capabilities of each separate device are respected (e.g., by not providing greedy interfaces in terms of page size or CPU utilization).

Three categories of stakeholders related to a web portal are benefited, namely content providers, software engineers of the adaptation mechanism and developers of

new portals. Content providers should update the content of their sites only once, for both the desktop and the mobile version. Software engineers would use a scalable, extensible and fully documented architecture. Finally, the developers of new portals could easily use the components of the proposed adaptation mechanism in order to achieve mobile adaptation for their portals.

A final objective is the adapted interface to comply with the major standards, guidelines and rules in the areas of *usability* (e.g., Nielsen's usability guidelines), *accessibility* (e.g., Web Content Accessibility Guidelines 1.0) and *mobile web* (e.g., Mobile Web Best Practices 1.0).

1.3 Thesis statement

The proposed adaptation system is an application of the *Unified User Interfaces Design Methodology* (U^2ID) [2]. The characteristics of U^2ID are especially useful for the accomplishment of the set objectives. In short, a unified user interface is actually an interface that can automatically adapt to the individual user attributes (e.g., requirements, abilities, preferences), as well as to the particular characteristics of the usage-context (e.g., interaction technology, computing platform, surrounding environment). A detailed presentation of this methodology is provided in Chapter 3 (§ 3.2).

The primary contributions of this thesis are:

- The development and assessment of a methodology for adapting the navigation mechanism of a portal on mobile devices.
- A software library of adaptable Web User Controls (e.g., Table Linearization Control, Paging Control, Image Manipulation Control) for PDA devices, called UniMob.
- An adaptation mechanism that provides the capability to control the presentation of the final adapted interface, thus enhancing usability and functionality.
- A usability evaluation of the mobile adaptation mechanism.

1.4 Thesis outline

The present thesis is organized as follows. Chapter 2 presents a definition and a basic classification of adaptation systems. In addition, the most important adaptation techniques for each category are presented, together with the most representative systems for each one. Chapter 3 presents the foundations of the proposed adaptation mechanism, the Unified User Interfaces Design methodology and Architecture. Chapter 4 presents implementation issues of the adaptation system. In Chapter 5 a case study of development of the mobile version of a fully functional Web portal by means of the proposed adaptation system is presented. Chapter 6 describes the evaluation process used to measure the level of usability, consistency and efficiency of the final adapted interface. Finally, Chapter 7 draws conclusions and discusses future work plans.

Chapter 2

Background & Related Work

This Chapter presents a classification of adaptation mechanisms tailored to mobile devices. In particular, this classification is based on some major adaptation parameters. These parameters are: the type of the adaptation (static vs. dynamic), the stage where the adaptation is implemented and the characteristics to be enhanced or altered (device resources, page navigation, page content). In addition, each adaptation mechanism is briefly described and the most representative studies are presented.

2.1 Adaptation

A mobile adaptation process can be defined as a transformation of a tool's functionality from one configuration to another. The result can be influenced by several factors. These factors are called contextual constraints and describe the information related to the user's situation. Several types of contexts can be identified regarding interaction with a mobile device, including task, social, personal and environment contexts [3]. Connection bandwidth, network reliability, CPU usage, display dimensions, input devices and user preferences can also influence the adaptation process.

Adaptation systems are not classified in a unique way. The classification used here is based on some major adaptation parameters, which will be discussed in the subsequent subsections.

2.1.1 Type of Adaptation

The first parameter discussed is the type of the adaptation process. Before going any further, it is essential to point out the difference between the terms adaptability and adaptivity. *Adaptability* is the modification of aspects of the user interface during initiation of each interaction session, according to user abilities, requirements and preferences that are known prior to interaction and are assumed to remain unchanged

within a single session (e.g., user expertise, user preferences). *Adaptivity* is the modification of aspects of the user interface dynamically, according to user characteristics and events that are detected at run-time. Systems that follow the adaptability pattern are called Adaptable and systems that follow the adaptivity pattern are called adaptive. However in order for a system to be complete both types of adaptation should be supported [4].

Examples of adaptable system are personalized portal sites, such as MyYahoo [5], iGoogle [6], MyExcite [7], etc. These sites are supported by systems that guide users to change the background colors, choose content modules or redesign the page layout. The home page is customized according to user's preferences each time the user visits the portal. This manual customization is a useful tool for web interfaces, but provides minimal degree of automation. The site complexity is further increased as the user is required to learn the use of the manual adaptation component. Additionally, manual customization does not adapt navigation.

Contrary to the adaptable systems, adaptive systems adjust to the habits, working methods and preferences of their users. These systems "learn" from monitoring and analyzing several factors such as user's past browsing behaviour and web access logs. Letizia [8] and WebWatcher [9] are systems that record the hyperlinks the user follows, and the revisits of documents previously encountered or of a document stored in a hot list, and then recommend which of the several possibilities available, concerning the hyperlinks the user could follow, is more likely to match the user preferences. WebWatcher also communicates with the user and collects feedback. The suggested links that best match the interests of the user are marked with a small icon. Additionally, it provides the capability of search with keywords, and offers users commands such as "show me similar pages", "how many users followed this hyperlink", etc. Footprints [10] is a system that uses navigation history mining. Specifically, during the navigation of a user on a site, the system records how often each link is traversed. New users may then be guided to use the most frequently traversed paths as indicators of the most interesting pages. Moreover, users need not provide any information about them in order to take advantage of the system. Footprints provides a variety of tools such as maps, path views, annotations, comments by other users, etc.

The AVANTI Web Browser [11] is an example of a system that combines adaptivity and adaptability. Characteristics gathered prior to interaction include: a)

physical abilities, b) familiarity of the user with: computing, networking, hypermedia, the web, c) the overall usage target: speed, ease, accuracy, error tolerance, d) the language of the user, e) user preferences regarding specific aspects of the application and the interaction. The dynamic user situations that are taken into account concern: a) familiarity with specific tasks (e.g., user's ability to successfully initiate and complete certain tasks), b) ability to navigate using the documents' navigation elements, c) error rate, d) disorientation, e) user idle time, f) repetition of interaction patterns. These characteristics and situations are taken into account during the adaptation of the AVANTI web browser's interface. The result is a unified interface that can adapt itself to suit the requirements of three user categories: able-bodied, blind and motor impaired. The system also provides enhanced history control for blind users, as well as linear and non-linear history visualization for sighted users, link review and selection acceleration facilities, document review and navigation acceleration facilities, intra-document searching facilities, etc.

2.1.2 Stage of Implementation

The second parameter refers to the stage where the adaptation process is implemented. Adaptation can occur at server side, client side and proxy server.

Server-side adaptation offers maximum author control over the delivered content, including the ability to change the amount of content, the styling, navigation and layout. Therefore, transmission times are reduced by delivering already adapted content. This technique has also an important advantage, as it does not exhaust the resources of the mobile device (CPU usage, memory size, battery life). However, in order to produce the most appropriate adaptation, the server must have sufficient information about the delivery context, including the device's capabilities, the user characteristics, the environment, the executed task, etc. These parameters are crucial for systems that perform server side adaptations, thus an efficient and effective mechanism that could be able to collect and utilize such parameters is essential. Methods used to populate the delivery context information include on-line questionnaires, user profiles, system information, client-side scripting languages, User Agent headers analysis, various sensors, user monitoring, log files and much more. There are many techniques for achieving server-side adaptation. In the past, for example, web designers have been delivering different versions of their content to

accommodate nonstandard browser implementations. The author might provide multiple versions of the content and let the server select which to deliver. The Apache Cocoon Server [12] uses an Extensible Stylesheet Language Transformation (XSLT) to generate appropriate delivery markup (e.g., WML) from a common XML content representation. The author might provide several XSL style sheets to generate markup suitable for different delivery devices, the server, based on the delivery context, selects the appropriate style sheet.

The second alternative is the client side approach, where the adaptation occurs in the content delivery device (the web browser). In this model, content authors do not need to create different versions of the content, since the same content is always delivered. Also, an advantage is that the adaptation code usually has direct access to the device's capabilities. Some client side adaptations are independent of content directives. Many browsers, for example, let users increase or decrease text display size. However, this can have unexpected consequences for author-defined layouts. Client side adaptations can also occur based on directives within the content. An example is the use of CSS, which authors often use to style HTML elements. CSS media types are names that identify different device types, such as screen, handheld, TV, print, projection, aural and Braille. In CSS media types, authors can define different styling rules for different media types. For example, a site can use a stylesheet with variable values for foreground and background colors and text fonts, and then let the browser adjust the presentation for conventional screens, printers, handheld devices, televisions, etc. CSS media types can also used to omit display of parts of a web page. Of course, delivering a complete page suitable for any device, then omitting large parts of it, is not an effective use of delivery bandwidth.

A complete category of client side adaptation techniques tailored to mobile devices has been recently developed. This category is named "Extended Client side Capabilities" and will be presented in subsection 2.2.3 in detail. Systems of this category usually are comprised of two parts: a) a proxy server or a server component that takes the standard source code of the web page and transforms-adapts it, and b) a client application that allows the user to view and interact. Many systems follow this model and it can be said that these are not pure client based. A combination of techniques (flip, zoom, pan, focus, resize, multiple views) is usually available at the client side in order to enhance the user's experience with the adapted web page. These

techniques are resource consuming and the tradeoff between device's resources and interaction quality must be carefully considered.

The last option is the proxy based solution. In order to avoid changing either the server that provides content or the client that consumes it, intermediaries in the content delivery chain can offer adaptation. A proxy server analyzes and transcodes the content on-the-fly, before sending the result to the client. The proxy server can also cache the adapted content for later use, so the transformation only needs to be performed once. Intermediate adaptation can shift computational load from content server to proxy servers [13]. One important benefit of the proxy approach is that it is totally transparent to the content providers, they do not have to change the way they author or serve content. However, a drawback is that content providers have no control over how their content will appear to different clients.

Transcoding proxies can transform image formats or even subsets of markup languages. This gives data enabled phones access to web sites by either omitting a server's full-resolution color images or transcoding them into low-resolution or monochrome versions, depending on the phone's displaying capabilities. However, on the fly transcoding is difficult to apply to many media types such as video and audio. Typically proxies lack special information about content, and thus their adaptation abilities are limited. This is usually not a problem for individual images, which typically include resolution and size metadata. Because authors are increasingly marking up content with presentation rather than semantic information, however, it is much harder to transcode anything but trivial markup and obtain an acceptable result. Transcoding HTML into Wireless Markup Language (WML), Compact HTML (cHTML), or Handheld Device Markup Language (HDML) for phones is only possible for some constructs (those who do not use "hidden semantics", such as tables to control layout or text embedded within images). The proxy server solution is mainly used by Transcoding, an adaptation mechanism that will be presented later (§2.2.2) in this chapter. Proxy server adaptation can help reduce origin server loads, but it is only fully successful when it is based on both knowledge of the target device capabilities and author-provided metadata. In practice, however, it has been found that complex or badly designed pages translate poorly or not at all [14]. The technique depends too much on the proxy's ability to interpret the site structure and web sites are just too heterogeneous.

Proxies can be entirely remote or used in a combined local/remote configuration [15]. The local/remote proxy approach has been concerned primarily with bandwidth problems [16, 17], and has addressed screen size problems only partly. Instead, it is the single proxy approach that most frequently has been used to reformat web pages to better match the display capabilities of client devices [18, 19]. This discrimination between the characteristics that will be enhanced by the adaptation process (device resources vs. web page content and layout) constitutes another important adaptation parameter, discussed in the next subsection.

2.1.3 Enhanced Characteristics

The third adaptation parameter is the most obvious and refers to the characteristics that are aimed to be enhanced by the adaptation process. More specifically, the current adaptation systems target either to benefit the resources of the device (resource specific adaptation) or the web page content and layout (page layout adaptation), or both of them.

Most mobile devices face three types of resource constraints: limited processor speed, bandwidth and power supply. With technology continuing to advance at such a quick speed, the processor power has already exceeded the 600 MHz (Dell Axim X30 Wireless, 624MHz), and an average PDA device has a CPU of 300-400 MHz processing power. Although the currently available CPU power in PDAs can not be compared with the CPUs of the desktop computers, this factor is not currently the major burden to adaptation techniques. However, the fact that faster CPU means less battery life should not be overlooked.

Bandwidth adaptation has been studied extensively over the last decade, since this was a major problem at the first steps of PDA networking. Some interesting ideas in this area include disconnected operation [20], distributed adaptation [21], on-demand dynamic distillation [22], visual proxy [23] and other [24, 25, 26, 27]. Technically speaking, a system that adapts bandwidth will almost certainly conserve battery for small devices. This is because one of the heavy power consumers is the Network Interface Card (NIC). However, a lot of work has been done that specifically targets at conserving energy expense on handheld devices. Typical solutions focus on improving the efficiency of the CPU, or optimizing the energy consumption for LCD display [28, 29, 30, 31, 32].

The page layout adaptation techniques deal with the optimization of the content and the structure of a web page under the constraint of limited screen size. Section 2.2 is dedicated to the classification, presentation and discussion of techniques that fulfill this target. Briefly, there are three categories: a) device specific authoring, b) transcoding and automatic re-authoring and c) extended client-side capabilities. The studies of each category propose completely different approaches to the adaptation process and use several techniques to address this problem, including summarization, overview + detail, fisheye views, single column, pan, zoom, focus, transcoding of multimedia content (images, video, audio), etc. This thesis proposes an adaptation technique that falls in this category.

Table 2.1 summarizes adaptation parameters as described in the above sections.

Table 2.1: Adaptation parameters.

Type of Adaptation	Stage of Implementation	Enhanced Characteristics
static (adaptability)	server side	device resources (resource-specific adaptation)
dynamic (adaptivity)	client side	web page content & structure (page layout adaptation)
	proxy server	

2.2 Classification of Page Layout Adaptation Mechanisms

This section presents an overview of page layout adaptation mechanisms. These mechanisms are divided in the following indicative categories: a) device specific authoring, b) transcoding and automatic re-authoring and c) extended client-side capabilities. The adaptation parameters examined in section 2.1 are not specific for the systems of one category, hence a system can be adaptive-proxy, while another system of the same category could be adaptive-server or adaptable-proxy and all of them focus to the adaptation of the content and/or the structure of a web page.

Despite the fact that the major outcome of the adaptation process is an enhanced user interface, several resources of the target device are also affected. For example, a transcoding mechanism that converts a video file into a sequence of images conserves valuable wireless network bandwidth. A summarization technique can reduce the size of a page of than 50% of the initial size. However, usually the systems of the third

category (extended client-side capabilities) may have negative effects on the resources of the mobile device, especially the CPU utilization, because of the visualization algorithms-techniques they use (zoom, pan, focus, fisheye views). Some systems are able to run only on desktop emulators [33]. It must be stated however that this is not the rule for the majority of the "extended client-side capabilities" adaptation systems.

2.2.1 Device Specific Authoring

Device specific authoring involves authoring a set of web pages for a particular display device. The rationale underlying this approach is that a professional web designer re-engineers the web site for each individual device, based on their screen size, resolution, input facilities, etc. The different capabilities of these devices have led to the development of a variety of parallel webs, each one accommodating a particular device feature. This situation poses a serious problem to web designers, since every piece of information needs to be authored for every type of terminal with the associated protocols (e.g., WML, cHTML). Also, it is hard to predict what devices will emerge in the market and if there is a new one introduced, a large amount of repetitive software development becomes necessary. Due to the mass amount of work involved, this technique can only apply to a small set of web sites [34], and hence is limited in scalability. The obtained results are of high quality but, due to its nature, this method is non-generic and can not be applied to already existing pages. Thus the practical applicability of this approach is limited. Another inconvenience is that the visitors of these sites are forced to use different URLs depending on the device type.

The content creators usually produce a number of versions of a particular content item (the cost of maintaining different content bases can be enormous). Each version is typically a transcoded version of the original content. For example, a movie web site can feature the same movie trailers at 3 different resolutions (e.g., low, medium, high), where each clips are transcoded versions of the original high quality content. The various versions of the same content are typically created in such a manner that they are tailored for consumption on particular types of devices. For example, the high resolution movie trailer can be served to desktop, whereas the medium and low resolution versions can be served to PDAs and cell phones, where bandwidth is very limited. In general, the creators of the web sites content create different versions that are optimized for the devices that they are interested in supporting. The fact that

specific content has to be created for every class of devices prior to access by that device is a major disadvantage of this method.

A slightly different approach is authoring for a range of target devices, just as described before for the content. Mappings from a single source document to a set of rendered documents are defined to cover the devices within the range. An example of this approach is the StretchText system [35], in which portions of a document can be tagged with a "level of abstraction" measure. Upon receiving the document, users can specify the level of abstraction they wish to view and are presented with the corresponding detail. Another example is HTML cascading style sheets (CSS) [36]. In CSS, a single style sheet defines a set of display attributes for different structural portions of a document. A series of style sheets may be attached to a document, the user can specify a stylesheet, the browser (the "default" style sheet) and also the author. Although the author's style sheets normally override the user's, the user can selectively enable or disable the author's, providing the ability to tailor the rendering of the document to their particular display.

An example of a web site that follows the device specific approach is the CNN web site, in which the designers have taken steps to design different layout versions of their site that fit in different screen sizes [37]. A subset of the available desktop services is also available in the mobile version (top stories, business, sport, entertainment, search, weather, etc.) [38]. BBC mobile [39] adopts an equivalent approach, several adapted services are available in the mobile version of the site (News, Sport, MyClub, Entertainment, Movies, Lottery, Traffic, Weather, What's on TV/Radio, etc.).

Google mobile [40] has offered its services to PDA users for some time. Five results are displayed at a time, and a compact page summary is associated with each item in the result list. However, when a page is selected, the user is still led to the full page, possibly having to scroll in both directions to view information. No pre-processing of the result pages is carried out. In addition, compact page summaries provide very little information for users to decide which link to follow.

Kirda et al. [41] followed an XML-XSL approach. The first step is to create many versions of the same web content at different resolutions, annotate them by using XML and then specify which version of the content to display and the appropriate layout via the associated stylesheet language, in this case the eXtensible Stylesheet Language (XSL). Unwired Planet's UPLink service [42] uses HDML (Handheld

Device Markup Language) technology to deliver only essential content from the web to small displays on wireless phones. Users of such specialty devices will only have access to a set of predefined services, and the pages for these services can all be designed up-front for the device's particular display. Among services available is public data like stock quotes (Bloomberg, QuoteCom), traffic and weather reports (Accuweather), directories and movie and flight schedules (Mapquest, The Trip.Com, Shanklin). Corporate information (invoice details, dispatch instructions, work orders, shipping records and price lists) can also be accessed, along with proprietary information normally retrieved through PDAs, such as electronic mail, schedules, calendars, contact names and numbers.

2.2.2 Transcoding & Automatic Re-authoring

This category includes two techniques, namely transcoding and automatic re-authoring. These two techniques share some common characteristics, but can not be considered as one. Many automatic re-authoring systems use multimedia transcoding methods, and some transcoding systems apply transformation techniques similar to these in the automatic re-authoring approaches.

Transcoding

Transcoding [43] aims at the direct manipulation of the HTTP stream [44]. The transcoding engine interprets client device's request, fetches the requested content, adapts it, and sends the adapted version to the client [45]. Format conversion and summarization are techniques for transcoding. Format conversion is the most popular method to convert web content into formats that fit in small mobile devices. Wireless Application Protocols such as Wireless Markup Language (WML) [46] and Handheld Device Markup Language (HDML) [47], or Compact HTML (cHTML) [48] of the popular i-mode service in Japan, belong to this category. These protocols can split original HTML pages into different small cards [49], so that each card can fit into one screen. Links are made between cards to enable browsing. The conversion is not only between language formats but also between multimedia formats. A video can be replaced with a series of images or a lower frame-rate video. Since this is an automated solution, it does not require keeping and maintaining separate content bases, and the conversion can be dynamic and in real time.

However, several weaknesses are associated with the format conversion method. First of all, it does not allow retention of layout and presentation style of the original document. Users are likely to be confused when accessing the same web site on a desktop and a PDA. Secondly, navigation may become extremely difficult after pages are split into cards. One of the criticisms of WAP sites is that they involve too many selections, and there are too many moves between cards for the user to achieve the intended goal. This problem could lead to severe usability issues when a web site involves deep hierarchies. In those situations, the number of clicks an average user needs to perform is too high, which potentially discourages the users from browsing with small devices. Especially complex web pages with large nested table structures are difficult to process and lead to poor quality results [50]. The third and most important weakness of transcoding is that these proxies typically employ static, ad-hoc, content adaptation strategies. A common policy is to scale all images by a fixed factor. Thus, these transcoding proxies fail to dynamically address the variation in the resource requirements of different web documents.

Three representative systems in this category are AvantGo, Cocoon and IBM Websphere Transcoding Publisher.

AvantGo [51] provides access to certain web sites that have been customized for viewing on small screens, typically small volume data such as stock quotes or calendar information. The system consists of a special browser and a content adaptation system that is deployed on a desktop PC. The transcoder at the desktop PC downloads web content and customizes it for the browser (translates HTML and images into other formats, and compresses and converts images to match device characteristics). When the mobile device is “synchronized” with the desktop PC, the transcoded content is transferred to it for offline browsing.

Cocoon [52] is an open-source Java server framework that allows dynamic multi-channel web publishing of XML content using XSLT transformations. By relying on XML to describe content, and XSLT as a means of transforming that content into multiple formats, Cocoon provides a platform for building applications with separation between content, logic and presentation. Cocoon can serve static files as well as dynamically generated responses with wide variety of output formats including XML, XHTML, PNG, JPEG, SVG and PDF. Cocoon includes CC/PP and UAProf (User Agent Profile Specification) support through another open-source product called DELI.

IBM Websphere Transcoding Publisher [53] is server-based software that dynamically translates web content and applications into multiple markup languages and optimizes it for delivery to mobile devices based on user preferences and device capabilities. The product can convert images to links to retrieve images, convert tables to lists, and remove comments and features not supported by the device (e.g. applets and Shockwave files). The product comes with a wide variety of standard transcoders including: a) HTML to WML, HDML, PalmOS HTML, and VoiceXML, b) XML to wide variety of formats through XSLT style sheets, c) JPEG/GIF/WBMP image transcoding and rescaling, natural language translation. The complete transcoding is performed at the server level. As all these techniques are applied at run-time, no (or few) choices are possible regarding the transformations.

Automatic Re-Authoring

Automatic re-authoring involves developing software which can take an arbitrary web document designed for the desktop, along with characteristics of the target display device, and re-author the document through a series of transformations, so that it can be appropriately displayed on the device. The web page is analyzed and the structure and content are extracted, then the page is re-engineered in order to fit the characteristics of the specific device. This usually creates a multi-dimensional document structure from the flat two-dimensional document. The re-authoring process can be performed either on the client, on the server, or an intermediary.

This approach depends on the extraction of the original semantic structure of the content. Some success has been achieved in some areas, but generally it is a hard problem because of the nature of reverse engineering and the diversity of the web. HTML's original aim was to provide a device independent markup language that was based on document semantics. It identified document elements such as headings, paragraphs and lists without specifying presentation. Early on, however, browser developers introduced many ad hoc presentation-specific elements and attributes to HTML that blurred the distinction between semantics and presentation. On top of that, integration of multimedia objects, such as audio, video, games, flash technology, javascript, image maps and the CSS have made the web a combination of content and layout. The introduction of XHTML as a proposed standard by the W3C is expected to support the reverse engineering methods, but of the wide adoption of XHTML can not be expected in the near future.

A possible solution is for the web site to provide meta-information about its content, structure and organization. One way to provide meta-information is to represent the site's content in a formal framework with precisely defined semantics. Strudel web-site management system [54] is an example of this approach. Strudel attempts to separate the information available in a web site from its graphical representation. Instead of manipulating web sites at the level of pages and links, web sites may be specified using Strudel's view-definition language. With all of the site's content encoded, its presentation may be easily adapted.

In order to adapt the presentation of the web page, the adaptation systems alter the presentation of its elements. Generally there are two policies for this task: Rule-based and Constraint based. Rule-based policies use high-level mathematical formulas to drive the adaptation. An example of a rule can be to serve only 30% of the original content in file size of a progressive JPEG object. Constraint-based policy extends rule-based policies by encoding tradeoffs between adaptation strategies. In particular, a constraint captures the relationship between resource utilizations and user satisfaction in mathematical formulas. An adaptation solver adapts content by finding a solution that meets all constraints specified by the user of the system.

In general, automatic re-authoring requires minimum user intervention. In addition, it can take arbitrary web documents and generate the adapted version on-the-fly. Two are the major automatic re-authoring approaches: Single Column/Narrow Layout, and Overview + Detail.

Single Column / Narrow Layout

To avoid the need for horizontal scrolling, some of the recent browsers narrow down cells (MS Pocket Internet Explorer) [55] or rearrange the table structure into a single column to make the page fit the screen width (Opera Small Screen Rendering) [56]. Figures 2.1 and 2.2 show this transformation in action. NetFront browser [57] calls the above approach Smart-Fit Rendering. Similarly, Opera's Medium Screen Rendering identifies a web page's content and adapts its elements to fit on medium-sized screens ranging from PDAs used in landscape mode to low-resolution TV screens [58]. In all solutions, the content is shown in the order it appears in the markup file, and is typically scaled to fit the screen width.

As stated in [59] the main benefit of the narrow layout is that the process of reading text is easier. If a text column was wider than the screen, it would be too hard

to read it in the mobile device. Also, just by scrolling the user has access to the rest content of the page. Of course, horizontal scrolling is fully eliminated (however this results in correspondingly larger amount of vertical scrolling). In addition, some of the elements of the original page are preserved in the narrow view (images, headers, style attributes), in this way the user is not fully unfamiliar with the adapted interface. These elements can be used as landmarks in order to assist the user to locate the requested content in the page tube.

Figure 2.1: The IMDB site as it appears in a desktop PC.

Unfortunately, there are several problems with the narrow layout of web pages. Images that contain detailed information are scaled to fit the screen, and size conversion makes the details of the image difficult to read (e.g., texts in wide images). A second problem concerns data tables (e.g., time tables). A row that does not fit into the screen flows to the next row. This rearrangement makes it difficult for the user to find the interrelations between the data of the table.

On typical web pages, the leftmost column contains a list of links (navigation), and the main content is located in the middle of the page. In the narrow layout, the user has to scroll through the list of links before getting to the main content. This recurrent action in every page of the web site could be very time consuming. Usually, the top

section of a web site is the same through the pages of this site. In the narrow layout all pages look the same as long as the first part of the initial page is the same. The difference is that the user of a desktop PC usually focuses on the content first and then to the navigation, menus, etc. The content in a desktop PC is enough to differentiate the pages of a web site. A similar behaviour in narrow layout is not feasible.

Also the order of the content causes difficulties. It is hard to foresee at which point of the long tube the wanted information is located. The user must locate some landmarks in the content while scrolling. This means that the scrolling can not be too fast. On the other hand, because the content in the narrow layout may be really long, scrolling should be fast. The faster the scrolling, the more probable is for the user to overlook a landmark and to need back and forth scrolling in order to locate it.

Figure 2.2: The IMDB site as it appears after the Small Screen Rendering process of the Opera browser that produces the narrow layout.

Overview + Detail

The Overview + Detail methods split a web page into multiple sections (detail pages) and provide an overview page with links to these sections. The overview page can be either a thumbnail view or a text summary of the web page. The content in the detail pages is transformed in order to fit the device capabilities (e.g., screen dimensions, color support, etc.) Text summarization applies natural language techniques [60] to summarize the textual part of the content. Thumbnail views provide an overview of the web page and allow users to zoom in on content in order to view it in a close-up view.

Overview + Detail interfaces have four basic benefits. First, navigation is more efficient because users may navigate using the overview page rather than using the detail page [61]. Second, the overview page aids users in keeping track of their current position in the information space [62]. Third, the overview page itself might give users task-relevant information. Fourth, overview + detail interfaces give users a feeling of control [63].

An advantage of using text summaries is the simplicity in the display, but the downside is that the visual context (e.g., styles, images) in the original web page is lost. Consequently, the users' browsing experience on desktop PCs can not be utilized on PDAs to help discover the target information. In contrast to text summaries, thumbnail overviews [64, 65, 66, 67] preserve the visual context information of the original web page. Users familiar with a web page on the desktop can often recognize it in thumbnail form.

The key disadvantage of Overview + Detail methods is that users need to go back and forth between the overview of the page and its individual blocks. Every time the user finishes reading one block and wants to continue to the next, first the index page needs to be pointed back, and then the new block needs to be located. In addition, the thumbnail index pages being generated may sometimes be unreadable, depending on the screen resolution of the device or the size of the original page. In such situation, it is impossible for the user to identify the important blocks. As a result, scrolls and pen-taps may not be effectively reduced for large pages.

An additional problem is that summarization always means replacing the real content with an approximation, and the correctness of this approximation may vary depending on many variables. Also text summarization techniques work well when

most of the content on a web page is pure text. However, the method fails when many images are added to the webpage, since these images can not be summarized.

The first work that directly addresses the need of automatic tools for layout adaptation is the Digestor project [68], which was carried out back in 1997. This system provides device-independent access to web pages by focusing on the reduced I/O capabilities of the mobile platforms. This problem is tackled by elision techniques. For example, in a large text document, the headers are elided and transformed into links that give access to the content of the section. An algorithm applies a series of 15 transformations with space consumption as performance measure after each transformation step. If the results are unsatisfactory in terms of space reduction, another transformation technique is tried. The transformations are made through an HTTP proxy. Some examples of transformation methods include text outlining (provision of outlines to paragraphs of text to allow quicker access to content in small devices), image reduction (reduction of graphical content size to appropriate sizes to fit small devices), image elision (elimination of images that do not contribute information to the page), first sentence elision (replacement of each text block with a link of its first sentence or phrase that leads to the original text block).

Power Browser [69] is a system developed by Stanford University. In this system, a proxy fetches web pages on the client's behalf and dynamically generates summary views to be transmitted to the client. These summaries represent both the link structure and contents of a set of web pages, using information about link importance. The Power Browser displays consist of a set of "link descriptions" which are generated heuristically from anchor text, URL structure, or ALT tags, as appropriate to the link. The tree arrangement of the links displays both the user's location and the neighboring navigation environment at all times. One positive consequence is that users can jump to a sibling page with a single action. The Power Browser allows users to specify three sorting schemes for the ordering of the links: original, alphabetical and page ranked. Images are ignored by default (the target device had a low-resolution monochrome display), and an alternative text string (ALT tag) is displayed instead of the image. White space and additional text attributes (color, size, font, etc.) that are visible on standard browsers are also ignored. On the other hand, structural elements (lists, tables, paragraphs) are used to format the text within the view. The system provides shortcuts, animation for the moving of the tree control of links and gestures. This browser works for goal-directed tasks, rather than for recreational

browsing. It also works best on sites that use sets of links, rather than extensive text and images.

WebThumb [70] is a system that uses thumbnail overviews. The thumbnail view shows the page as a desktop browser would, with wrapping and layout identical to that of a typical desktop browser. With the arrow tool, tapping on the thumbnail produces the same result that mouse-clicking does on the desktop. A user can also pick up items from the thumbnail and retain them in a larger, easily readable size. WebThumb uses RSVP (showing of text word-by-word in a fixed focal position) to support reading unlimited text in a limited space. The input mechanisms of the target device (iPaq) are assigned special roles. The hard buttons to the left of the joypad can be used by the user's left-hand thumb to cycle through the tools. The right hand, holding the stylus, can then take action on the interface. This mimics the way humans select objects with the non-dominant hand and take actions on those objects with the dominant hand.

Another thumbnail overview system is Slicing*-Tree [71]. In this system, the original page that does not fit into the screen is transformed into a set of pages, each of which fits into the screen. This transformation is done through slicing the original page iteratively with several factors considered, including the size of the screen, the size of each page block, the number of blocks in each transformed page, as well as the semantic coherence between blocks. The resulting set of transformed pages forms a multi-level tree structure, called a slicing*-tree, in which an internal node consists of a thumbnail image with hyperlinks and a leaf node is a block from the original web page. Through this transformation, the contextual information in the original web page is preserved, thus reducing page scrolling effort. The text summary method can be used complementarily to embedding a text summary for each leaf block.

2.2.3 Extended Client-side Capabilities

In extended client-side navigation, the user is offered the possibility to interactively navigate a single web page by altering the portion of it that is displayed at any given time. Methods used are panning, scrolling and zooming. Panning changes the area of the information space that is visible, while zooming changes the scale at which the information space is viewed. A lot of systems in this category apply gesture interactions to ease the limited input capabilities of the mobile devices.

Many interaction methods have been implemented as client-side navigation techniques. The focus + context visualization techniques attempt to give users both an overview and a detailed focus at the same time. This enables users to move their center of attention to different areas, while maintaining an overview. One such visualization technique is the generalized fisheye view [72], which provided a basis for much subsequent work. Focus + context techniques include fisheye views [73], bifocal displays [74], and table lenses [75]. Active Outlining [76] has also been implemented as a client-side navigation technique. In this approach, the user can dynamically expand and collapse sections of the document under their respective section headings. Other techniques which fall into the category of extended client-side navigation include semi-transparent widgets [77] and the Magic Lens system [78].

The Focus + Context techniques, as described above, reduce or eliminate scrolling. However, depending on different factors (i.e., amount of text, images, length of web page) they may make the distortion level too great. In this case, the user has to zoom-out or use extensive panning to find a familiar view or item in the page. The Fisheye systems have been studied in a number of contexts, and have been shown to perform well for navigating networks [79], for using menus [80], and for large tracing and steering tasks [81]. However, distortion-oriented techniques can also cause problems for some interactions such as targeting [82].

Zooming changes the visible proportion of the information space. When zooming in, more scrolling is required to achieve the same transformation of the viewport, and vice versa. Users must understand the relationship between scroll distance and zoom level, adding further overhead to scrolling operations. A comparison of zoomable interfaces with overview + detail interfaces (section 2.2.2 - Automatic Re-authoring) revealed that zoomable interfaces are faster for navigation tasks but overview + detail interfaces lead to higher satisfaction [83]. However, it is not known whether the speed difference observed might diminish when users learn to cope with the complexity of the overview + detail interface.

Client-side adaptation systems are resources consuming, especially in terns of CPU processing power and system memory of the mobile device, but are faster than any other adaptation technique. This is due to the advanced visualization techniques they use (zoom, pan, focus, fisheye view). Some systems of this category are able to run only on desktop emulators [33].

Fishnet [84] is a Fisheye web browser that shows a focus region at a readable scale while spatially compressing the page content which is outside the focus region, as its name suggests. In this Fisheye view, users need to scroll to move their desired information into the focus region. Fishnet is targeted to desktop browsing, and therefore, does not attempt to address horizontal scrolling.

Pad++ [85] applies another kind of zooming/panning technique to display content. The user can focus on and zoom into any part of the content, and smooth animation during the zoom insures that the semantic link between the zoomed information and its context is maintained.

A browser that is targeted for mobile devices is the RSVP browser [86]. This browser uses Rapid Serial Visual Presentation (RSVP) [87] to provide a rich set of navigational information for web browsing. When a user is asking, “where can I go from here”, the range of alternative, or links, may be presented sequentially in rapid succession in order to explore the “neighborhood” of the start-off page. Each of the links is represented by a preview image that is distinctive enough to be recognized when presented for only a very brief moment. The RSVP stream can be paused and restarted using a control that also shows the relative position of the currently displayed image. When a potentially interesting page is spotted, the neighborhood of this page may, in turn, be explored by the rapid presentation of its links. After this process has been repeated several times, a history of the browsing process is available. Every time the user accesses another page, a button is added to the browsing history. Users can move backward and forward in the browsing history. When the user taps the RSVP button, the view in the viewing area changes to show a deck view of the current page, meaning that the preview images for all the links are stacked up into a deck and each one is presented briefly followed by the next one. This way the user can riffle through all the links of a page quickly, thereby answering the question “where can I go”.

Collapse-to-zoom [88] allows users to zoom into relevant areas, and collapse areas deemed irrelevant, such as columns containing menus, archive material, or advertising. Collapsing content causes all remaining content to expand in size and reveal more detail, which increases the user’s chance of identifying relevant content. When finally switching to the full-size view, the page has been reduced significantly, which allows users to scroll through the remaining content in an effective way. Collapse-to-zoom uses a special gesture menu, called marquee menu. This menu offers four commands

for collapsing content areas at different granularities and to switch to a full-size reading view of what is left of the page. This technique allows users to make continuous progress towards the page content that is relevant to the user's task. Table 2.2 summarizes page layout adaptation mechanisms.

Table 2.2: Page Layout Adaptation Mechanisms.

Page Layout Adaptation Mechanisms	
Device Specific Authoring	
Transcoding & Automatic Re-authoring	- Single Column / Narrow Layout - Overview + Detail
Extended Client-side Capabilities	

2.3 Summary

This chapter presented the major adaptation mechanisms tailored to mobile devices. The proposed classification was based on the following parameters: type of adaptation, stage of implementation and enhanced/altered characteristics of the mobile device by the adaptation process. The most representative studies of each category were presented. These studies showed that, despite the improvement of the recent years in the area of mobile adaptation, there is still room for improvement for much usable and accessible mobile adaptation mechanisms.

Bickmore and Schilit in their presentation of Digestor concluded that automatic re-authoring is the ideal approach to provide broad access to the web from a wide range of devices, if it can be made to "produce legible, navigable and aesthetically pleasing re-authored documents without loss of information". The proposed approach in this thesis is claimed to successfully accomplish these four requirements.

Chapter 3

Unified User Interfaces

This chapter presents the foundations of the proposed adaptation mechanism. The *EAGER* toolkit, which provides the basic functionality for the development of the mobile adaptation mechanism, is presented first. The proposed system is developed as an extension of the *EAGER* toolkit. Their exact relationship, at a component-based level, will be presented at the end of this chapter. Both systems are based on the Unified User Interface methodology for the design and implementation of the adaptable interfaces. The Unified User Interface Design methodology and the Unified User Interface architecture are also presented in detail in this Chapter.

3.1 UniMob

The developed mobile adaptation system is named *UniMob* (**Uni**fied **Mob**ile User Interface) in order to reflect its Unified underlying methodology and approach. *UniMob* is an extension of the *EAGER* toolkit [89]. This toolkit proposes the Unified Web Interfaces (UWIs) method as a novel approach to the design and development of Web-based applications capable of automatic adaptation to user and context characteristics and requirements. This method builds on well-established Design for All principles and on the Unified User Interfaces methodology. *EAGER* provides developers with means of producing adaptable and adaptive web interfaces tailored to different user categories and contexts of use. Both able bodied users and users with disabilities can be facilitated by the adaptations built-in in the toolkit. The system uses six parameters to produce the adapted interface. These parameters are: assistive technology, device and display resolution, language, input device, disability and web familiarity.

The *EAGER* toolkit adopts the Unified User Interfaces Design methodology (U^2ID) [2] for the design of the provided alternative interface styles. Moreover, the implementation architecture used is based on the Unified User Interfaces architecture (U^2I) [2] in order to support the selected design methodology. The components of the

EAGER toolkit are implemented in C# programming language under the Microsoft .NET Framework.

UniMob follows the same principles in order to be compatible with the toolkit's underling functionality. These principles will be presented in detail in sections 3.2 and 3.3. The following paragraphs categorize *UniMob* according to the classification of mobile adaptation mechanisms presented in Chapter 2.

As previously mentioned, *UniMob* follows the U^2ID methodology. According to U^2ID, only a single unified user interface is designed and developed, which comprises alternate interaction components, appropriate for different target device characteristics and user characteristics-preferences. The unified interface may undergo multiple instantiations at the initiation of interaction (adaptability). Moreover, each interface instance can be continuously enhanced at run-time (adaptivity). The adopted architecture is the U^2I architecture. This architecture provides the ability to web developers to easily add a new interaction component or a different instance of an existing interaction component.

A server side solution is offered in order to minimize the processing load of the client device. All adaptation decisions and interface instantiations are performed at server side. This process produces a minimum sized page to be delivered to the client device, resulting in minimum network load. Server side adaptation depends highly on knowledge about the device characteristics, therefore an accurate and reliable mechanism for collecting such data is essential. In order to collect the device characteristics, client side scripting and the headers of the request messages are used. Of course, the user is able to change these preferences by accessing a personal profile. The system also provides a set of predefined profiles that support different contexts of use. All device characteristics and user preferences are stored in a database and are retrieved prior to the initiation of the interaction.

The proposed system falls in the category of page layout adaptation, and more specifically in the automatic re-authoring subcategory. Both the content and the navigation of the page are adapted in order to eliminate horizontal scrolling and reduce vertical scrolling to a minimum. The interaction of the alternative UI components is designed in such a way as to address the limited input capabilities of PDA devices (stylus, virtual keyboard). Another basic concept of this approach is that web designers have total control on the final form of the adapted interface. The

interaction quality of the outcome of the adaptation process is a matter of careful user-centered design, and not the result of an algorithm in a reverse engineering process.

3.2 Unified User Interfaces Design Methodology

The Unified User Interfaces Design (U^2ID) methodology proposed in [2] provides the capability of producing automatically adapted interfaces intended to be used by users with diverse requirements and operated in different usage contexts.

Contrary to other design methodologies, the U^2ID methodology does not lead to a single interface design. The outcome is a collection of different designs. Each one of these designs encapsulates the designer's knowledge about the user of this specific design and the context of use. This knowledge (who, when, why, how) is very important not only during the design of the alternative Unified interfaces, but also during the definition of the conditions under which each of these interfaces should be used. For example, the task of displaying images in a web page can lead to three alternative designs. In the first one, the image is displayed in the regular way. This design is activated in desktop browsers and is common to expert, regular and naïve users. The second design displays images as thumbnails. This design is activated in mobile browsers and is used by naïve and regular users. The third design displays images as links to the original image. This design is activated in mobile browsers and is used by expert users and in case the device does not support images (the link provides the possibility to save the image).

The U^2ID methodology has been defined so as to address two objectives: a) enable the "fusion" of all potentially distinct design alternatives into a single unified form, without, however, requiring multiple design phases, b) produce a design structure which can be easily translated by user interface developers into an implementation form.

The Polymorphic Task Hierarchy is the basic technique that is used in the U^2ID methodology. It combines a) hierarchical decomposition, b) design polymorphism, and c) operators based on user tasks. The hierarchical decomposition adopts the properties of hierarchical task analysis in order to decompose each task into its lower level actions. Design polymorphism provides the capability of alternative designs in each level of the task hierarchy according to the user and context parameters. The task operators are based on the CSP (Communicating Sequential Processes) language for

describing the behaviour of reactive systems [90]. However, interface designers are not constrained to using a particular model for describing user actions. The concept of polymorphic task hierarchies is illustrated in Figure 3.1. Each alternative task decomposition is called a decomposition style and is given an arbitrary name. The alternative task sub-hierarchies are attached to their respective styles.

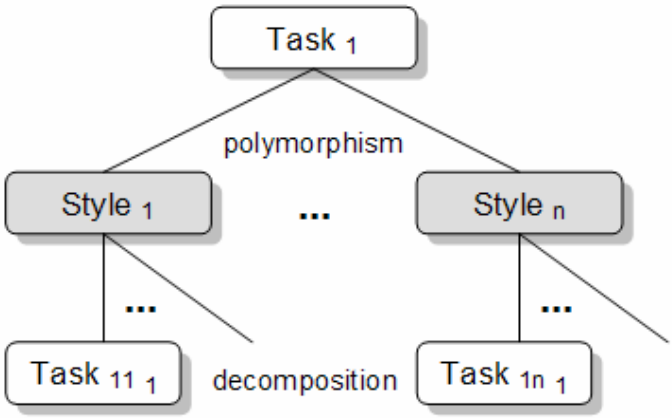


Figure 3.1: Polymorphic Task Hierarchy.

A polymorphic design artifact is a collection of alternative solutions for a single design problem, each alternative addressing different problem parameters. There are three categories of design artifacts subjected to design polymorphism: a) User Tasks (what the user has to do), b) System Tasks (what the system has to do), c) Physical Design (the interface components on which user actions are to be performed).

The U^2ID methodology requires an exhaustive hierarchical task analysis ranging from the abstract task level to specific physical actions, and decomposition is applied either in a unimorphic fashion, or by means of alternative styles. Figure 3.2 illustrates the polymorphic task decomposition process.

The process starts from an abstract of physical task design depending on whether the top-level user tasks can be abstract or not. An abstract task can be decomposed either in a polymorphic fashion or in a traditional manner following a unimorphic decomposition scheme. In the latter case, the transition is realized via a decomposition action, leading to the task hierarchy decomposition state. In the case of polymorphic decomposition, the transition is realized via a “polymorphose” action, leading to the design alternative sub-hierarchies state. Reaching this state means that the alternative styles have been identified. The next step is to further decompose the different sub-tasks. For each sub-task at the abstract level, there is a transition to the abstract task

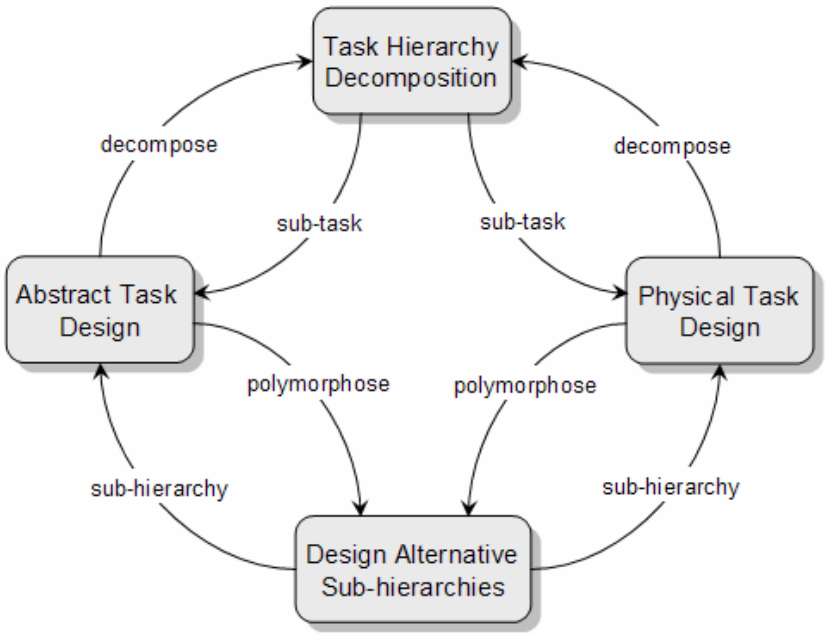


Figure 3.2: Polymorphic Task decomposition process.

design state. If the sub-task explicitly engages physical interaction means, a sub-task transition is taken to the physical task design state. Physical tasks may be further decomposed in a unimorphic fashion, or in a polymorphic fashion.

A running interface, implementing such alternative designs, should encompass decision making capability, so that, before initiating interaction with a particular end-user, the most appropriate of those artifacts are activated for all polymorphic tasks. In order to enhance such capabilities, a set of decisions should be taken and documented in a design rationale. For each alternative style in a polymorphic task hierarchy, the design rationale should contain: the related task, the targets of this specific design, the context parameters and the run-time relations among alternative styles. In the U^2ID method, four design relationships between alternative styles are distinguished, defining whether alternative styles may be concurrently present at run-time. These relationships are exclusion, compatibility, substitution and augmentation.

Exclusion is related to styles that are mutually exclusive, thus are not activated concurrently. Compatibility concerns styles that can be concurrently available to the user without introducing usability problems. Substitution is connected with adaptivity techniques. At runtime, a style is decided to be substituted by another style. Finally, augmentation aims to enhance the interaction with a particular style that is valid, but not sufficient to facilitate the user's task.

The outcomes of the U^2ID methodology include:

- the design space which is populated by collecting and enumerating design alternatives
- the polymorphic task hierarchy which comprises alternative concrete artifacts
- the recorded design rationale, for each design artifact produced, which has led to the introduction of this particular design artifact.

3.3 Unified User Interface Architecture

The Unified User Interface (U^2I) architecture is compliant with the definition of architecture provided by the Object Management Group [91]. According to this definition, the architecture should supply components, functional roles per component, communication protocols or APIs, as well as implementation and inter-operability issues. The U^2I architecture comprises of four independent intercommunicating components with well-defined roles and behaviour. These components are: Context Information Server (CIS), User Information Server (UIS), Decision Making Component (DMC) and Dialogue Patterns Component (DPC). Each component will be presented in brief.

Context Information Server

The CIS provides context attribute values for the platform and the environment. These attributes can be divided into static and dynamic. Static attributes remain unchanged during the interaction (e.g., peripheral devices). Dynamic attributes can vary during the interaction (e.g., noise level).

CIS contains information about the installed IO devices and their characteristics (e.g., handheld binary switches, speech synthesizer (UK, GR), graphics card (mode 16/24 bits, 1024x768, 75Hz, 64MB)). It also contains information about parameters of the operating environment that can change (e.g., noise level, light level, user presence).

The information exchange between CIS and DMC is initiated in two ways: by request, when all the stored characteristics or part of them is sent, and, secondly, when a dynamic attribute is changed (by modification) during the interaction. In this case, only the name and the value of the changed attribute are sent. This type of communication supports the adaptivity of the system. The context characteristics are transmitted as attribute-value pairs (e.g., “environment noise, 78db”).

User Information Server

The UIS provides user profile characteristics in two cases. During the initiation of the interaction, and dynamically upon the discovery of a new user characteristic by monitoring runtime interaction events (adaptivity).

UIS can be a user profile repository stored in a database. However, it should be capable of receiving and processing interaction monitoring information. An appropriate architectural design pattern has been derived and can be found in detail in [2].

The information exchange between UIS and DMC takes place in three cases. By request, when all the user characteristics or part of them is sent to the DMC. By modification, when a user characteristic changes during the interaction, only the attribute name and its value are sent. By detection, when a new user characteristic is detected during the interaction, the name and the value of the new attribute are sent. The last two cases support the adaptivity of the system.

Decision Making Component

DMC has a primary role among the other components. It is the "brain" of the adaptation process. The final decisions about the alternative designs that will be activated in order to construct the final adapted interface are taken by considering the user profile and the context profile attributes. This component contains the interface decision logic provided by the designers of the interface.

DMC contains the logic of matching user and context attributes with alternative interface designs. For example, the decision logic for the presentation of links in a web page can be recorder in the following way: if "web knowledge" in {normal, low} then "push buttons", if "web knowledge" in {very good, good} then "underlined text". A special language that describes rules of this type is the Decision Making Specification Language (DMSL) [92].

CIS and UIS send the context and user profiles to the DMC. The DPC receives from DMC commands regarding the interface patterns. There are two types of commands, activation and cancellation. With the activation command, the DPC is informed that a particular interface pattern was selected to be included the final interface. With the cancellation command, the DPC is informed that a particular interface pattern, which is already part of the interface, must be deactivated.

Dialogue Patterns Component

The DPC executes the adaptation decisions of the DMC. It assembles the interface from the selected interface patterns. DPC is capable of performing such assembly because it knows the location of each implemented alternative interface pattern. Each implemented dialogue pattern is referred to as style.

DPC is the source of all the implemented interface patterns. The implementation can have several forms, e.g., dynamic libraries, components in component based architectures, static implementations, etc.

DMC sends activation and cancellation commands to the DPC. Also the UIS sends to the DPC control parameters of the monitoring process. From its side, the DPC sends to the UIS data collected during the monitoring process. It also requests from the DMC adaptation decisions regarding alternative interface patterns.

Figure 3.3 and Table 3.1 show the structural components of the U^2I architecture, their relations and the communication messages exchanged between them.

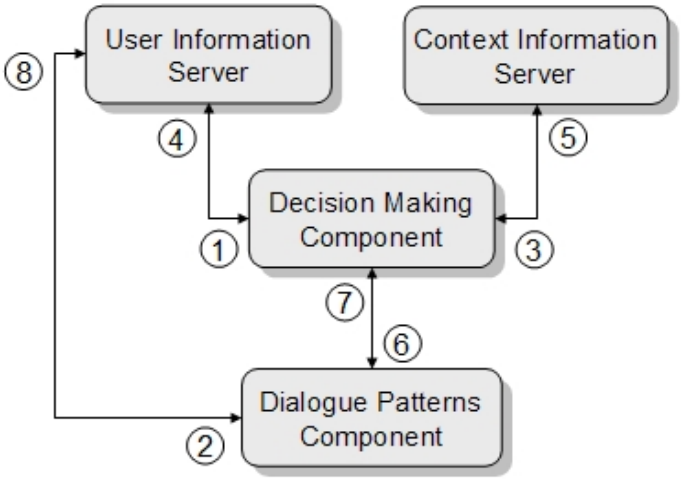


Figure 3.3: Structural components of the Unified User Interface architecture.

Table 3.1: Communication messages exchanged between the structural components of the Unified User Interface architecture.

1.	UIS → DMC	(User profile before initiation) (Dynamic user attribute values)
2.	UIS → DPC	(Interaction monitoring control)
3.	CIS → DMC	(Context profile before initiation) (Dynamic context attribute values)
4.	DMC → UIS	(Request user profile)
5.	DMC → CIS	(Request context profile)
6.	DMC → DPC	(Activation/Cancellation of dialogue patterns)
7.	DPC → DMC	(Requesting which dialogue patterns to activate)
8.	DPC → UIS	(Monitoring data)

3.4 UniMob - EAGER Relation

UniMob is an extension of the *EAGER* toolkit. This means that in order to add mobile adaptation functionality to a portal, this portal must have been implemented by means of the *EAGER*. When the context of use is detected to be a mobile device, then the styles of the *UniMob* are used in the construction of the interface.

EAGER implements the U^2I architecture that consists of four components (Dialogue Patterns Component, User Information Server, Context Information Server and Decision Making Component). The following sections describe in detail how *UniMob* extends the components of the *EAGER* toolkit, and how they are used in the mobile adaptation process.

3.4.1 Dialogue Patterns Component

EAGER is a toolkit tailored to desktop adaptation, and the implemented alternative styles it contains are intended to be used for interaction on desktop computers. *UniMob* extends all the base interfaces for each one of the user tasks that are supported by the *EAGER* with alternative styles tailored to mobile devices. The design and implementation details of these styles will be presented in Chapter 4. The extended Dialogue Patterns Component (DPC) now contains all the alternative desktop styles-patterns and all the alternative mobile styles-patterns. All of them are implemented in C# under the Microsoft .NET Framework.

3.4.2 User Information Server

In *EAGER,* the information regarding the user characteristics is stored in a database system (Microsoft SQL Server 2000). The user characteristics are separated into two groups, the basic characteristics and the extended characteristics. The basic characteristics are required for performing decision making by the Decision Making Component (DMC). The extended characteristics are used to override decisions made by the DMC in case the user selects manually a specific style for a certain task (e.g., Paging task - Link mode style / Drop Down style).

Not all the characteristics provided by the toolkit are used in the mobile adaptation process of *UniMob*. Tables 3.2 and 3.3 contain the total characteristics of the user profile that are available in the toolkit. The shaded characteristics are those used also

by *UniMob*. Characteristics marked with (*) were added to the user profile. The remaining characteristics are either used, but are not directly related to the mobile adaptation (e.g., the Language characteristic is available to the user but does not have any special PDA alternative) or they have only one alternative style for the mobile adaptation (e.g., fieldsets, options). Moreover, three new extended characteristics were added to the collection of the toolkit. These are: "Paged Text", "Text List" and "Time Selection". As long as the *EAGER* toolkit is tailored primarily for desktop adaptation, it was essential to develop for each one of the new characteristics an alternative desktop style additional to the PDA styles.

Table 3.2: Basic characteristics of the user profile. Shaded cells contain characteristics used by the *UniMob*.

Assistive Technology	Language
Device	Web Familiarity
Disability	Input Device

Table 3.3: Extended characteristics of the user profile. Shaded cells contain characteristics used by the *UniMob*. Characteristics marked with (*) were added to the user profile.

File Uploading	Module Options
File Displaying	Module Functions
Image Uploading	Template Linearization
Image Displaying	Table Linearization
Date Selection	Charts
* Time Selection	Images
Paging	Color Settings
Tabs	Font Sizes
Fieldsets	Font Families
Quick Access Links	Search variations
Section Breaks	Commonly used options (favorites)
Dynamic Adaptation	Navigation
Text Entry	Skins
Display links as buttons	* Paged Text
Text to Speech	* Text Lists
Text editing	

In order to support dynamic adaptations (adaptivity), the User Information Server (UIS) is responsible for monitoring specific user actions during interaction. The controls that support adaptivity are a) the "Favorites" control (Figure 3.4) and b) the order of options in navigation menus. If the user enables dynamic adaptation, then the UIS provides the necessary monitoring data to a) create the list of most commonly

used options (favorites) and b) rearrange the order of options in navigation menus in order to display first the most frequently used.

Figure 3.4: Favorites control (adaptivity).

3.4.3 Context Information Server

This component provides information about the context of use. There are two basic parameters that must be detected in case of mobile adaptation, namely device type and display resolution. The CIS of the toolkit was extended in order to support the detection of both of them.

In the case of *UniMob,* it is of major importance to be able to detect the device type. A straightforward solution is to use javascript. Although most mobile browsers currently support javascript, there are still many devices with earlier versions of browsers that do not support it. A more reliable solution that was adopted in this thesis is to extract this information from the headers of the http request message. In order to detect the display resolution, a script (javascript) is executed in the mobile device and returns the results to the server. This information is held by the CIS and is passed to the DMC upon the initiation of the adaptation process. When javascript is not supported by the browser, a default resolution is used (320x240). If the user is not satisfied with the adaptation decisions of the default resolution, an alternative one can be selected from the “General Settings” area that is available in the adapted portal.

3.4.4 Decision Making Component

The decision logic of the toolkit was extended with the appropriate rules according to the design rationale of the alternative PDA styles. The rationale of each style states the conditions under which this style is activated. Initially, the decisions of the DMC are based exclusively on the basic characteristics of the user profile. The user can

modify these decisions by selecting directly an alternative style through the "Custom Accessibility" and "Interaction Preferences" areas that are available in the adapted portal.

3.5 Summary

This chapter presented the foundations of the *UniMob* adaptation system. The relation of the UniMob with the *EAGER* toolkit was discussed. The Unified User Interfaces Design methodology and Architecture, that constitute the basis of both EAGER and UniMob, was described in detail. The next chapter presents the design and implementation details of the alternative styles that are used in the mobile adaptation. These styles constitute the major extension of the *EAGER* toolkit by the *UniMob* system.

Chapter 4

Unified Design of UniMob's Interactive Artifacts

The final interface of a portal is built of several interface components. The alternative interface styles of these components are stored in the Dialogue Patterns Component. This chapter presents the steps that were followed during the application of the U^2ID methodology in the design of the alternative styles of the mobile components included in UniMob. In [2], the outcomes of the Unified user Interface Design method have been defined as:

- a polymorphic task hierarchy
- the design rationale for the designed style
- the design space populated by design alternative.

Accordingly, the next section of this Chapter presents the polymorphic task hierarchies of each user task addressed in UniMob. Next, the design rationale for each alternative style is discussed. The chapter ends with the presentation of the implemented styles, along with some interesting details about their implementation and functionality. These styles constitute the major extension of the *EAGER* toolkit provided by the *UniMob* system.

4.1 Designing the Mobile Styles

The primary goal of the designer of a Unified User Interface is to detect all the possible parameters that raise the need for alternative styles. The next step is to document the relationship of these parameters with each one of the alternative styles. This is done in the design rationale.

Several parameters related to the user and the context of use can be used in order to populate the desired design space. The adaptation components share a de facto context parameter. All the components presented here are tailored to **mobile devices**. Based on this major parameter, all the rest parameters that will be described next are

deployed. The constraint of mobile device can be further decomposed into a series of other context parameters such as screen resolution, color support, input capabilities, sound capabilities, and available bandwidth. The term “mobile device” is used as a reference to all the above context parameters assuming a common state for these. This common state can be described as: limited screen resolution, full/limited/no color support, virtual keyboard and stylus for the interaction, sound support but not multi-channel and limited bandwidth. In order to facilitate a wide range of possible users, two user related parameters are adopted. The first one is the **user expertise**. Novice, Moderate and Expert are the categories in which the users are classified. Each one of those categories differs significantly in the quantity-quality of the on-screen information and the available functions. This distinction can lead to totally different interface styles. The other parameter is **user disability**. Alternative styles are constructed for the following user states: blind, color blind, motor impaired and low vision.

The next section presents the task hierarchies of the adapted components. The last section contains the design rationale for each task hierarchy.

4.1.1 Polymorphic Task Hierarchies

The diagrams and their explanations that are used in polymorphic task hierarchies are shown in Figure 4.1.

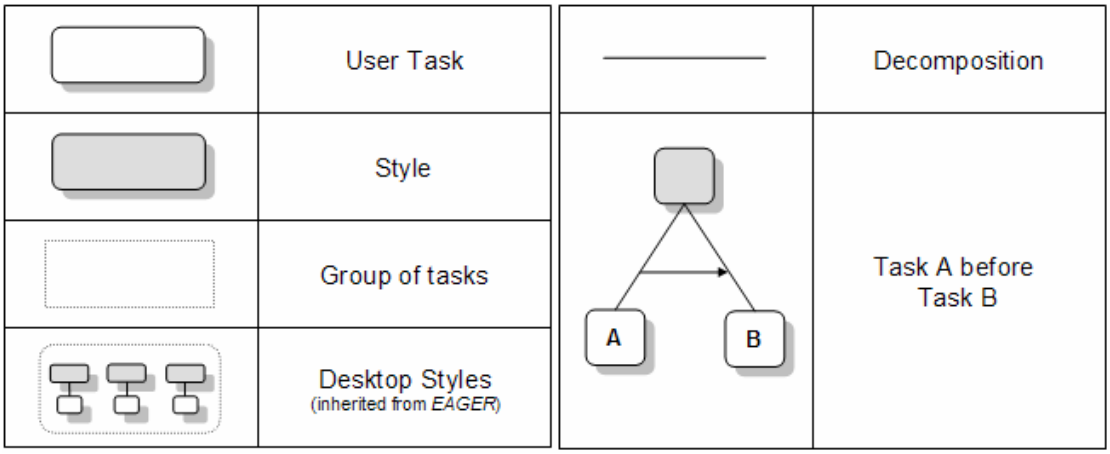


Figure 4.1: Diagrams used in polymorphic task hierarchies.

4.1.2 Content Access Styles

The styles of this category are related to the content of a web page. The way the user interacts with the elements of a web page is associated with the alternative style used for each element. In the following paragraphs these alternative styles will be presented.

View Image

Figure 4.2 depicts the task hierarchy of the "View Image" task. This task has six alternative styles. According to the selected style, the system creates the preview or the adapted versions of the original image. In case of adapted image, three alternative styles are available (S1_1, S1_2, S1_3). Each one creates a different version (full color, grayscale, pure black-white) of the original image. This transcoding is very important in mobile environments. The user tasks are simple and lead to the alternative versions of the original image. There are two thumbnail styles, Thumbnail Image (S1_5) and Thumbnail Link (S1_6). These styles show a very compact version of the image, either displayed as thumbnail or a link. Their distinct role will be discussed later on the design rationale section.

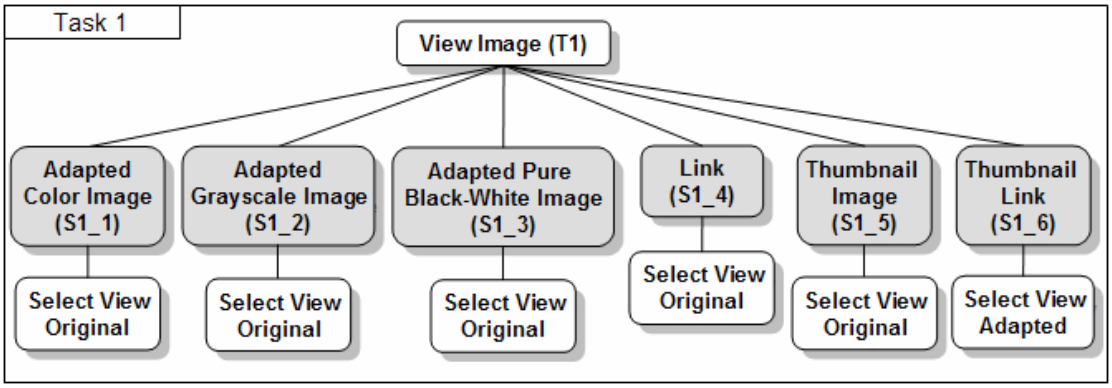


Figure 4.2: Polymorphic Task Hierarchy of "View Image" task.

View Table

Figure 4.3 shows the task hierarchy of the "View Table" task. This task has two styles. The "Linearized" style requires the system to transform the table into a serialized form. The second design is much more interactive and flexible, as it requires the user to select the visible columns of the table. The user tasks must be executed in a fixed order consistent to common sense.

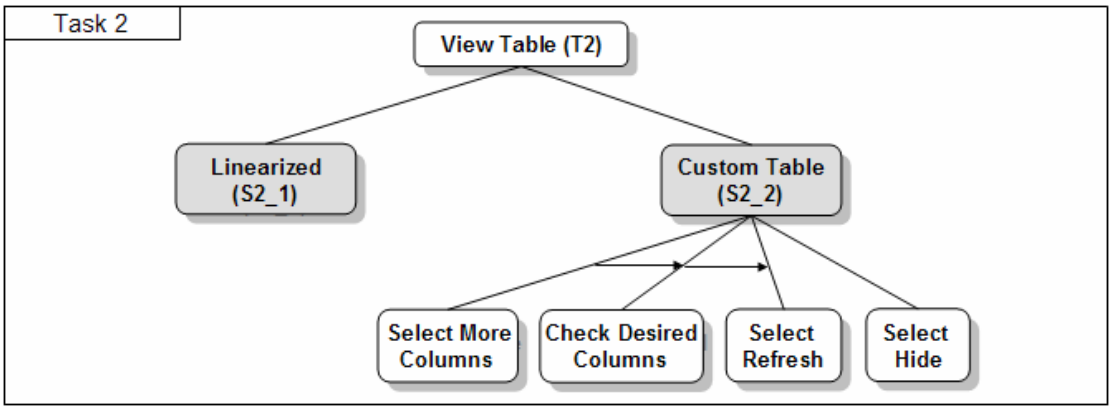


Figure 4.3: Polymorphic Task Hierarchy of "View Table" task.

View Paged Content

Figure 4.4 shows the task hierarchy of the "View Paged Content" task. This task provides the underlying mechanism of a paging functionality. There are three alternative styles, "Links", "Dropdown" and "No Paging". The first style is straightforward, and requires a smaller number of user tasks with respect to the second style. Users are usually familiar with such types of design mainly from their experience with search engines. In any case, the metaphor of book pages is common among all kinds of users.

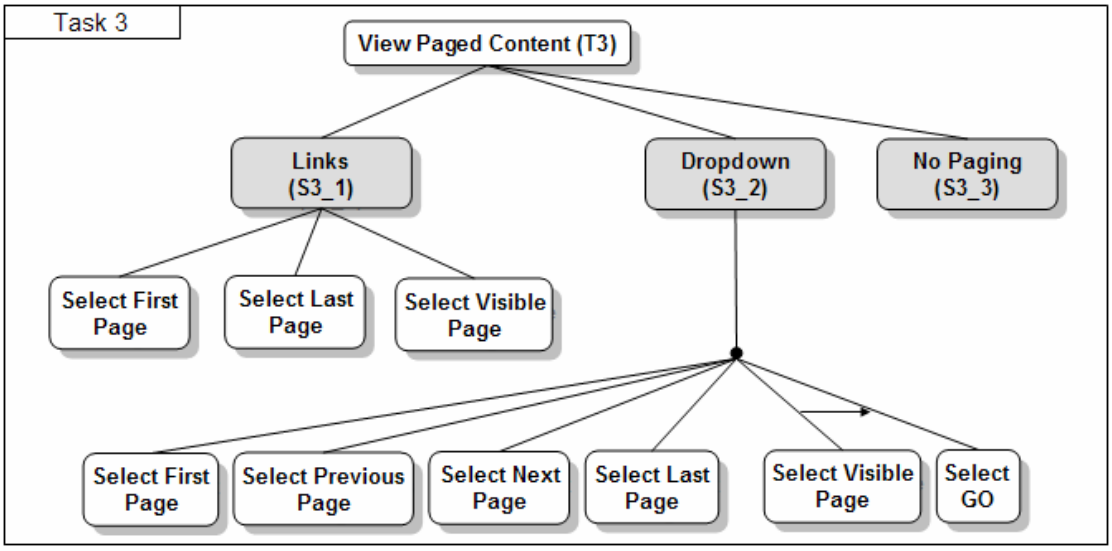


Figure 4.4: Polymorphic Task Hierarchy of "View Paged Content" task.

Select Date

Figure 4.5 depicts the task hierarchy of the "Select Date" task. Although this is a quite simple task, mobile devices require special designs in order to preserve this simplicity.

Three styles are available. In the "Direct" style, the user tasks are reduced to a minimum. Consistency of the number of days of each month-year combination is ensured. On the other hand, the "Indirect" style adopts a step by step logic, thus introducing more user tasks in the process. The core of the selection process (depicted as a group of tasks) is activated / deactivated through user tasks that are executed in a predefined order. The last style has the same user and system tasks as the "Direct" style, but the difference is in the way they are presented to the user. The last style uses a graphical calendar and the "Direct" style uses a set of dropdown menus. This distinction is important for users with different levels of web expertise.

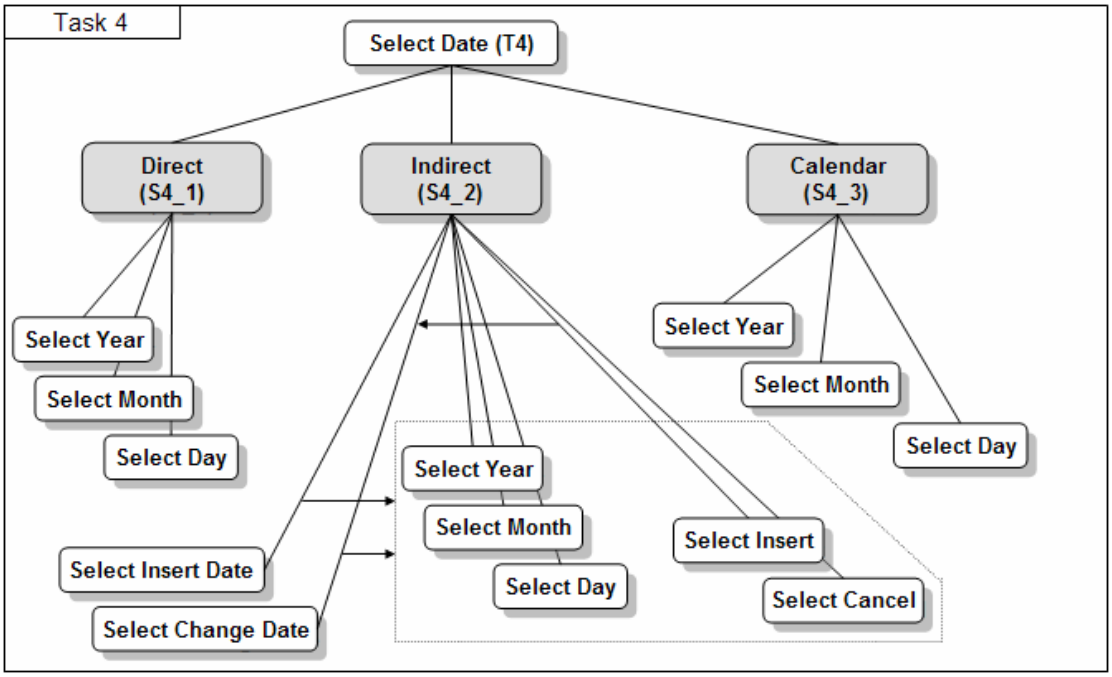


Figure 4.5: Polymorphic Task Hierarchy of "Select Date" task.

Select Time

Figure 4.6 shows the task hierarchy of the "Select Time" task. Two are the available styles, "Direct" and "Indirect". The first style has only two user tasks, insert hours and insert minutes. The "Indirect" task is divided into steps and, as stated earlier in the "Select Date" task, this introduces additional user tasks. The core of the select time process is depicted as a group of tasks.

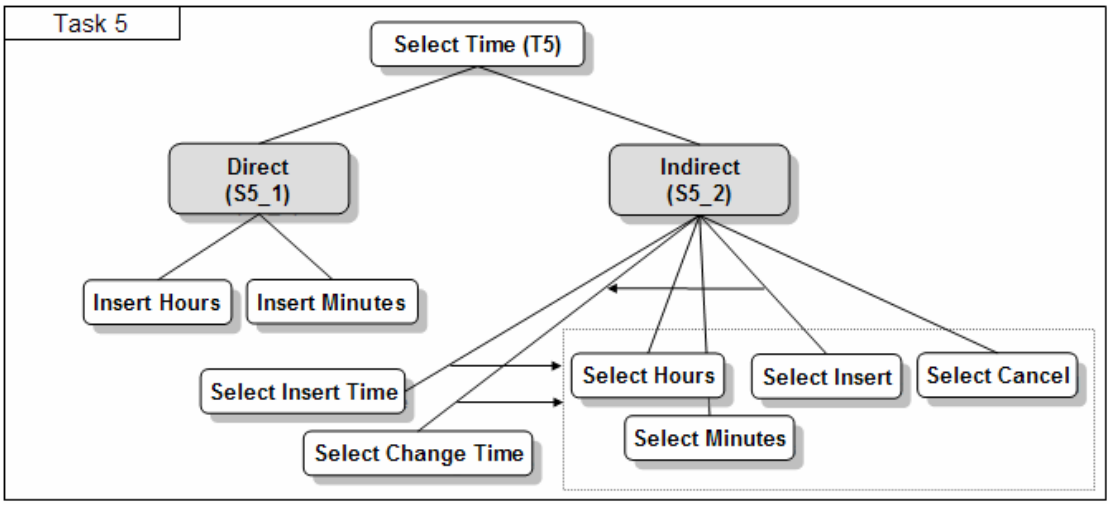


Figure 4.6: Polymorphic Task Hierarchy of "Select Time" task.

Upload File

Figures 4.7 and 4.8 show the task hierarchy of the "Upload File" task. This is actually a task for two functions "Add-Upload" and "Delete" a file into the uploaded set of files. This is clear through user tasks such as "Select Upload" and "Select Delete". All the alternative designs support this double functionality. The user needs to provide the "file" and the title of the file prior to the "Select Upload" task. Equally, the "Select Delete" task requires first the selection of some files from the uploaded files list. In different case the system will not be able to complete these tasks. The "Mixed" and "Indirect" styles adopt a step by step approach. The "Mixed" style

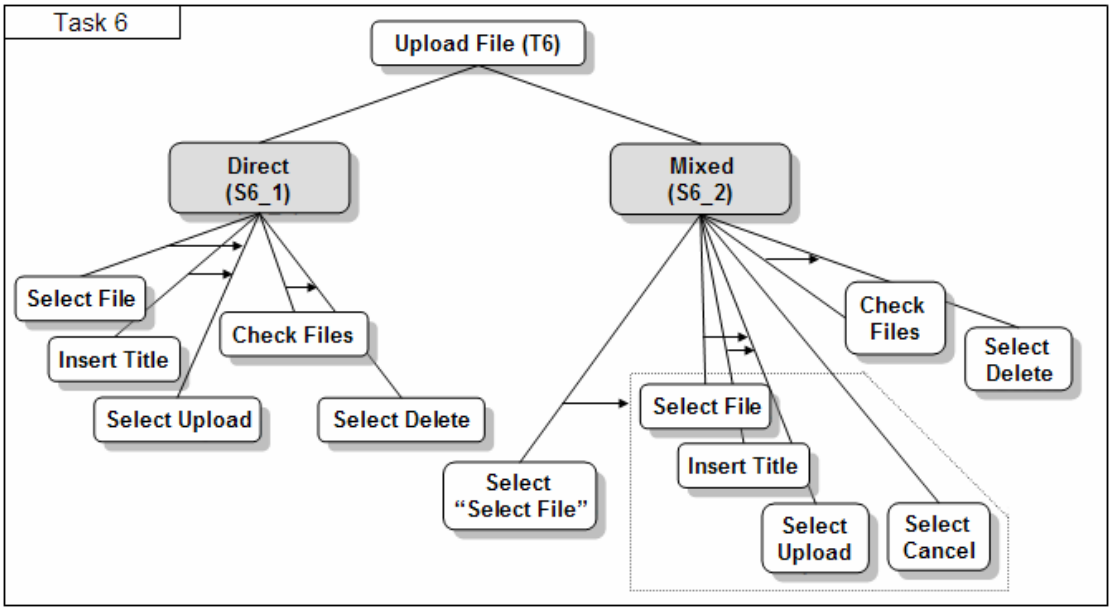


Figure 4.7: Polymorphic Task Hierarchy of "Upload File" task, "Direct" and 'Mixed" styles.

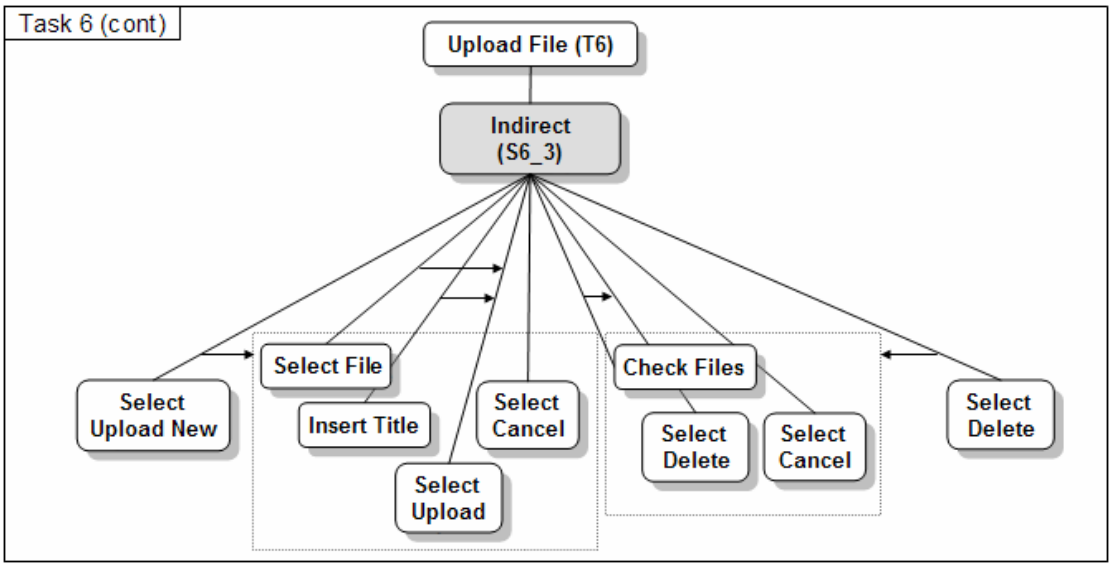


Figure 4.8: Polymorphic Task Hierarchy of "Upload File" task, "Indirect" style.

groups only the tasks required for the upload process. On the other hand, the "Indirect" style forms two task groups, one for the upload process and the other for the delete process. Access to these two groups adds more interaction steps to the "Indirect" style.

Upload Image

Figures 4.9 and 4.10 show the task hierarchy of the "Upload Image" task. The task decomposition adopts the same logic as the decomposition presented in the

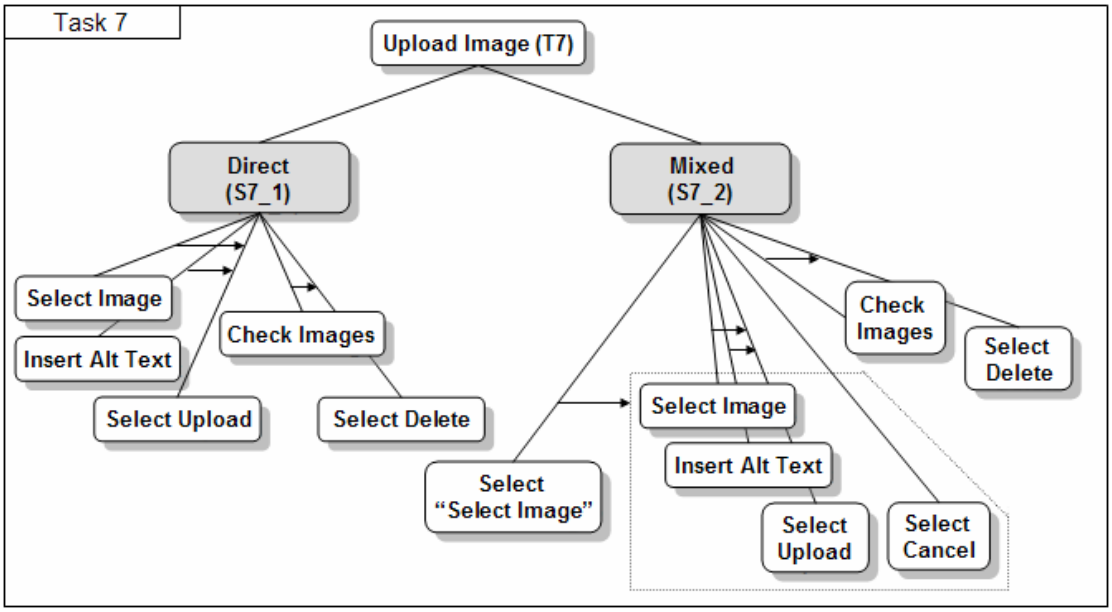


Figure 4.9: Polymorphic Task Hierarchy of "Upload Image" task, "Direct" and 'Mixed" styles.

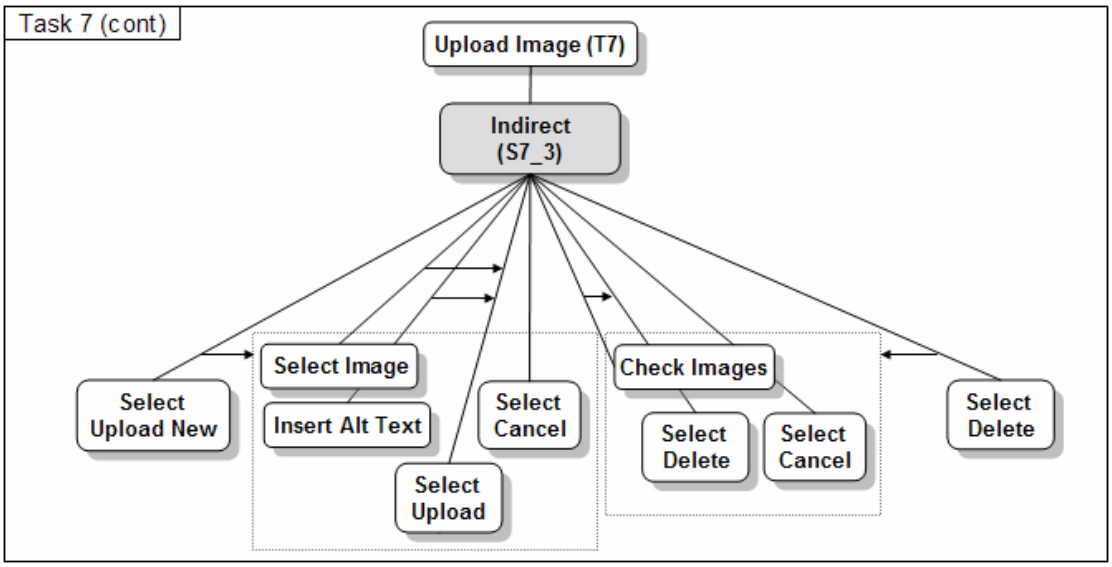


Figure 4.10: Polymorphic Task Hierarchy of "Upload Image" task, "Indirect" style.

previous task, "Upload File". These two tasks seem identical as long as an image is a file. However, their difference is at the level of implementation and will be discussed in the next section of this chapter.

View Function

Figure 4.11 shows the task hierarchy of the "View Functions" task. The alternative designs of this task are related with the help provisioning approaches. There are two styles that provide help and one that does not provide help. The "Multiple Help" and "Single Help" styles follow the same functional logic. User requests help and the system displays it, either as a whole or as help for a specific function. The difference is in the presentation and not so much in the involved tasks. The "No Help" task provides minimum functionality and enables the user to simply select a function to execute.

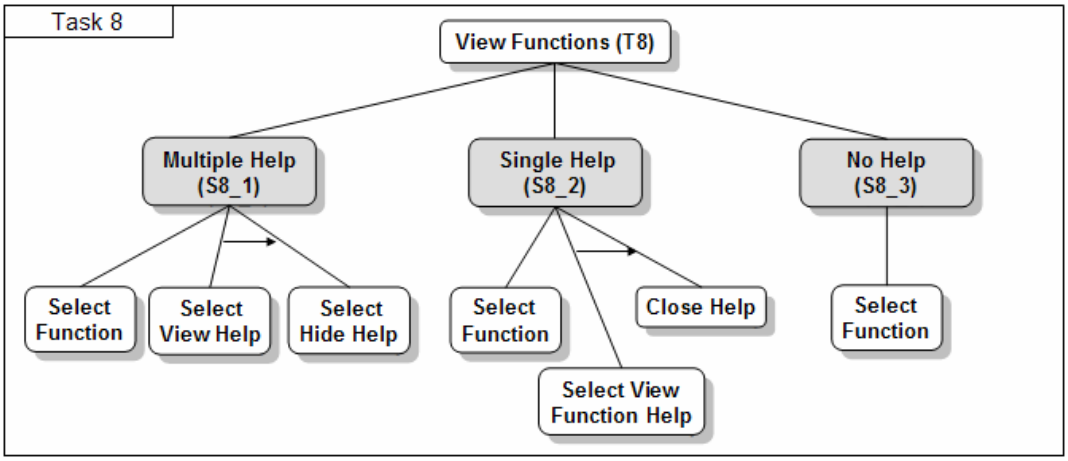


Figure 4.11: Polymorphic Task Hierarchy of "View Functions" task.

Display Files

Figure 4.12 shows the task hierarchy of the "Display Files" task. This task is decomposed into two alternative styles. These two styles seem to provide the same functionality. Both of them decompose to a user task (Select File). Although they may seem identical at this level, their discrimination is related to the actual design artifacts used to construct each one of them (lexical level). The "Compact" style, as its name denotes, provides a compact representation and the "Detailed" style a more extended version of file information. In this way, these styles are suitable for different user categories.

Display Image

Figure 4.13 shows the task hierarchy of the "Display Images" task. This task has the same design decomposition and logic with the "Display Files" task already described above. However, in this case, the "Detailed" style uses a list of images, and the "Thumbnails" style uses thumbnails. The user tasks remain as simple as in the "Display Files" task.

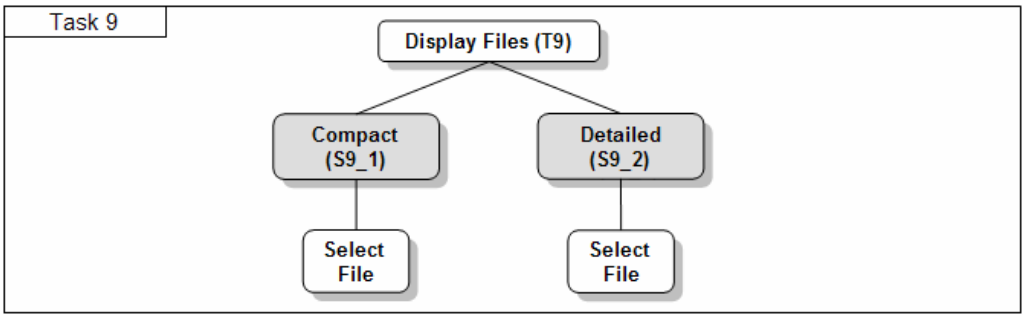


Figure 4.12: Polymorphic Task Hierarchy of "Display Files" task.

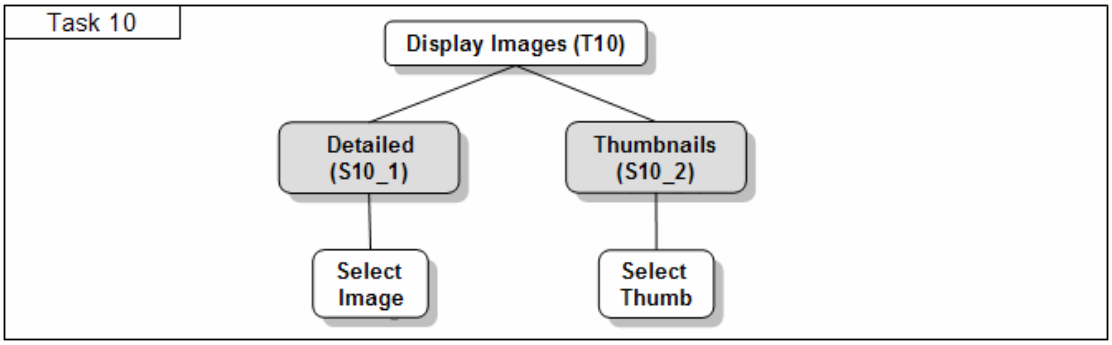


Figure 4.13: Polymorphic Task Hierarchy of "Display Images" task.

View Options

Figure 4.14 shows the task hierarchy of the "View Options" task. This task hierarchy is quite simple and provides an alternative view properly displayed in mobile devices. The actual decomposition is between this style and the alternative styles designed for desktop environments. The desktop styles are provided by the *EAGER* toolkit. Therefore, in UniMob, this task has only one style and straightforward functionality.

View Fieldset

This task is similar to the previous one. In Figure 4.15 the task hierarchy for the "View Fieldset" task is displayed. The task it contains is important in order to support different mobile browsers.

View Tabs

Figure 4.16 shows the "View Tabs" task, similar to the "View Fieldset" task. The graceful display of tabs in different mobile browsers is the aim of this task. The displayed options add an expected user task (Select Tab Option).

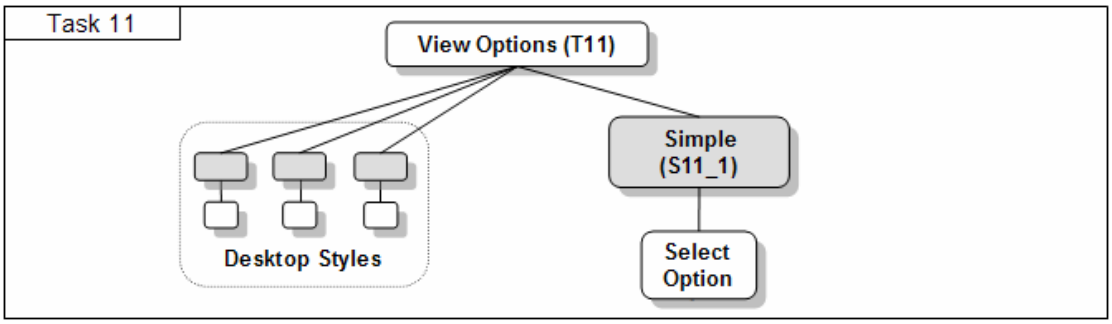


Figure 4.14: Polymorphic Task Hierarchy of "View Options" task.

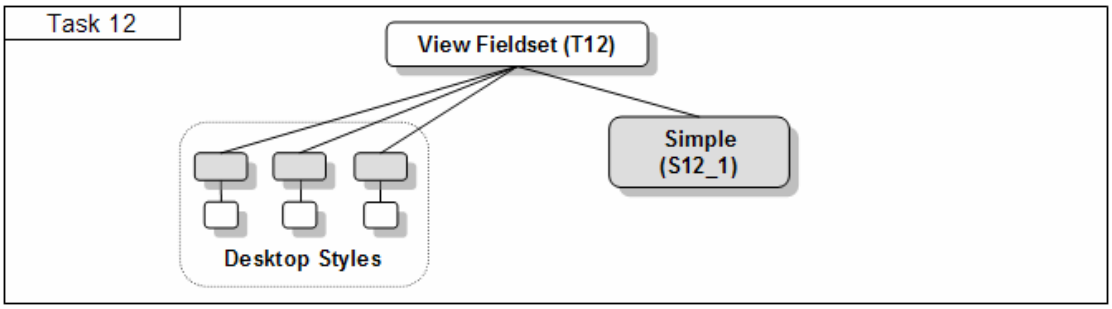


Figure 4.15: Polymorphic Task Hierarchy of "View Fieldset" task.

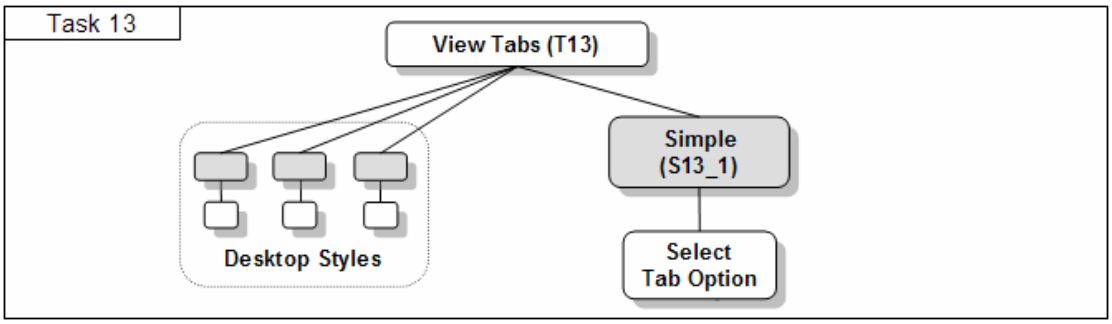


Figure 4.16: Polymorphic Task Hierarchy of "View Tabs" task.

Display List

The task showed in Figure 4.17, is the "Display List" task. This task has two styles tailored to mobile devices and two styles tailored to desktop environments. The "Normal Markup" styles are used in the desktop environment to display a list of textual elements (S14_1) or a list of links (S14_2). Equally, the "Compact Markup" is used in the mobile environment to display a list of textual elements (S14_3) or a list of links (S14_4). The additional user task in case of list of links is the selection of a link from the user. It is clear that the distinction of the styles is in the way they implement the displaying of the lists, the produced markup differs significantly.

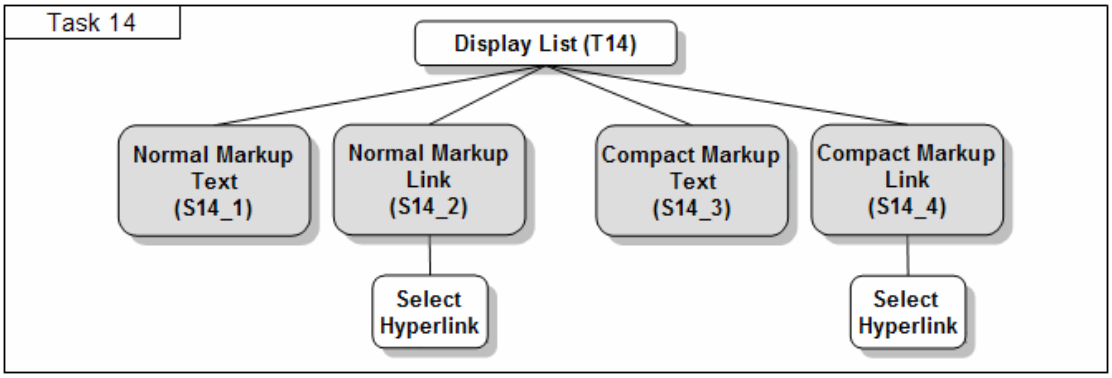


Figure 4.17: Polymorphic Task Hierarchy of "Display List" task.

Display Paged Text

The "Display Paged Text" task is shown in Figure 4.18. This task uses the "View Paged Content" task which was introduced earlier. When the first style "Paged HTML" is active, the given HTML code is separated into pages. The user task (View Paged Content) serves as a browser to this paged HTML code, and the system updates the content of the visible page. The second style, "Paged Text" has the same underlying functionality. The only difference is that the content that is being split in order to fit into pages is pure text and not HTML code. The last style does not import any paging functionality and displays the whole HTML code or text at once. This style does not have any user tasks.

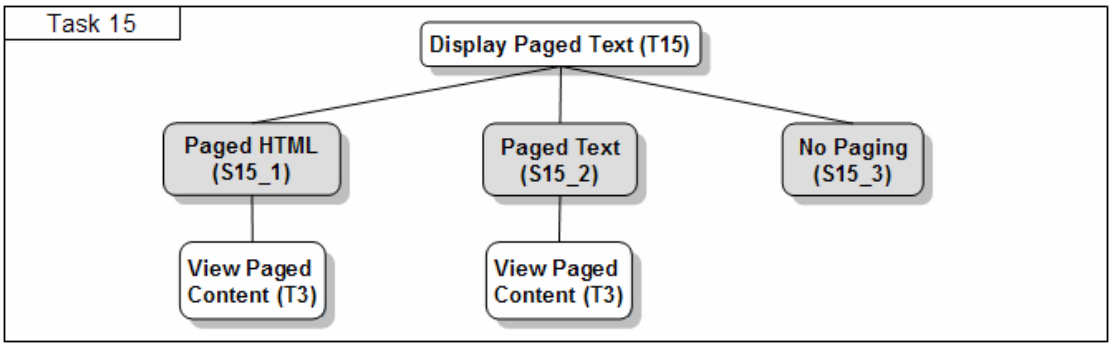


Figure 4.18: Polymorphic Task Hierarchy of "Display Paged Text" task.

Search

A common web task is the "Search" task, depicted in Figures 4.19 and 4.20. This task has three alternative styles, "Simple", "Advanced" and "Moderate". In the "Simple" style, the user inserts the desired text keywords and specifies the type of search ("All Words", "Any Word", "Exact Phrase"). The final task is to notify the system in order to start the search. This is done through the "Select Search" user task. The "Advanced" style is a more complex version of the Simple style. The difference is that it imports additional user tasks, depicted here as "Select Advanced Options", in order to perform a more specific search with the given text keywords. The "Moderate" style enables user to select whether to display or hide the advanced options. In case the user selects to display the advanced options, these options can be used in the performed search.

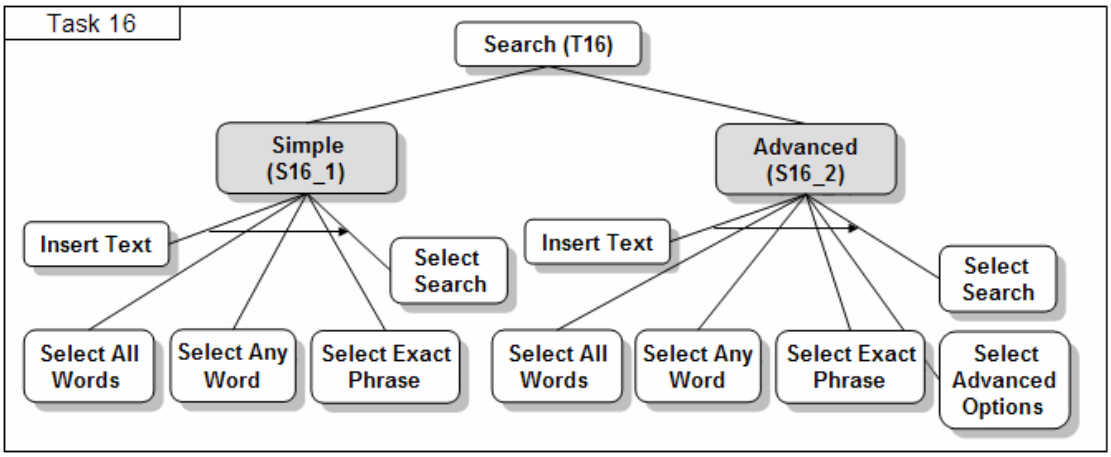


Figure 4.19: Polymorphic Task Hierarchy of "Search" task, "Simple" and "Advanced" styles.

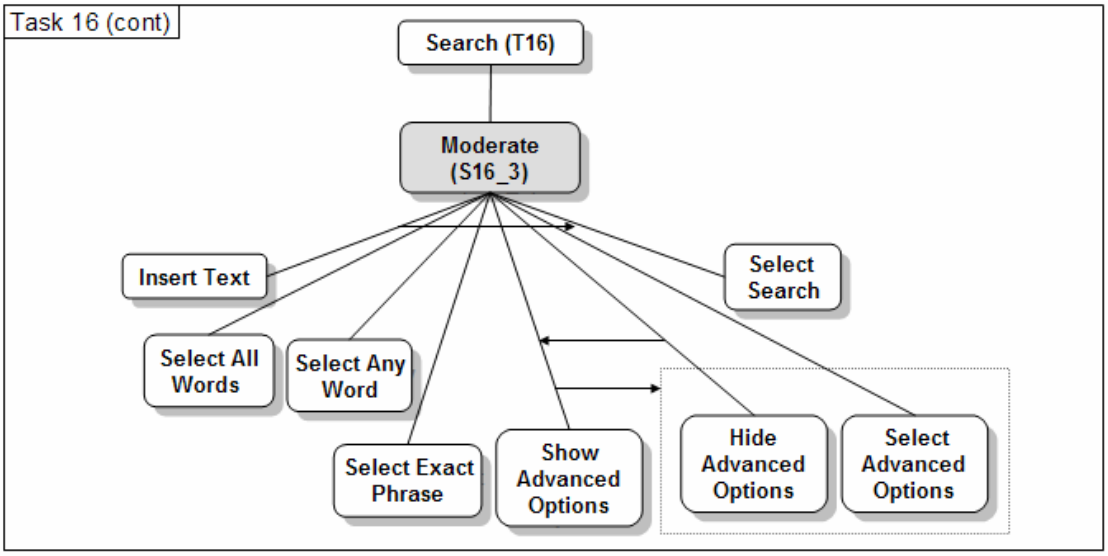


Figure 4.20: Polymorphic Task Hierarchy of "Search" task, "Moderate" style.

View Pie/Bar Chart

In Figure 4.21 the task hierarchy of the "View Pie/Bar Chart" task is displayed. The alternative display styles are "Image", "Protanope Image", "Deuteranope Image", "Tritanope Image" and "Table". The "Image" styles (S17_1-S17_4) display the chart as an image and use different color pallets in order to support different types of color blindness. The alternative "Table" style produces a table based representation of the given chart. This is done by using a task already discussed here, the "View Table" task.

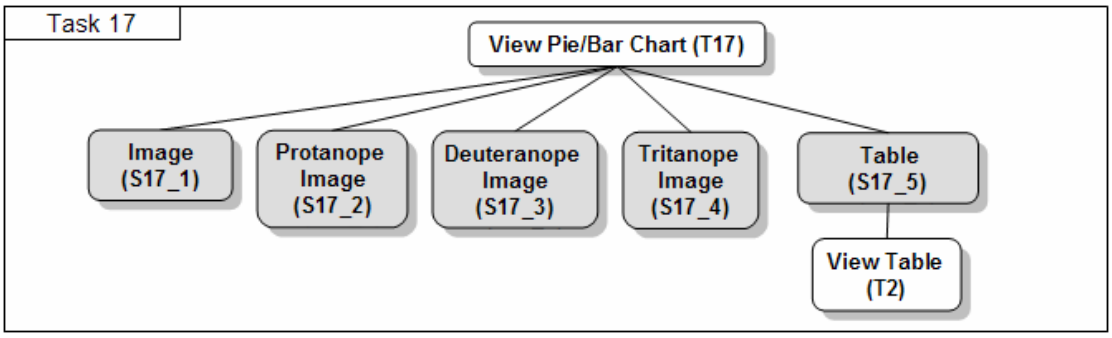


Figure 4.21: Polymorphic Task Hierarchy of "View Pie/Bar Chart" task.

4.1.3 Navigation Styles

All the task hierarchies presented so far belong to the same general category, as all tasks related to the page content. This section briefly describes navigation design in UniMob, reflecting the behaviour that most web users adopt. The user navigates to the desired web content and then uses this content. These two steps are clear and unambiguous. Therefore, in order to provide a complete adapted interface it is necessary to separate the navigation tasks from the content tasks. The U^2ID methodology is also used in the case of navigation tasks in order to provide a consistent adaptation platform.

In general, the navigation mechanism of the adapted site can be separated into two areas. The first is the "Navigation Path" area and the second is the "Selections" area. The path area consists of a path with links that the user can select in order to move back to previous navigation levels. The selections area consists of forward steps to the navigation hierarchy. These selections can be categories, subcategories or options and depend on the level of navigation the user is currently in.

Display Path

Figure 4.22 shows the task hierarchy of the "Display Path" task. This task applies to mobile devices and has only one style (PDA Path). The possible alternative styles refer to desktop environments. The path options are displayed and the user simply selects a path, then the path options are updated.

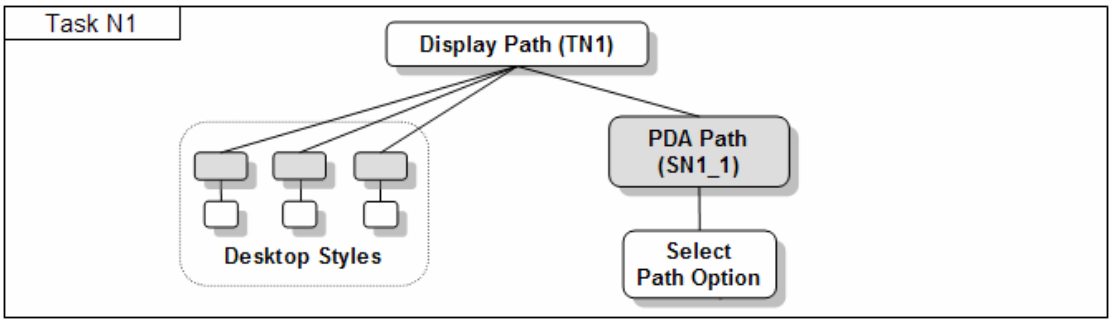


Figure 4.22: Polymorphic Task Hierarchy of "Display Path" task.

Selection Tasks

The task hierarchies that will be presented from now on refer to tasks that appear in "Selections" area of the navigation. In general the navigation elements of this area are designed to support two styles, "List" and "Thumbs". All of them have these styles as the only alternatives. The available level-specific options are displayed, and the user selects one from the available options.

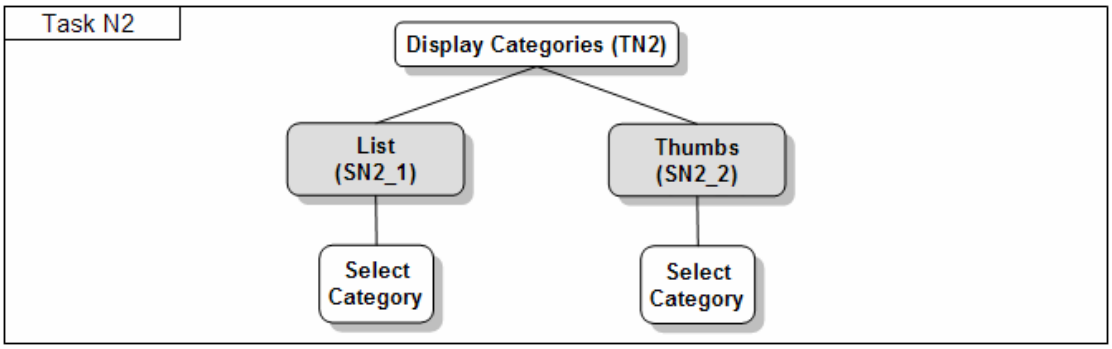


Figure 4.23: Polymorphic Task Hierarchy of "Display Categories" task.

Figure 4.23 shows the task hierarchy of the "Display Categories" task. This task is executed at the highest level in the navigation hierarchy.

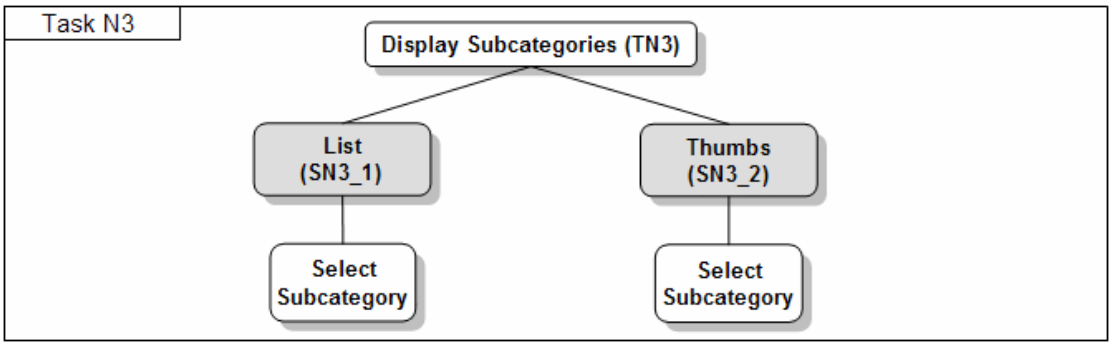


Figure 4.24: Polymorphic Task Hierarchy of "Display Subcategories" task.

The “Display Subcategories” (Figure 4.24) and “Display Options” (Figure 4.25) tasks appear after the “Display Categories” task in the navigation hierarchy. The lowest level task is the “Display Options” task.

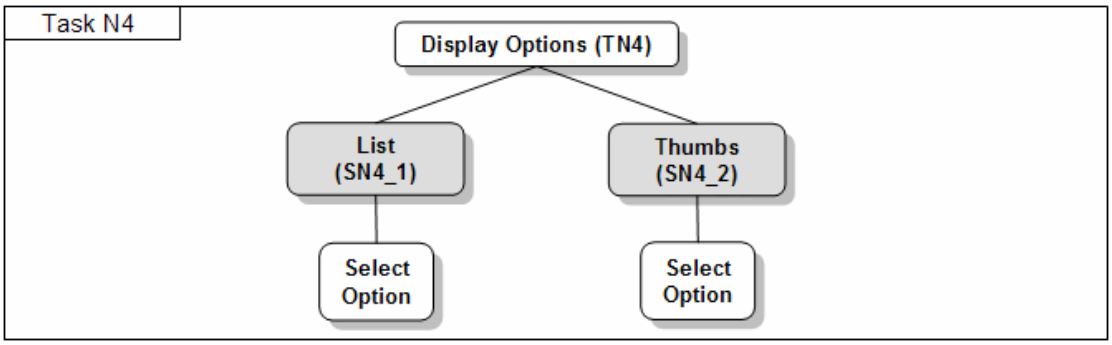


Figure 4.25: Polymorphic Task Hierarchy of “Display Options” task.

The final two tasks “Display Top Menu” (Figure 4.26) and “Display Bottom Menu” (Figure 4.27) are actually display options with level equal to the level of categories but are accessed through a different path from the navigation hierarchy.

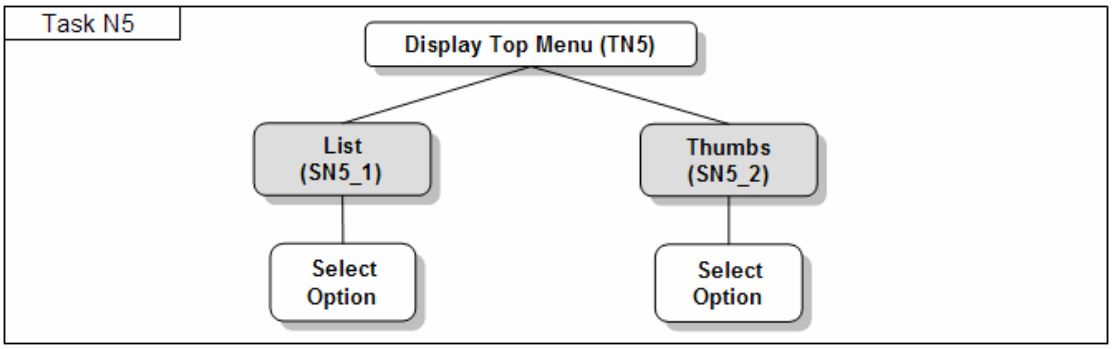


Figure 4.26: Polymorphic Task Hierarchy of “Display Top Menu” task.

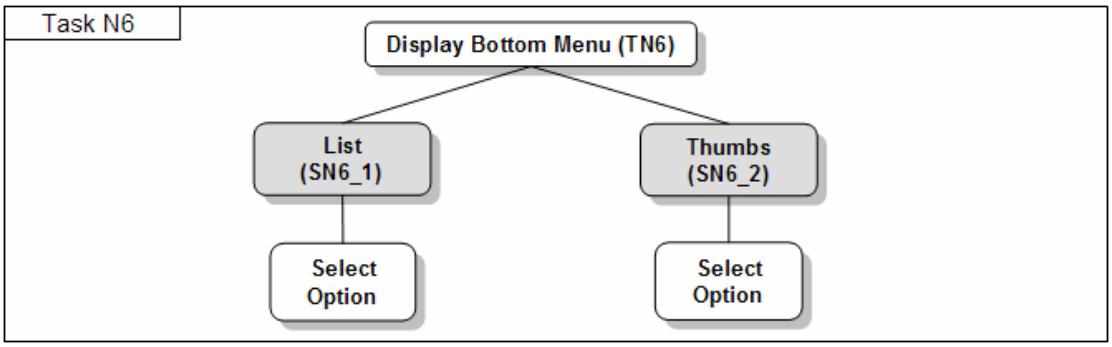


Figure 4.27: Polymorphic Task Hierarchy of “Display Bottom Menu” task.

These tasks complete the task hierarchies of the adapted components. The rest of this section will focus on the design rationale of the alternative task styles.

4.1.4 Design Rationale

The third step of the U^2ID methodology is the recording of the design rationale. This recording defines the parameters of each alternative style, the aims and the design relations between the styles. Some tasks appear to have only one style. It is assumed that the alternative styles of these single-style tasks are tailored to desktop browsing, so they are not presented here. The parameters in the following tables are displayed as disability - expertise pairs.

Table 4.1: Task "View Image".

TASK: View Image (T1)			
Style	Adapted Color Image (S1_1)	Adapted Grayscale Image (S1_2)	Adapted Pure Black-White Image (S1_3)
Targets	naturalness, better utilization of screen size, bandwidth saving, good quality and normal size	naturalness, better utilization of screen size, bandwidth saving, grayscale quality and medium size	naturalness, better utilization of screen size, bandwidth saving, poor quality and small size
Parameters	platform (mobile) (low vision - all) (motor impaired - all) (none - moderate, novice)	platform (mobile) (low vision - all) (motor impaired - all) (none - moderate, novice)	platform (mobile) (low vision - all) (motor impaired - all) (none - moderate, novice)
Relationships	Exclusion	Exclusion	Exclusion

TASK: View Image (T1) (cont)			
Style	Link (S1_4)	Thumbnail Image (S1_5)	Thumbnail Link (S1_6)
Targets	speed, bandwidth saving, compact view	speed, bandwidth saving, better utilization of screen size, image preview	speed, bandwidth saving, better utilization of screen size, compact view
Parameters	platform (mobile) (blind - all) (none - none)	platform (mobile) (all - all, directly selected)	platform (mobile) (all - all, directly selected)
Relationships	Exclusion	Compatibility	Compatibility

The Thumbnail styles (S1_5, S1_6) are created in order to be used in the construction of the "Display Images" task. These styles are not matched to any specific disability -

expertise pair (depicted as all-all) because they provide a very compact view, although they can be selected directly by the user.

Table 4.2: Task "View Table".

TASK: View Table (T2)		
Style	Linearized (S2_1)	Custom Table (S2_2)
Targets	safety, speed, minimize communication with server	speed, compact view, better utilization of screen size, guided steps, bandwidth saving
Parameters	platform (mobile) (blind - all) (low vision - all) (motor impaired - all) (none - novice)	platform (mobile, desktop) (none - all)
Relationships	Exclusion	Compatibility

Table 4.3: Task "View Paged Content".

TASK: View Paged Content (T3)			
Style	Links (S3_1)	Dropdown (S3_2)	No Paging (S3_3)
Targets	naturalness, bandwidth saving, limited available choices	speed, bandwidth saving	naturalness, safety
Parameters	platform (mobile) (none, motor impaired - high, moderate)	platform (mobile) (none, motor impaired - low)	platform (mobile) (blind - all) (low vision - all)
Relationships	Exclusion	Exclusion	Exclusion

Table 4.4: Task "Select Date".

TASK: Select Date (T4)			
Style	Direct (S4_1)	Indirect (S4_2)	Calendar (S4_3)
Targets	speed, minimize task steps, minimize communication with server	guided steps, safety	flexibility, safety, compact view
Parameters	platform (mobile) (blind - all) (motor impaired - all) (none, low vision - moderate)	platform (mobile) (none, low vision - novice)	platform (mobile) (none, low vision - expert)
Relationships	Exclusion	Exclusion	Exclusion

Table 4.5: Task "Select Time".

TASK: Select Time (T5)		
Style	Direct (S5_1)	Indirect (S5_2)
Targets	speed, minimize task steps, minimize	guided steps, safety

	communication with server	
Parameters	platform (mobile, desktop) (blind - all) (motor impaired - all) (none - expert) (low vision - expert)	platform (mobile) (none - novice, moderate) (low vision - novice, moderate)
Relationships	Exclusion	Exclusion

Table 4.6: Task "Upload File".

TASK: Upload File (T6)			
Style	Direct (S6_1)	Mixed (S6_2)	Indirect (S6_3)
Targets	speed, minimize task steps, minimize communication with server	speed, safety	guided steps, safety
Parameters	platform (mobile) (all - expert)	platform (mobile) (all - moderate)	platform (mobile) (all - novice)
Relationships	Exclusion	Exclusion	Exclusion

Table 4.7: Task "Upload Image".

TASK: Upload Image (T7)			
Style	Direct (S7_1)	Mixed (S7_2)	Indirect (S7_3)
Targets	speed, minimize task steps, minimize communication with server	speed, safety	guided steps, safety
Parameters	platform (mobile) (all - expert)	platform (mobile) (all - moderate)	platform (mobile) (all - novice)
Relationships	Exclusion	Exclusion	Exclusion

Table 4.8: Task "View Functions".

TASK: View Functions (T8)			
Style	Multiple Help (S8_1)	Single Help (S8_2)	No Help (S8_3)
Targets	minimize task steps, limited available choices, minimize communication with server	compact view, bandwidth saving	bandwidth saving, speed
Parameters	platform (mobile) (all - novice)	platform (mobile) (all - moderate)	platform (mobile) (all - expert)
Relationships	Exclusion	Exclusion	Exclusion

Table 4.9: Task "Display Files".

TASK: Display Files (T9)		
Style	Compact (S9_1)	Detailed (S9_2)
Targets	compact view, bandwidth saving	naturalness

Parameters	platform (mobile) (all - expert, moderate)	platform (mobile) (all - novice)
Relationships	Exclusion	Exclusion

Table 4.10: Task "Display Images".

TASK: Display Images (T10)		
Style	Detailed (S10_1)	Thumbnails (S10_2)
Targets	compact view, bandwidth saving	naturalness
Parameters	platform (mobile) (all - novice)	platform (mobile) (all - expert, moderate)
Relationships	Exclusion	Exclusion

Table 4.11: Task "View Options".

TASK: View Options (T11)	
Style	Simple (S11_1)
Targets	compact view, better utilization of screen size, bandwidth saving
Parameters	platform (mobile) (all - all)
Relationships	Exclusion

Table 4.12: Task "View Fieldset".

TASK: View Fieldset (T12)	
Style	Simple (S12_1)
Targets	compact view, better utilization of screen size, cross-browser compatibility, bandwidth saving
Parameters	platform (mobile) (all - all)
Relationships	Exclusion

Table 4.13: Task "View Tabs".

TASK: View Tabs (T13)	
Style	Simple (S13_1)
Targets	compact view, better utilization of screen size, bandwidth saving
Parameters	platform (mobile) (all - all)
Relationships	Exclusion

Table 4.14: Task "Display List".

TASK: Display List (T14)		
Style	Normal Markup Text (S14_1)	Normal Markup Link (S14_2)
Targets	naturalness	naturalness
Parameters	platform (desktop) (all - all)	platform (desktop) (all - all)

Relationships	Exclusion	Exclusion

TASK: Display List (T14) (cont)		
Style	Compact Markup Text (S14_3)	Compact Markup Link (S14_4)
Targets	naturalness, cross-browser compatibility, speed	naturalness, cross-browser compatibility, speed
Parameters	platform (mobile) (all - all)	platform (mobile) (all - all)
Relationships	Exclusion	Exclusion

Table 4.15: Task "Display Paged Text".

TASK: Display Paged Text (T15)			
Style	Paged HTML (S15_1)	Paged Text (S15_2)	No Paging (S15_3)
Targets	speed, compact view, better utilization of screen size, flexibility, cross-browser compatibility	speed, compact view, better utilization of screen size, flexibility	naturalness, minimize server communication
Parameters	platform (mobile) (none, motor impaired - all)	platform (mobile) (none, motor impaired - all)	platform (desktop) (blind - all) (low vision - all)
Relationships	Compatibility	Compatibility	Exclusion

Table 4.16: Task "Search".

TASK: Search (T16)			
Style	Simple (S16_1)	Advanced (S16_2)	Moderate (S16_3)
Targets	limited available choices, safety	extended capabilities	safety, flexibility, extended capabilities
Parameters	platform (mobile) (all - novice)	platform (mobile) (all - expert)	platform (mobile) (all - moderate)
Relationships	Exclusion	Exclusion	Exclusion

Table 4.17: Task "View Pie/Bar Chart".

TASK: View Pie/Bar Chart (T17)			
Style	Image (S17_1)	Protanope Image (S17_2)	Deuteranope Image (S17_3)
Targets	better utilization of screen size, naturalness	better utilization of screen size, naturalness	better utilization of screen size, naturalness
Parameters	platform (mobile) (none, motor impaired - all)	platform (mobile) (color blind protanope - all)	platform (mobile) (color blind deuteranope - all)
Relationships	Exclusion	Exclusion	Exclusion

TASK: View Pie/Bar Chart (T17) (cont)		
Style	Tritanope Image (S17_4)	Table (S17_5)

Targets	better utilization of screen size, naturalness	better utilization of screen size, speed, compact view
Parameters	platform (mobile) (color blind tritanope - all)	platform (mobile) (blind, low vision - all)
Relationships	Exclusion	Exclusion

Table 4.18: Task "Display Path".

TASK: Display Path (TN1)	
Style	PDA Path (SN1_1)
Targets	better utilization of screen size, provide shortcuts, bandwidth saving, compact view
Parameters	platform (mobile) (all - all)
Relationships	Exclusion

Table 4.19: Task "Display Categories".

TASK: Display Categories (TN2)		
Style	List (SN2_1)	Thumbs (SN2_2)
Targets	speed, bandwidth saving, compact view	naturalness, intuitive choices, faster browsing
Parameters	platform (mobile) (blind - all) (none, motor impaired - expert)	platform (mobile) (low vision - all) (none, motor impaired - novice, moderate)
Relationships	Exclusion	Exclusion

Table 4.20: Task "Display Subcategories".

TASK: Display Subcategories (TN3)		
Style	List (SN3_1)	Thumbs (SN3_2)
Targets	speed, bandwidth saving, compact view	naturalness, intuitive choices, faster browsing
Parameters	platform (mobile) (blind - all) (none, motor impaired - expert)	platform (mobile) (low vision - all) (none, motor impaired - novice, moderate)
Relationships	Exclusion	Exclusion

Table 4.21: Task "Display Options".

TASK: Display Options (TN4)		
Style	List (SN4_1)	Thumbs (SN4_2)
Targets	speed, bandwidth saving, compact view	naturalness, intuitive choices, faster browsing
Parameters	platform (mobile) (blind - all) (none, motor impaired - expert)	platform (mobile) (low vision - all) (none, motor impaired - novice, moderate)

Relationships	Exclusion	Exclusion

Table 4.22: Task "Display Top Menu".

TASK: Display Top Menu (TN5)		
Style	List (SN5_1)	Thumbs (SN5_2)
Targets	speed, bandwidth saving, compact view	naturalness, intuitive choices, faster browsing
Parameters	platform (mobile) (blind - all) (none, motor impaired - expert)	platform (mobile) (low vision - all) (none, motor impaired - novice, moderate)
Relationships	Exclusion	Exclusion

Table 4.23: Task "Display Bottom Menu".

TASK: Display Bottom Menu (TN6)		
Style	List (SN6_1)	Thumbs (SN6_2)
Targets	speed, bandwidth saving, compact view	naturalness, intuitive choices, faster browsing
Parameters	platform (mobile) (blind - all) (none, motor impaired - expert)	platform (mobile) (low vision - all) (none, motor impaired - novice, moderate)
Relationships	Exclusion	Exclusion

4.2 Implemented Mobile Styles

This subsection describes the implemented mobile styles of the adapted components (the implemented dialogue patterns of the DPC). The functionality of each style is presented along with some interesting implementation issues.

All the alternative styles are implemented in the C# programming language under the Microsoft .NET Framework. A typical class hierarchy of the styles of a task is displayed in Figure 4.28. All task styles are located in the ics.Adaptation. DialogueControls namespace. Each task component has a base class (icsDatePickerBase in figure) that all alternative styles (for desktop and mobile adaptation) inherit. This class encapsulates common properties and functionality. All

ics.Adaptation.DialogueControls.DatePickers Hierarchy

System.Object
- ics.Adaptation.DialogueControls.DatePickers.month
- System.Web.UI.Control ---- System.ComponentModel.IComponent, System.IDisposable, System.Web.UI.IDataBindingsAccessor, System.Web.UI.IParserAccessor
  - System.Web.UI.TemplateControl ---- System.Web.UI.INamingContainer
    - System.Web.UI.UserControl ---- System.Web.UI.IAttributeAccessor, System.Web.UI.IUserControlDesignerAccessor
      - ics.Adaptation.DialogueControls.DatePickers.icsDatePickerBase
        - ics.Adaptation.DialogueControls.DatePickers.icsDatePickerDropDownAutoEnforcement
        - ics.Adaptation.DialogueControls.DatePickers.icsDatePickerDropDownManualCheck
        - ics.Adaptation.DialogueControls.DatePickers.icsDatePickerGraphical
        - ics.Adaptation.DialogueControls.DatePickers.icsDatePickerPDADropDown
        - ics.Adaptation.DialogueControls.DatePickers.icsDatePickerPDADropDownSteps
        - ics.Adaptation.DialogueControls.DatePickers.icsDatePickerPDAGraphical
        - ics.Adaptation.DialogueControls.DatePickers.icsDatePickerTextBoxAutoEnforcement
        - ics.Adaptation.DialogueControls.DatePickers.icsDatePickerTextBoxManualCheck

Figure 4.28: Class hierarchy of the "Select Date" styles implementation.

style classes are used as UserControls. This figure shows five date picker styles tailored to desktop adaptation (*EAGER* toolkit) and three styles tailored to mobile adaptation (*UniMob*). This is an actual screenshot from the produced documentation of the *UniMob* system[1].

Each style builds and aligns programmatically all the elements that appear in its final form. The elements used are WebControls, HtmlControls or other styles in the ics.Adaptation.DialogueControls namespace. This namespace has a total of 24 base classes and 58 alternative mobile styles. The following sections present the implemented alternative styles of the different user task components.

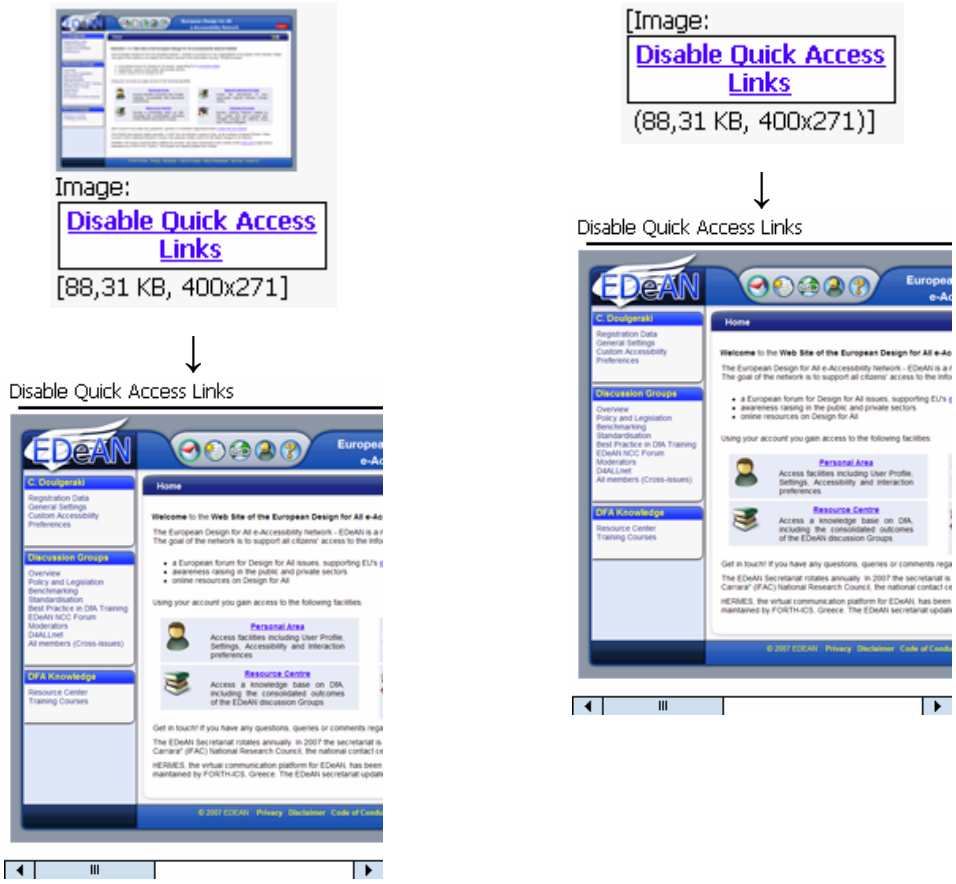
"View Image" Styles

The implemented alternative styles for this task are shown in Figures 4.29, 4.30 and 4.31.

The "Adapted Color Image" style (a) consists of an adapted image and the image details. The adapted image is created according to the following rule: if the dimensions of the original image are greater than the screen resolution of the device then the image is resized in order to fit the width and height of the screen. The developer can select the resampling method (GDI+[2]/custom method) and the filter to use (13 filters including Lanczos8, Mitchell, Quadratic, Box, Bell, etc.). The details that are shown beneath the adapted image inform the user about the size and the

[1] Documentation produced with NDoc 1.3 (http://ndoc.sourceforge.net/)
[2] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdicpp/GDIPlus/aboutGDIPlus.asp

dimensions of the original image. The user can view the original image by following the provided link. If the dimensions of the original are smaller than the screen resolution, then the original image is presented with no further details. The two alternative image styles (Grayscale and Pure Black-White) further decrease the size of



(a) Adapted Color Image (b) Link

Figure 4.29: Alternative styles of the "View Image" task.

the adapted image, thus reducing the download time. The pure black-white adapted image is shown in Figure 4.30a and the grayscaled adapted image in Figure 4.30b. The functionality of these styles is the same as the "Adapted Color Image" except the chromatic differentiation.

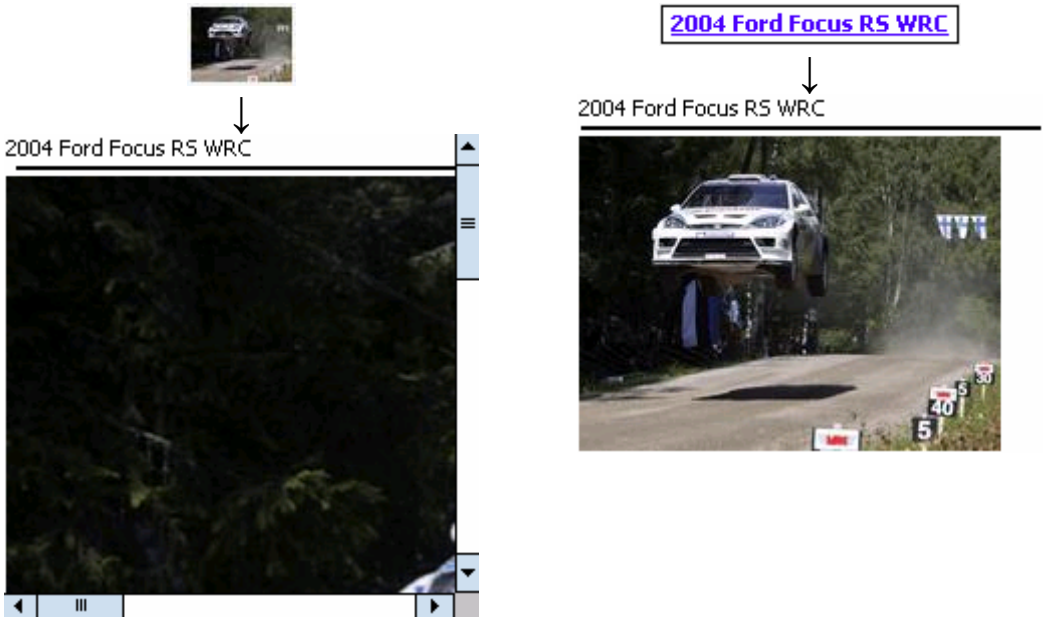
(a) (b) (c)

Figure 4.30: Alternative Adapted Image styles, “Pure Black-White” (a), “Grayscale” (b) and “Color” (c).

The second style is “Link” (Figure 4.29b) and differs from the first one because the image details are not accompanied with an image (adapted or original). This approach is more compact than the first one. However, the user can view the original image by following the provided link.

The “Thumbnail” styles are shown in Figure 4.31. As already stated in the design rationale of this task, these styles are used in the construction of the “Display Images” task. However, the user can select them from the relevant preferences menu. Thumbnail Image style (Figure 4.31a) provides a thumbnail link with size 1/3 of the screen dimensions. This link leads to the original image. The Thumbnail Link style (Figure 4.31b) provides a link to the adapted version of the original image. The adapted image is created according to the rules described above.

Adapted images and thumbnails are created only once when the page that contains them is requested for first time. The second time the same page is requested, the adapted images already exist and there is no need to create them again. Upon the end of the session for the current user (“logout”), all the created images are deleted. This approach saves server computational power and offers faster loading times.



(a) Thumbnail Image (b) Thumbnail Link

Figure 4.31: Displaying modes of the "Thumbnail" style.

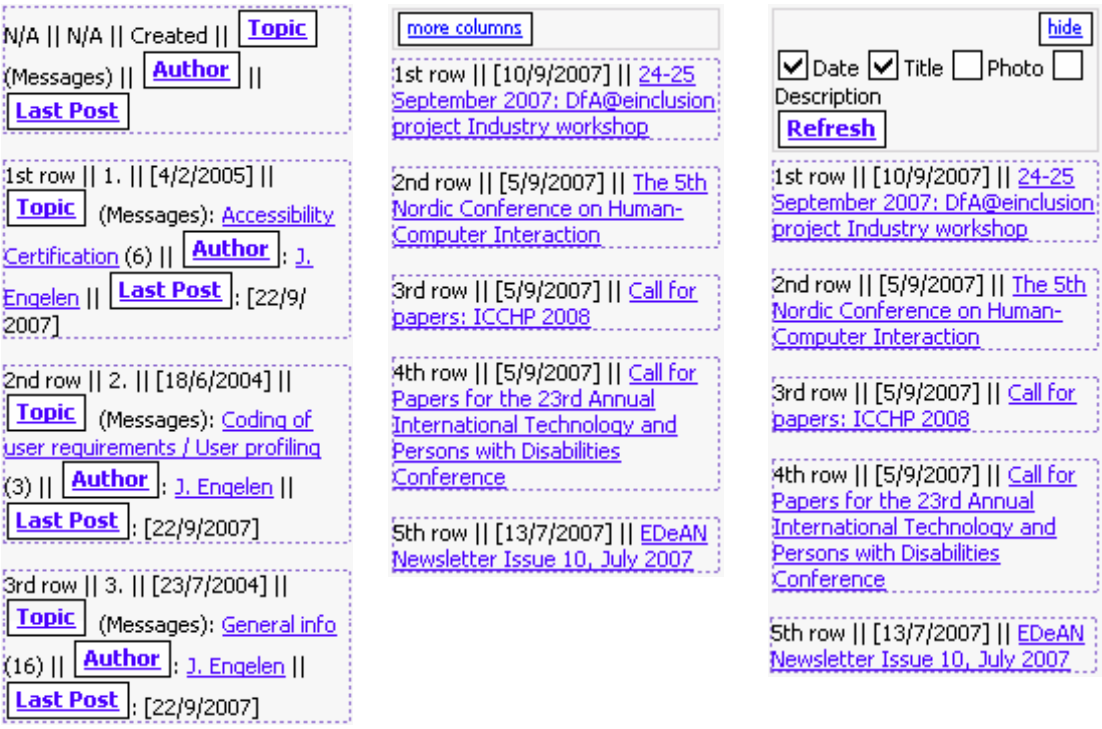
"View Table" Styles

This task has two alternative styles "Linearized" and "Custom Table" that are shown in Figure 4.32.

The linearized style (a) provides a row linearization representation of the table. The table is presented row by row and each cell is accompanied with its column name. The markup is simplified since the table is converted into blocks with break elements and horizontal lines. This style may not be very elegant, but prevents horizontal scrolling regardless the number of columns of the table.

In the second style (b, c) the approach is different. The user selects the columns of the table to be viewed. The table is initially displayed with some pre-selected columns. These columns are related to the role of the specific table and are chosen by the developer of the web page. For example, a table that stores the salaries of the employees of a company can contain the following entries: name, id, email, telephone, basic salary, overtime, bonus and total. In a mobile environment, where the displayed information must be accurate and useful, the entries that may be displayed are: name, email, telephone and total. This style supports such functionality. Moreover, the implementation of this component enables the selection of columns not only for the ordinary table view but it can also be applied to the linearized view and in any potential future table style. The desktop styles that can use this component include

"Row Linearization" (repeat title every cell, every line, no repeat) and "Column Linearization" (repeat title every cell, every line, no repeat). Its usage resembles the usage of an add-on component.



(a) Linearized (b) Custom Table (c) Custom Table (options)

Figure 4.32: Alternative styles of the "View Table" task.

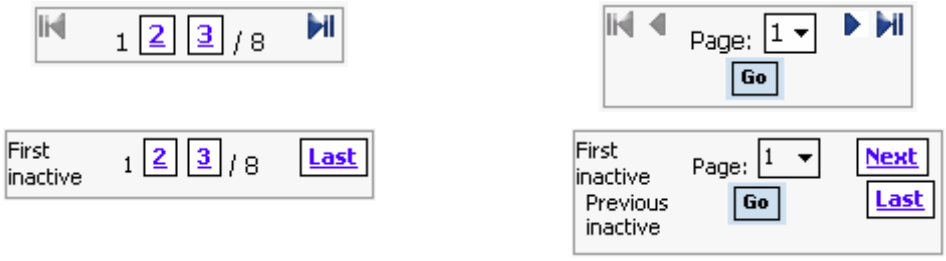
"View Paged Content" Styles

The alternative styles for this task are shown in Figure 4.33. These styles provide the underlying mechanism of paging functionality.

The first style is "Links" (a). In this case, pages appear as links that change according to the currently selected page. A "window" is moved over the row of page links and changes the visible ones. This ensures that any time the number of available pages is approximately the same. Links for the first and the last page are available as long as the page "jumps" someone can make through the page links are of two or three pages each time. Disabled links are displayed if all the pages can be displayed in the "window" of the page links.

In the second style (b) a drop down menu is used for the selection of the page to be displayed. Also direct links to the first, last, next and previous pages are available. The last style does not provide any paging functionality and all the available content is displayed at once.

Both styles "Links" and "Dropdown" are capable of displaying the direct links (first, last, next, previous pages) either as text links or as image links. This variation makes better utilization of horizontal space in case the number of pages is too large. This applies to both styles but mainly in the "Links" style. Apart from the practical benefit the arrow-style images that are used provide a more intuitive interaction for some users.



(a) Links (b) Dropdown

Figure 4.33: Alternative styles of the "View Paged Content" task.

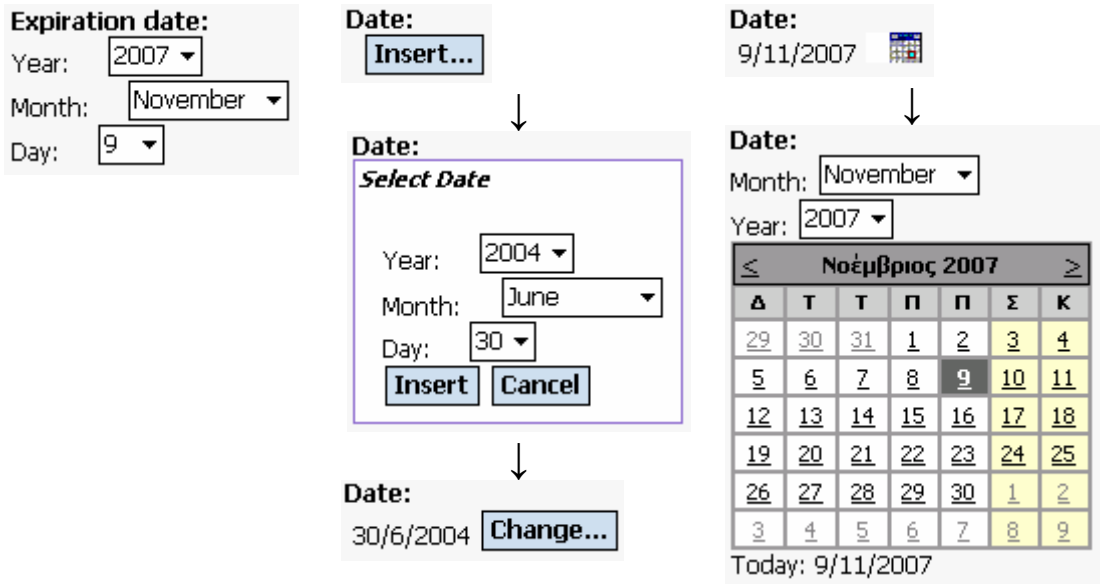
"Select Date" Styles

The three alternative implemented styles for this task are displayed in Figure 4.34.

In direct style (a) three drop down menus are used in order to select the year, the month and the day. When the year or the month selection is changed the component automatically (with javascript) alters the number of days to conform to the month-year combination. If the browser does not support javascript, the update to the days drop down is not possible. In this case the developer can use the provided function by the component to check the validity of the selected combination of day-month-year.

The indirect style (b) adopts a step by step process. The major step is the second one, in which the same drop down menus as in the direct style are used to select the day, month and year.

The third alternative style, the "Calendar" (c), also adopts a step by step process. In this style a graphical calendar is displayed along with two drop down menus one for the direct selection of the month and the other for the direct selection of the year.



(a) Direct (b) Indirect (c) Calendar

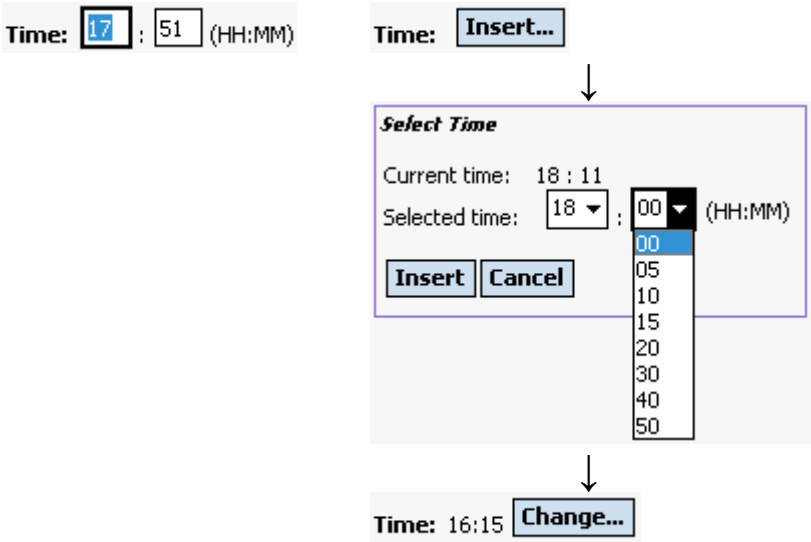
Figure 4.34: Alternative styles of the "Select Date" task.

"Select Time" Styles

Figure 4.35 shows the alternative styles of this task. Although the time selection task is quite simple in desktop browsers a careful design must be done in order to avoid the transformation of it into an awkward one when executed in mobile browsers.

Direct style (a) is straightforward and uses two textboxes where the user can insert the desired hours and minutes. When the focus is set to these textboxes the whole text is selected (with javascript) in order to avoid double selections or cursor movements inside the text box. This brings a reduction of two or three pen taps for each text box in a PDA device. In order to check the validity of the inserted time the developer can use the special function provided by the component.

The indirect style (b) adopts a step by step process. The user can select the hours and minutes from the drop down menus provided in the second step. These drop down menus ensure the validity of the selected time. The drop down menu for the selection of minutes contains a limited number of selections (00, 05, 10, 15, 20, 30, 40, 50) in order to avoid the crowding of the small screen with numbers.



(a) Direct (b) Indirect

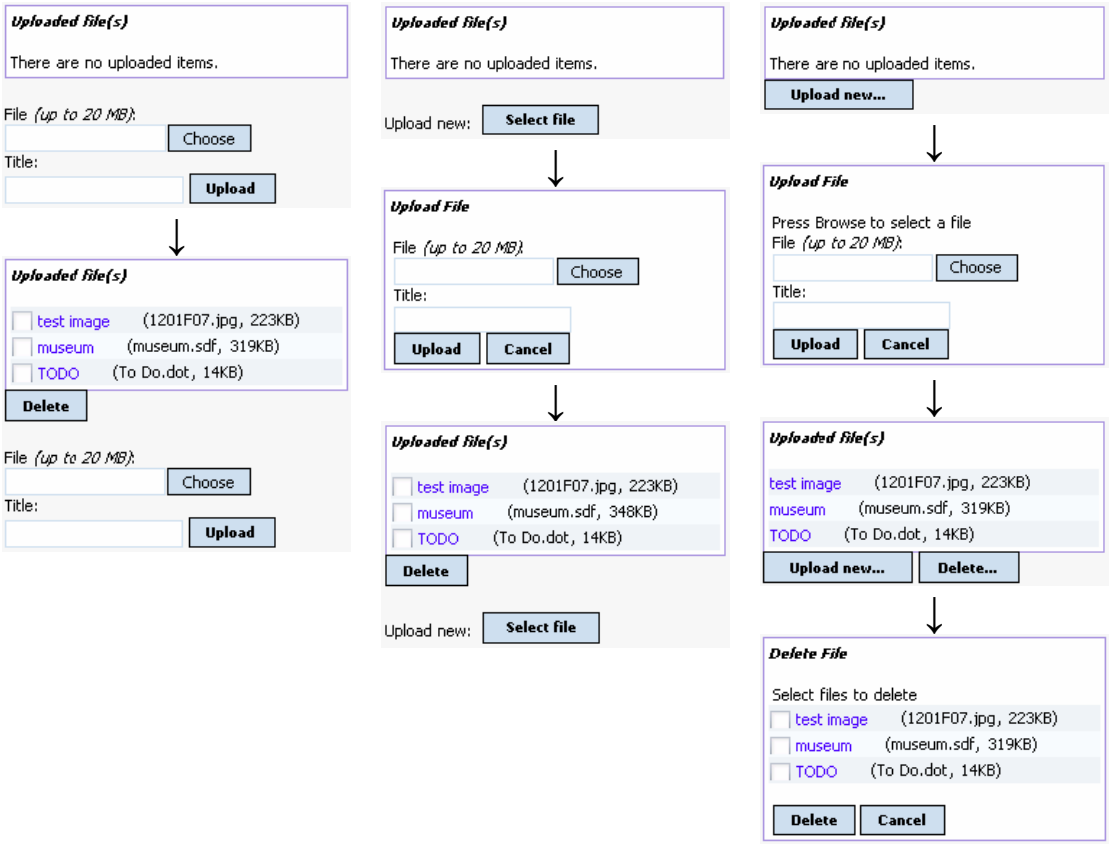
Figure 4.35: Alternative styles of the "Select Time" task.

"Upload File" Styles

The alternative styles for this task are shown in Figure 4.36. These styles enable users to upload files. The available functions are "Add file" and "Delete file". In all styles the user can view the filename, the size and has access to view or download the uploaded file by selecting the link provided in the uploaded files list. The developer must define a temporary location in which the uploaded files will be stored during the usage of the component and another location where the finally uploaded files will be stored.

In the "Direct" style (a), both functions can be used directly (no intermediate steps are required). Firstly, the user selects the file to upload and then inserts the desired title for this file. The "Upload" button performs the final uploading. In order to delete an uploaded file, the user must check the desired file(s) and then press the "Delete" button. Errors such as empty file location, empty title and no files checked are detected by the component and informative messages are displayed to the user.

Indirect style (c) adopts the opposite logic of the previous style. Intermediate steps are introduced in both functions in order to create a safe step by step process. Finally, the "Mixed" style (b) adopts only for the "Add file" function the step by step process leaving the deletion of files directly exposed to the user.



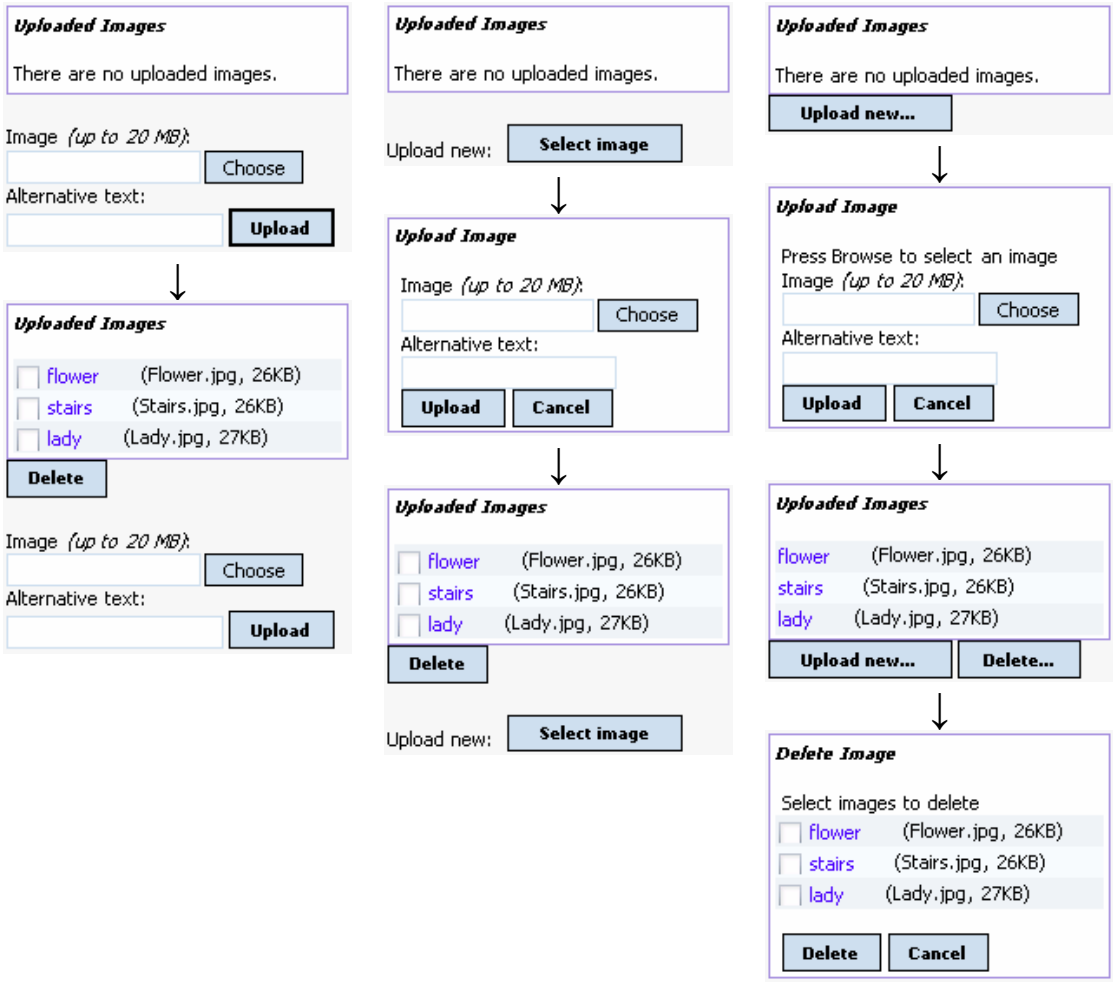
(a) Direct (b) Mixed (c) Indirect

Figure 4.36: Alternative styles of the "Upload Files" task.

"Upload Image" Styles

Figure 4.37 shows the alternative implemented styles for this task. The functionality and the implementation of these styles are similar to the "Upload Files" styles, and will not be presented again. However, some important aspects of these styles will be discussed.

All styles of this task use the "Thumbnail Link" style of the "View Image" task. This means that the links in the images list lead to the adapted version of the original image. When a new image is added to the uploaded images list, the original image is copied to the predefined temporary location and an adapted image is created. Similarly, when an uploaded image is deleted, the adapted image is also deleted. Apart from this difference, the "Upload Image" components check if the selected files are images (jpg, bmp, png, gif, tiff) and inform the user accordingly.



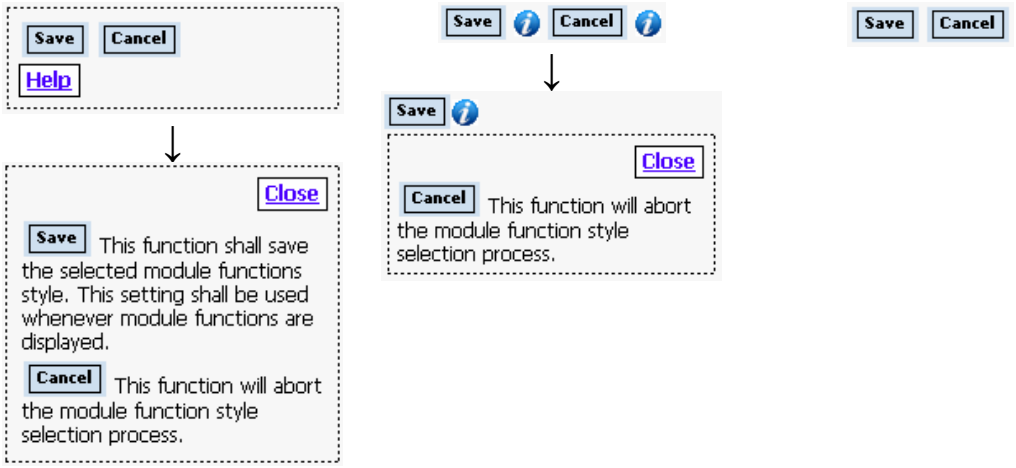
(a) Direct (b) Mixed (c) Indirect

Figure 4.37: Alternative styles of the "Upload Images" task.

"View Functions" Styles

The implemented styles for this task are shown in Figure 4.38.

The "Multiple Help" style (a) shows the available help instructions for all functions in one step. Users that are not familiar with these functions can benefit from this multiple presentation. In the second style, "Single Help" (b), users select to view the help instructions for a specific function by following the image marked with "*i*" (information) next to the desired function. The last style does not provide any help functionality. In all the above styles the developer can select the orientation of the buttons (horizontal/vertical) in order to better utilize the screen's space.



(a) Multiple Help (b) Single Help (c) No Help

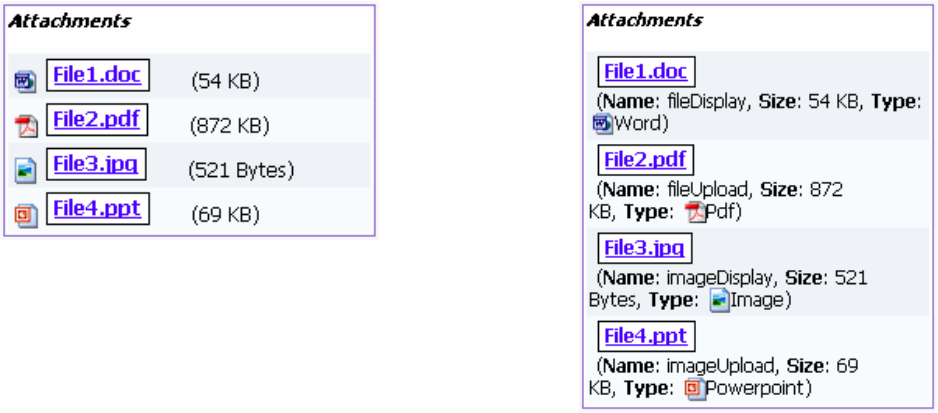
Figure 4.38: Alternative styles of the "View Functions" task.

"Display Files" Styles

This task has two alternative styles that are shown in Figure 4.39.

The first is the "Compact" style (a). In this style, the type of the file is marked with an icon in front of the file title link. This icon offers direct visual identification of the file type. However the user must be familiar with the associations of file types and icons. By clicking to the provided link the user can either view/open or download the selected file. The last element is the file size.

The other available style is the "Detailed" (b). This style uses labels in front of each file characteristic. Additionally, the file extension is added after the file type icon. Not only the title but also the name of the file is provided. This style is a more descriptive version of the "Compact" style, despite the fact that both of them provide almost the same information to the user.



(a) Compact (b) Detailed

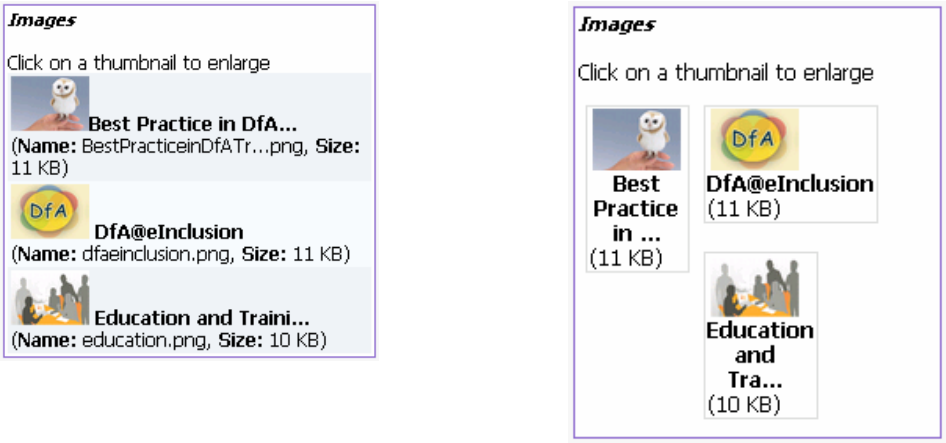
Figure 4.39: Alternative styles of the "Display Files" task.

"Display Images" Styles

This task offers the same functionality as the previous one, but it is targeted to displaying only image files. The alternative styles are shown in Figure 4.40.

In detailed representation (a), the "Thumbnail Image" style of the "View Image" task is used. This style provides a thumbnail at the left side of the images list for each image. The details of the images are tagged with labels in the same way as in the "Detailed" style of the "Display Files" task.

The other alternative is the "Thumbnails" style (b). This style resembles the view of a thumbnail gallery. Only the file size is provided to the user because the created thumbnails are links to the original images and it is necessary for the user to know the size of the file before trying to open it. The thumbnailed view provides a faster overview of the available images, as their number is considerably increased in relation to the displayed images in "Detailed" style.



(a) Detailed (b) Thumbnails

Figure 4.40: Alternative styles of the "Display Images" task.

"View Options" - "View Fieldset" - "View Tabs" Styles

The common characteristic of these styles is that they aim to achieve a consistent view in most mobile browsers. The styles for these tasks are displayed in Figure 4.41. In this figure the same styles are rendered in three popular mobile browsers. These styles replace complicated graphics with simple markup that is quickly and consistently rendered.

The "Simple" options style (a) consists of a rectangular area in which the available options are presented as simple links separated with the "|" character. The developer can set a title for the available group of options.

The Fieldset is an element widely used in web pages. Its normal style is replaced by a rectangular area with a label placed at the top left corner of it. This label has the role of a legend. The simplified fieldset fulfills the same aims as the fieldset element: to group and describe a collection of other elements. As it is easily seen in Figure 4.41 (b) the presentation of the "Simple" fieldset style is consistent in all the browsers.

Finally, the "Simple" tabs style (c) adopts an equal presentation form. The tabs are displayed as links that are separated with the "|" character. Each tab is marked with a rectangular area around the tab link. The tabs are separated from the content they alter with a horizontal line.

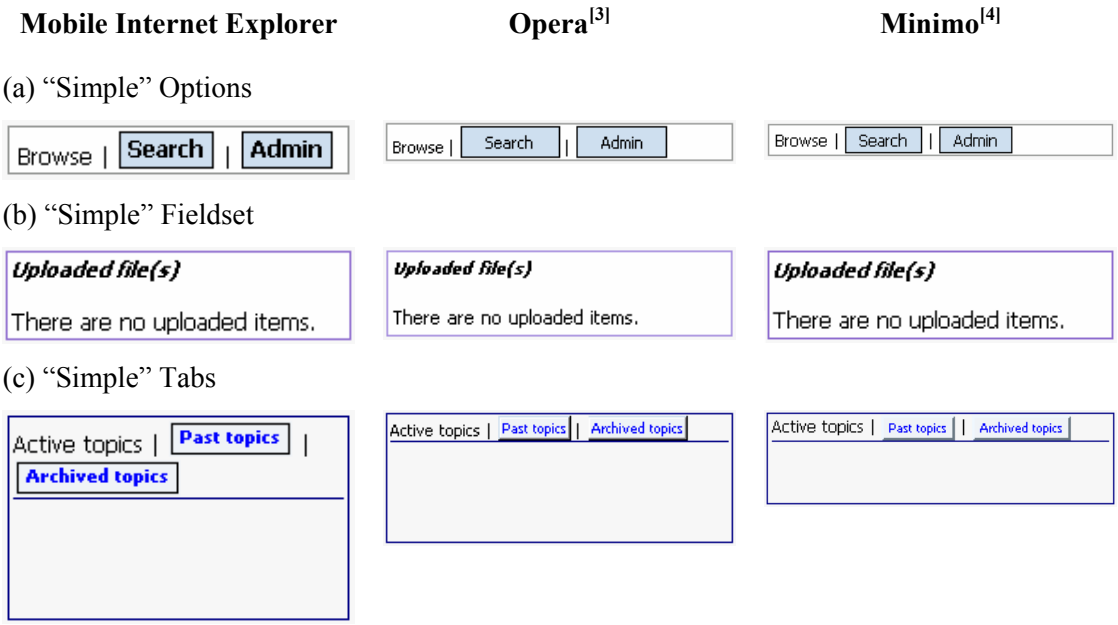


Figure 4.41: Alternative styles of the "View Options, Fieldset and Tabs" task.

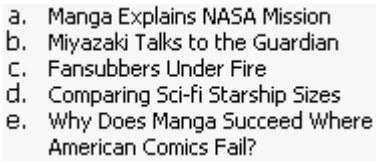
"Display List" Styles

The alternative styles for this task are shown in Figure 4.42. Ordered and unordered lists are widely used in web pages (table of contents, lists of news, lists of features, rankings) and the designed alternative styles aim to provide the common functionality adapted to the needs of mobile browsing. All styles support ordered and unordered lists. The "Text" styles (a, c) create lists of text elements and the "Link" styles (b, d) create lists of links. Also, the developer can set the numbering system for ordered lists and the bullet style for unordered lists.

[3] Opera v8.60 for Windows Mobile
[4] Minimo v0.2

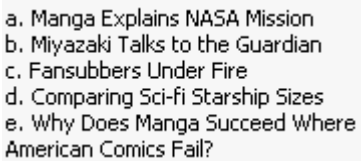
The "Normal Markup" styles (a, b) apply to desktop environments. Html elements ("ol, ul, li") are used for the construction of ordered and unordered lists.

The "Compact Markup" styles (c, d) transform the html elements ("ol, ul, li") into a combination of plain text (or html links), line break elements ("<br>") and non-breaking space characters (" "). The produced code is simple, shorter and in this way it can be rendered faster and consistently between mobile browsers. The benefit is proportional to the number of the elements of the list. The bigger the list, the greater the number of html elements that are removed. It is up to the developer to decide which lists could benefit from this type of transformation. In other words, to predict which lists are going to become overpopulated.



(a) Normal Markup Text

(b) Normal Markup Link



(c) Compact Markup Text

(d) Compact Markup Link

Figure 4.42: Alternative styles of the "Display List" task.

"Display Paged Text" Styles

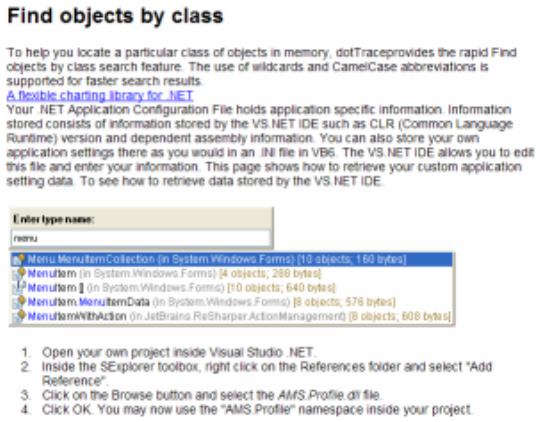
The aim of "Display Paged Text" task is to split a section of text or a section of HTML code into smaller (and in case of HTML into simpler) pieces that can be browsed with the "View Paged Content" styles that are already described. The number of split parts is defined by the developer. The alternative styles are presented in Figure 4.43.

The "Paged HTML" style (a, b) performs a simplification of the provided HTML code in order to split the textual content of the HTML code into pages. First, the possible textual content for each HTML element is extracted as follows:

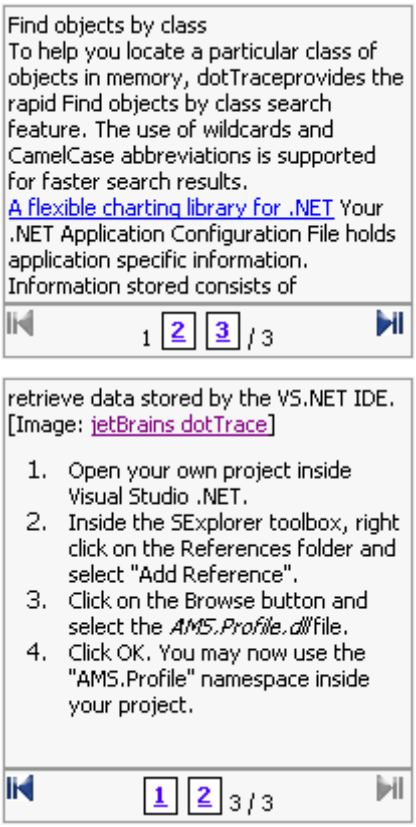
- in "div" elements no textual content is extracted
- in "span" elements the inner text is extracted
- named anchors are ignored
- in normal anchors the inner text is extracted

- in list elements the inner text is extracted
- in “br” and “hr” no text is extracted
- in image elements the alt text is extracted.

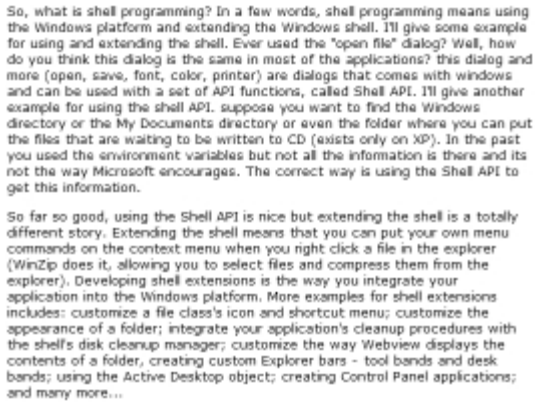
After the completion of this process, the text version of the HTML code is available and can be split into pages. At this stage, the elements of each page are selected. The final step is to construct the HTML code for the currently visible page based on the elements that are chosen to appear in it. In this reconstruction some elements are simplified and others remain unchanged. An identified “div” element is replaced with a “br” element, “span”, “br” and “hr” elements do not change, “ol”, “ul”, “li” elements also do not change, image elements are replaced with links to the original image tagged with an “Image” label in front of the link, and finally link elements remain the same and named anchors are ignored. Although this style offers advanced functionality, developers must carefully consider the HTML elements they are going to use, in order to preserve the desired behaviour in both styles, paged and non-paged.



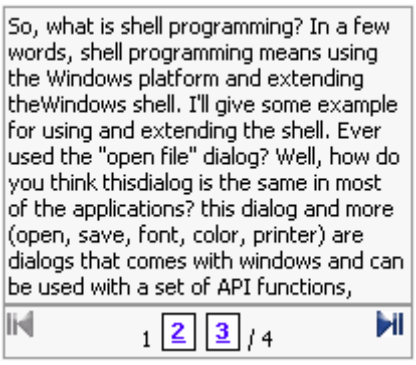
(a) Paged HTML (desktop)



(b) Paged HTML (PDA)



(c) Paged Text (desktop)



(d) Paged Text (PDA)

Figure 4.43: Alternative styles of the "Display Paged Text" task.

In "Paged Text" style (c, d) the text is split into pages in such a way that each page has approximately the same number of characters. This process is quite simple and the final text of each page has the same form as in the original text.

The "No Paging" style appears in desktop browsers in order to hide the functionality of this component, since it is of no use in desktop environments.

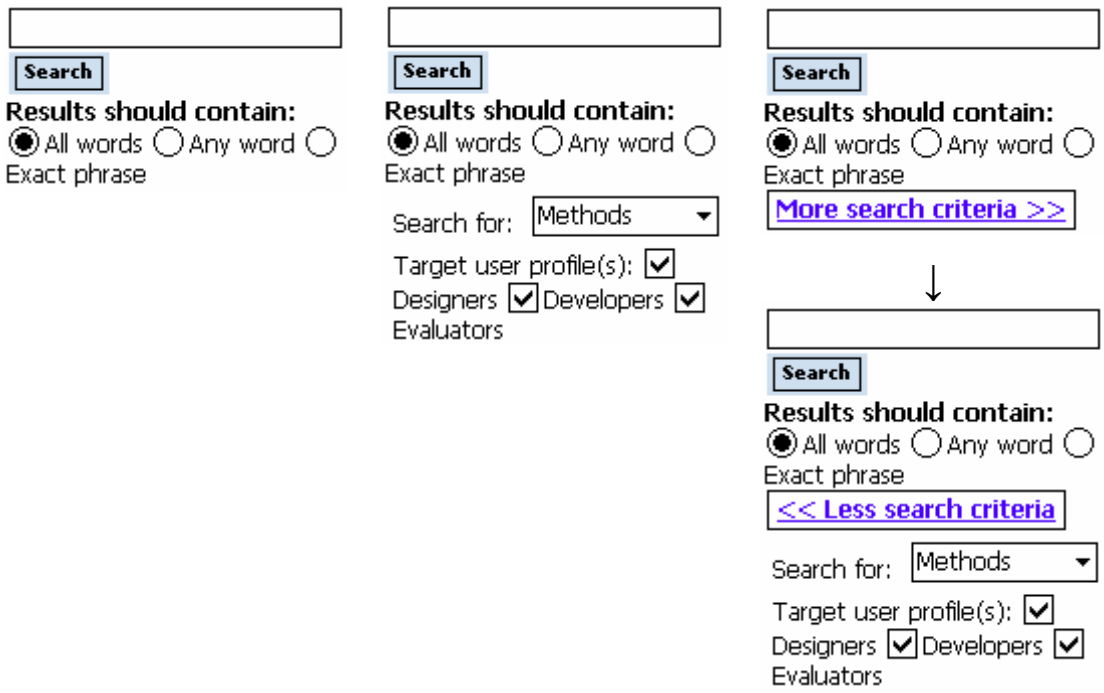
"Search" Styles

This task has three alternative styles that are shown in Figure 4.44.

The "Simple" style (a) provides the essential controls for search functionality. These controls include a textbox, a search button and three radio buttons to narrow down the search results. The user can insert up to ten keywords in the search textbox. It is the developer's responsibility to properly collect and display the search results.

The "Advanced" style (b) is enhanced with more search criteria. These criteria are added by the developer to the appropriate property of the advanced style component. As in the simple style, it is the developer's responsibility to display the search results.

The "Moderate" style (c) provides optional access to more search criteria. The user can either perform a simple or an advanced search.



(a) Simple (b) Advanced (c) Moderate

Figure 4.44: Alternative styles of the "Search" task.

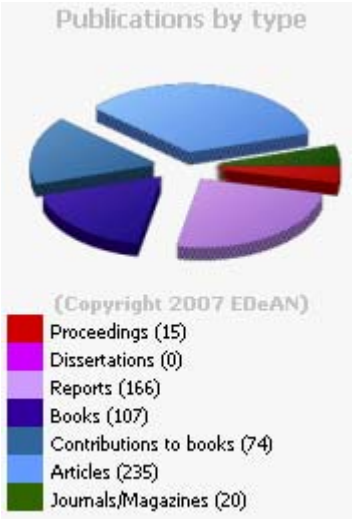
"View Pie/Bar Chart" Styles

The styles of this task aim at displaying pie and bar charts in a meaningful and compact manner. The approach is the same in both types of charts, therefore only the pie charts are discussed here. Figure 4.45 depicts the "Image" and the "Table" styles of this task.

The "Image" style consists of the chart title, the chart itself and the chart legend. The chart is created on the fly and displayed as an image. The dimensions of the chart image are defined by the developer before the creation of the image. The legend is placed at the bottom of the chart. In order to assist color blind users three alternative Image styles are available (Figure 4.46). These styles render the chart and the legend with different color palettes. These palettes enable users with protanopia[5], deuteranopia[6] and tritanopia[7] to distinguish the colors of the chart items and the legend.

In the "Table" style the chart data are displayed in table form. The "View Table" styles are used to create the data table. In this way, all the tables that appear in the portal (normal tables or chart tables) can use the same display style.

[5] Protanopia: One of the two varieties of red-green color blindness (red-dichromacy).
[6] Deuteranopia: One of the two varieties of red-green color blindness (green-dichromacy).
[7] Tritanopia: Yellow-blue color blindness.



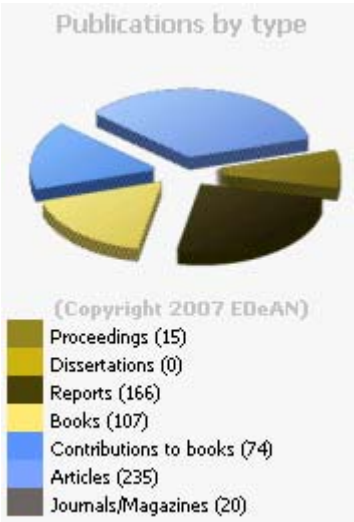
(a) Image

Publications by type

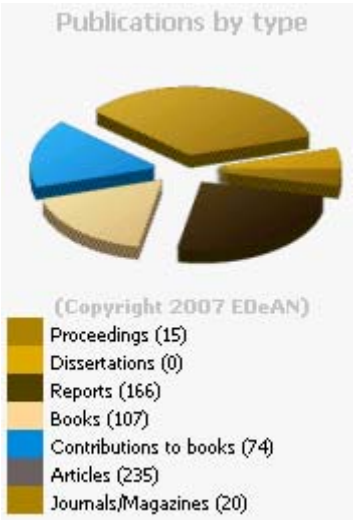
1st row	Proceedings	15
2nd row	Dissertations	0
3rd row	Reports	166
4th row	Books	107
5th row	Contributions to books	74
6th row	Articles	235
7th row	Journals/Magazines	20

(b) Table

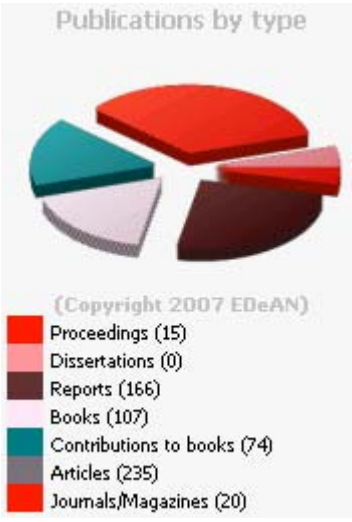
Figure 4.45: Image and Table styles of the "View Pie/Bar Chart" task.



(a) Protanope Image



(b) Deuteranope Image



(c) Tritanope Image

Figure 4.46: Alternative Image styles of the "View Pie/Bar Chart" task tailored to different types of color blindness.

Navigation styles share many common implementation characteristics and are hierarchically linked (category - subcategory - option). The "Display Path" style is tightly related to the other navigation styles and depends on the level of the user in the navigation hierarchy.

At this point it is important to describe the way the navigation is divided into stages. The first stage, named "Start", is the initial point where the user can select one from the three major navigation areas. The first area, named "Top Menu" contains the menu(s) that appear at the top side of the desktop version of the portal. The "Display

Top Menu" style provides access to the options of this area. The second area, named "Navigation", contains the menus that appear at the left side of the desktop version. The "Navigation" area contains some categories (menus) of areas. Similarly, these categories contain some subcategories of areas and these subcategories contain some options. These options have content that is displayed to the user. Options stand at the final stage of the navigation hierarchy. The "Display Categories, Subcategories, Options" styles provide access to the contents of this area. The last area, named "Bottom Menu", contains the menu(s) that appear at the bottom side of the desktop version. The "Display Bottom Menu" style provides access to the options of this area.

"Display Path" Style

This style is quite simple and provides bidirectional browsing capabilities to the user. The form of this style is shown in Figure 4.47 at different levels of the navigation hierarchy. The path component is constantly visible at the top side of the adapted portal. By selecting the "Start" link the user returns to the first stage of the navigation hierarchy. The styles that display the available options for each different level of the hierarchy are loaded at the bottom of the path component. In (a) the "Display Categories" style is loaded and in (b) the "Display Subcategories" is loaded. In (c) the Benchmarking WG has only one option and there is no need for displaying it as thumb at the bottom of the path. The new entries are added to the path as the user advances at the hierarchy by selecting options from the styles at the bottom of the path component. Entries are removed when the user selects a link from the path. This means that the user is moving at a previous stage in the hierarchy.

(a) Category (b) Subcategory (c) Option

Figure 4.47: The "Display Path" style in different levels of the navigation hierarchy.

"Display Categories" - "Display Subcategories" - "Display Options" Styles

The role of these styles in the construction of the path component was previously discussed. All these tasks ("Display Categories", "Display Subcategories", "Display Options") can have two different styles, "List" and "Thumbs". In Figure 4.48 the alternative styles of the "Display Categories" task are presented.

The first one is the "List" style (a). This style provides a compact view of the available options. Regardless the number of options, this list of links can be displayed very quickly. In "Thumbs" style (b) an icon is displayed for each of the available options. In order to select an option the user can tap with the stylus anywhere in the marked area of the option. The options that contain other options (lead to other options) have icons with the view of a folder and a mark relative to the content. On the other hand final options have icons that are relative to their content. In this way a more intuitive navigation is achieved.

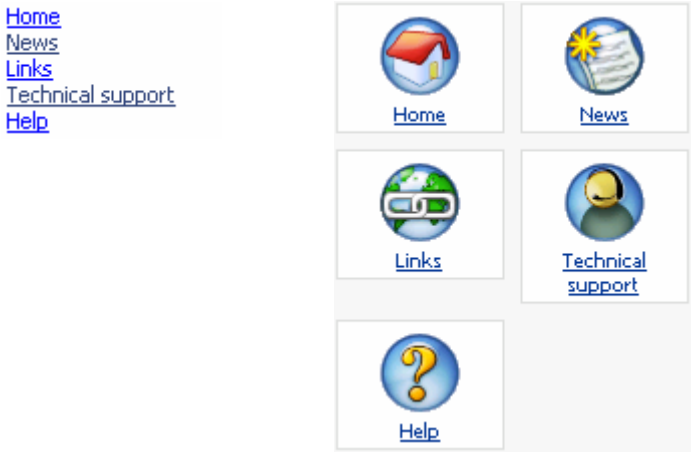
(a) List (b) Thumbs

Figure 4.48: Alternative styles of the "Display Categories" task.

"Display Top Menu" - "Display Bottom Menu" Styles

These tasks have also the same alternative styles, "List" and "Thumbs". Figure 4.49 depicts the alternative styles of the "Display Top Menu" task.

The implementation of these styles follows the same rules as the "Display Categories" style presented above. The difference is that the options of the top and bottom menus are final options, thus their icons are not folder-like and lead directly to content pages.



(a) List (b) Thumbs

Figure 4.49: Alternative styles of the "Display Top Menu" task.

PDA Page Template

An important aspect of the adapted interface, except the transformations of the navigation mechanism and the content, is the template used in the setup of the displayed page. The template of a page separates the page into areas. Each area contains specific elements throughout the pages of the site. Templates provide page consistency and save developing time.

A special PDA template was designed and developed for use in the mobile adaptation mechanism. The areas of the PDA template are shown in Figure 4.50. The Quick Access Links (QAL) are placed at the top and the bottom of the page. The site logo and the site title are placed in the same line above the QAL. The next line is visible only when the user is logged in. It contains the user name and the "Logout" button. The path control is placed above the navigation options area. Except the navigation options (categories, subcategories, options) this area contains the content of the final options (see previous paragraphs for distinction of options).

The PDA template provides equal functionality with the desktop templates. The controls are carefully placed in the areas of the template in order to ensure that the functions that are displayed first are the most useful to the user. Despite the fact that all the unnecessary controls of the desktop templates are omitted, the look and feel is preserved in the PDA template by means of template images, option names and portal skins. In this way the familiar user with the desktop version of the portal can easily shift to the mobile version.

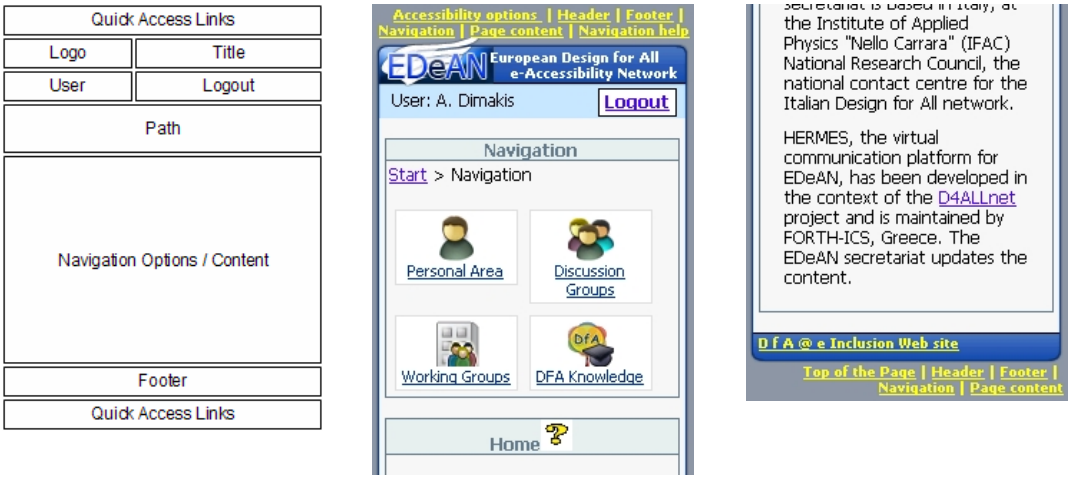


Figure 4.50: The template being used in the mobile adaptation (left). Two screenshots of the template with some controls loaded in it (right).

Navigation Search - Favorites

The navigation mechanism described in the previous paragraphs provides a linear navigation model. The user must return back to previous stages of the navigation menu in order to follow a different path.

Two additional components were added in order to enhance the site navigation (Figure 4.51). These components provide shortcuts to different navigation options. The first is the "Navigation Search" component. This component enables user to execute a keyword based search in the categories, subcategories, options of the navigation hierarchy. The second is the "Favorites" component. It provides shortcuts to the most frequently used options of the navigation hierarchy. In both components the results are displayed as paths in case the user needs access to previous levels of the hierarchy.

(a) (b)

Figure 4.51: "Navigation Search" and "Favorites" components.

4.3 Summary

This chapter presented the details of the design process of the alternative mobile styles. The U^2ID methodology was followed, and the results are presented accordingly. Next, the implemented styles were introduced along with some interesting implementation and functional details.

Chapter 5

Case Study: The EDeAN mobile portal

This Chapter reports a case study of development of the mobile version of a fully functional Web portal using the *UniMob* tool. The EDeAN portal is capable of desktop and mobile adaptations. The desktop adaptations derive from the usage of the *EAGER* toolkit and the mobile adaptations from the *UniMob*. The initial version of the portal supported only desktop adaptations. After the integration of the *UniMob's* extensions, the portal obtained its mobile alternative.

5.1 Introduction

The EDeAN portal is a prototype implemented by means of the *EAGER* toolkit and extended with the *UniMob* system in order to be capable of mobile adaptation. The European Design for All e-Accessibility Network (EDeAN) was established by the European Commission in 2002. The goal of this network is "to raise the profile of Design for All (DfA) and emphasize its importance in achieving greater accessibility to the Information Society for as many people as possible" (excerpt taken from the EDeAN portal). This portal will be used as a testbed in the evaluation process of the adapted mobile interface.

An essential precondition for the successful deployment of the proposed adaptation mechanism into a new portal is the existence of a separate interface layer responsible only for the presentation of the portal. In order to satisfy this need, the EDeAN portal is built as a three-tier application. The architecture used includes the following layers: Data Access-Storage Layer, Application (Business) Logic Layer and Presentation Layer. The three-tier architecture is commonly used in software engineering and provides scalability, reliability, platform independence and security.

The next section illustrates some screenshots from the desktop interface and the adapted version in the mobile device, explaining the differences between the two versions of the portal.

5.2 EDeAN Mobile User Interface

Figure 5.1 shows the mapping of the contents of the navigation menus that appear in the EDeAN portal into stages of the PDA navigation mechanism. For example, the categories of “Navigation” area are the “Personal Area”, the “DFA Knowledge”, the “Working Groups”, etc. The subcategories of “Personal Area” are the “General Settings”, the “Custom Accessibility”, etc. Finally, the only option of the “General Settings” subcategory is the “General Settings” option.

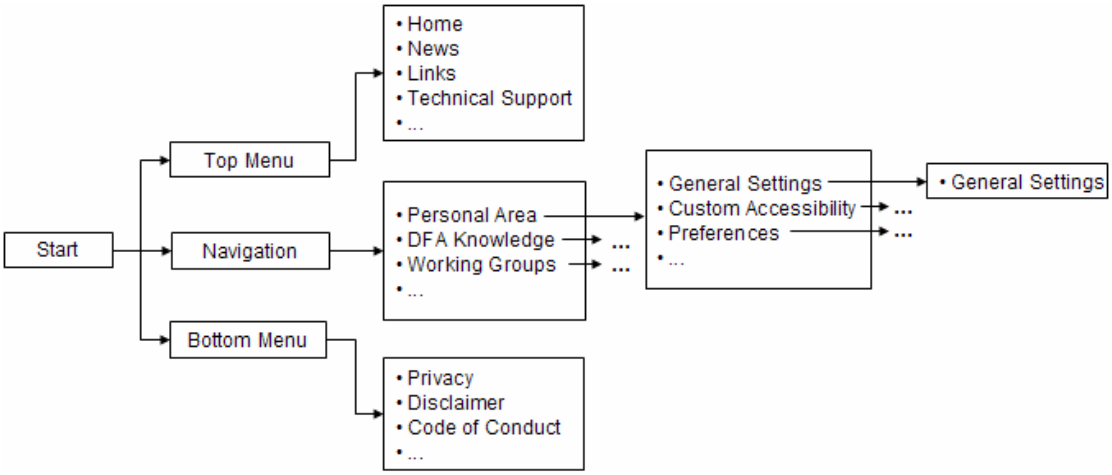


Figure 5.1: Navigation hierarchy of the EDeAN portal in PDA (moderator).

The mapping of the three major navigation areas of the PDA into physical areas of the desktop version of the EDeAN portal is shown in Figure 5.2.

Figure 5.2: Navigation areas in the EDeAN portal.

Figure 5.3: Logged out home page in desktop EDeAN.

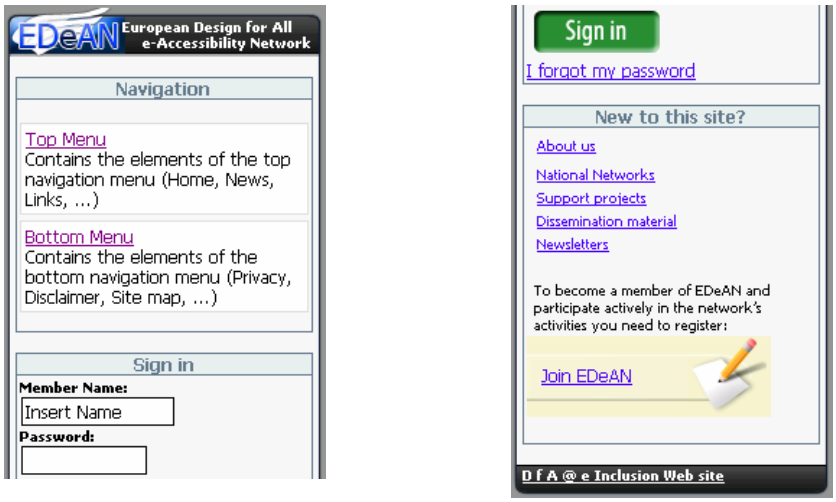


Figure 5.4: Logged out home page in mobile EDeAN.

The mobile version (Figure 5.4) provides a compact home page with the navigation alternative for the top and bottom menus and two essential functions "Sign In" and "Register". The two buttons that are used in these functions preserve the look and feel of the desktop version (Figure 5.3). The entire page is displayed with the PDA template and each area in the content is distinguished with a meaningful title.

Figure 5.5: Logged in home page in desktop EDeAN.

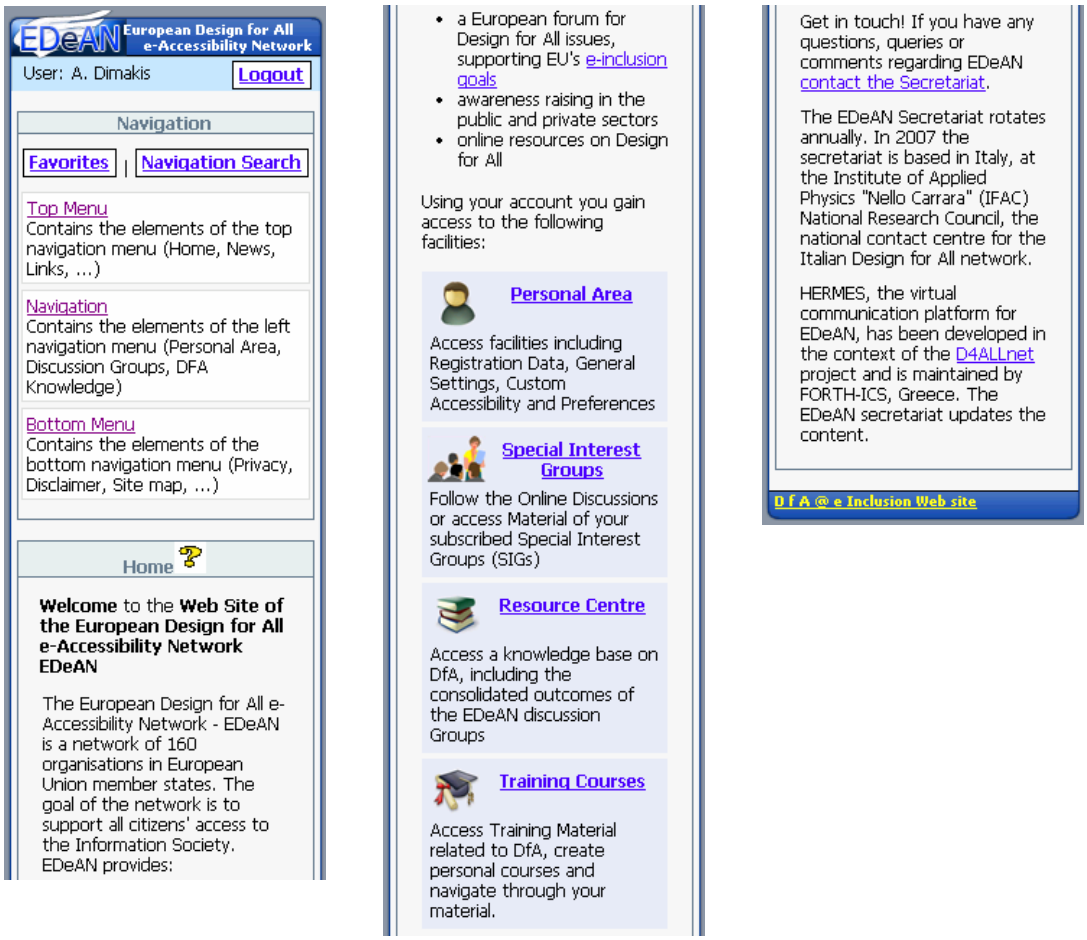


Figure 5.6: Logged in home page in mobile EDeAN.

When the user is logged in (Figure 5.6), the navigation area is enhanced with the "Favorites" and "Navigation Search" controls. Also, the option of the "Navigation" menu is added between the top and the bottom menu options. Each content area is accompanied with a link to a help page dedicated to the content of the specific area. The link to the printable version of the page is removed in the mobile version. The content is slightly rearranged and the images are reduced in size in order to fit in the small screen's width. All the content of the page that is available in the desktop version is also available in the mobile version.

Figure 5.7: Custom Accessibility area in desktop EDeAN.

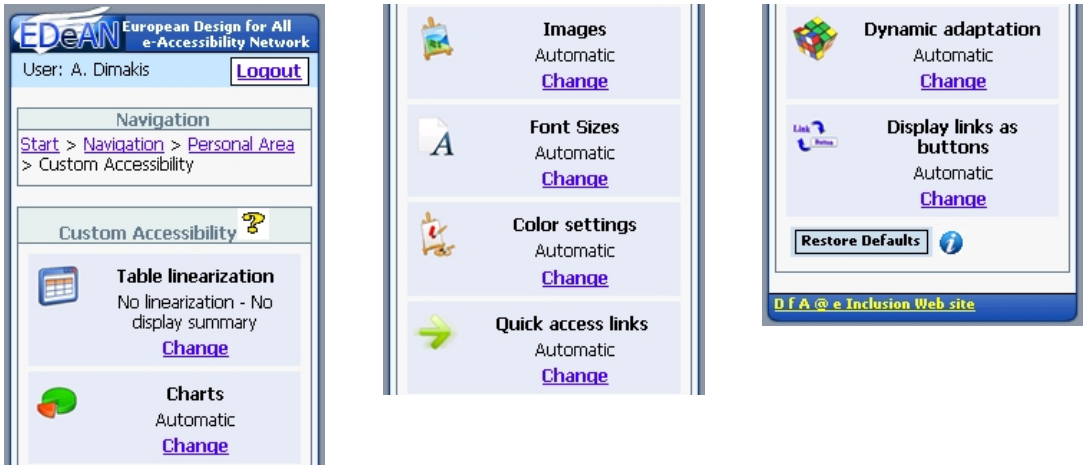


Figure 5.8: Custom Accessibility area in mobile EDeAN.

Figure 5.8 shows the "Custom Accessibility" area of the mobile EDeAN. Some preferences of the desktop page are not available in the mobile version. The mobile page contains the preferences that are used in the mobile adaptation process. Some of the removed preferences have only one alternative style (e.g., Fieldsets), others do not apply in the mobile device (e.g., Text To Speech, needs the browser speech plug-in that is not available in the Mobile Internet Explorer). Also, the "Display Functions" style of the desktop (indicated by the "Help" button near the function's button) is

replaced with the mobile equivalent (indicated by the image button with the "*i*" mark) that is appropriate for the current context of use (expertise, disability, etc.).

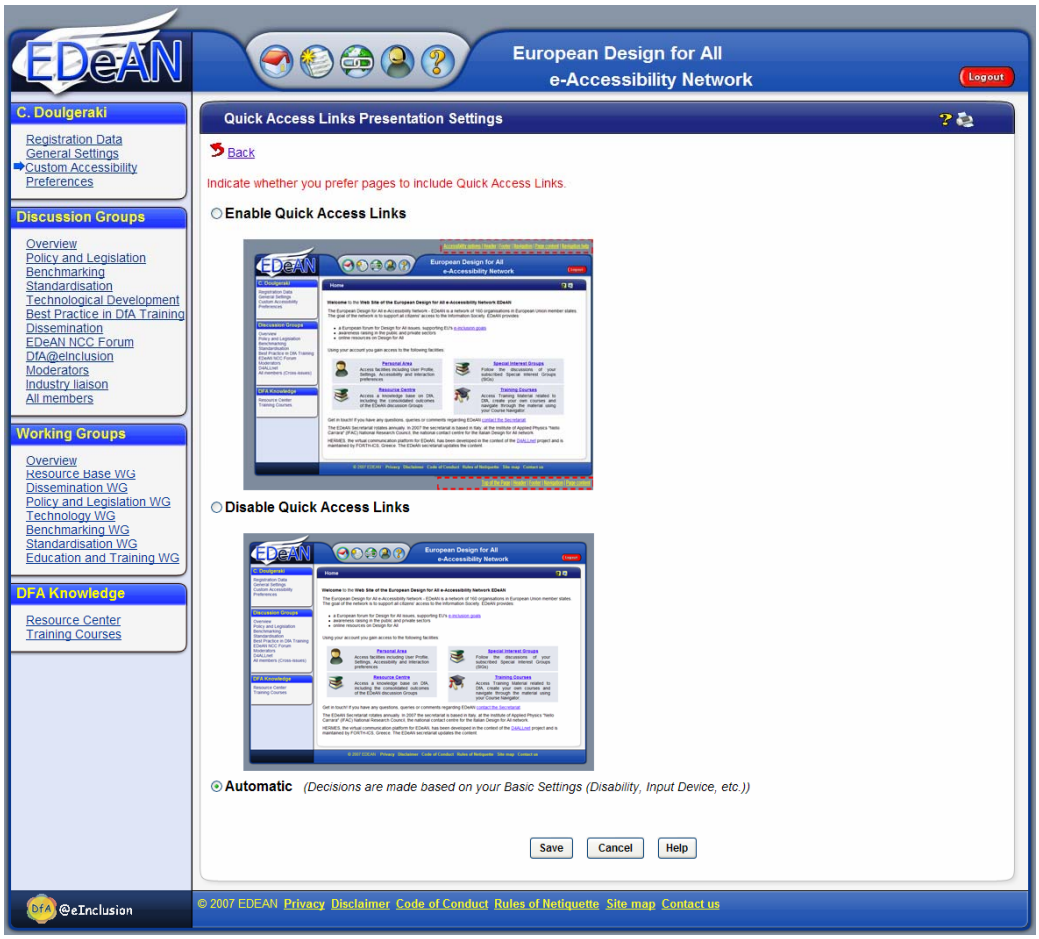


Figure 5.9: Modify Quick Access Links settings in desktop EDeAN.

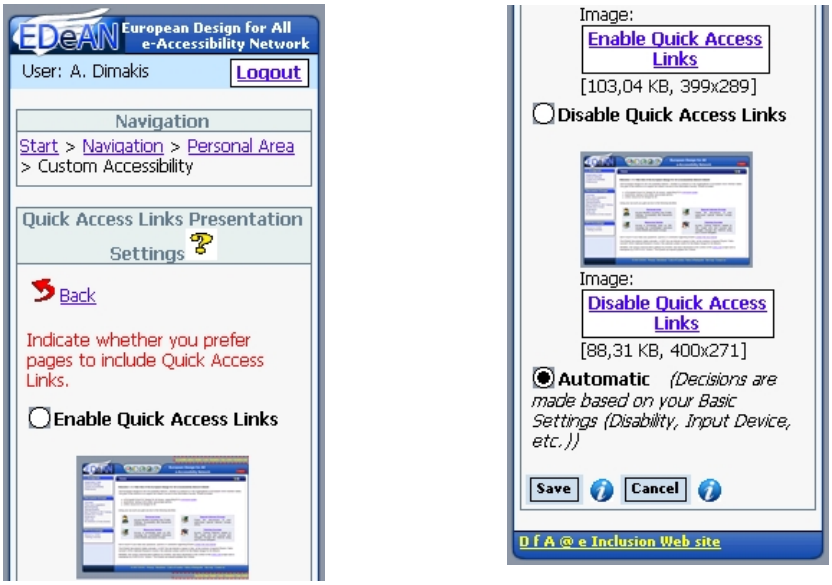


Figure 5.10: Modify Quick Access Links settings in mobile EDeAN.

The preview images of the desktop version (Figure 5.9) exceed the detected screen size of the mobile device. Therefore, they are replaced with an alternative mobile style (Figure 5.10) for displaying images that provides a preview of the original image, information about its size and dimensions and a link to the original.

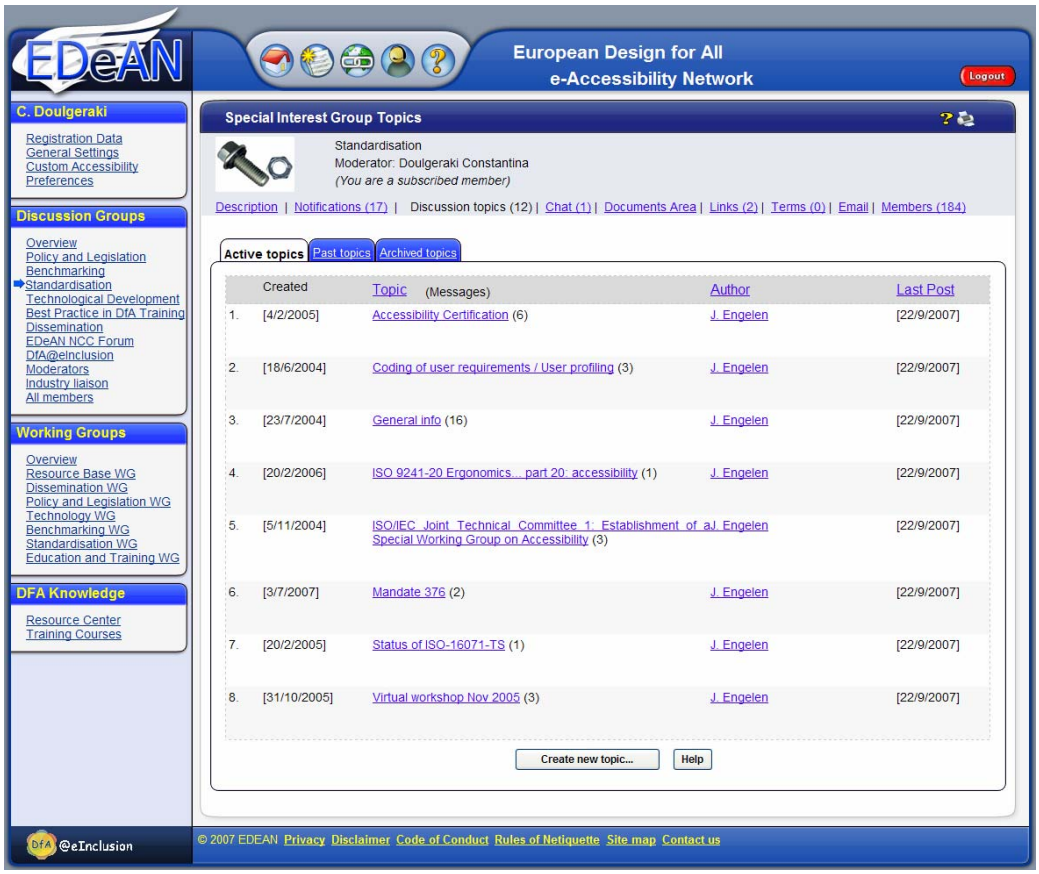


Figure 5.11: Discussion topics of Standardization SIG (Special Interest Group) in desktop EDeAN.

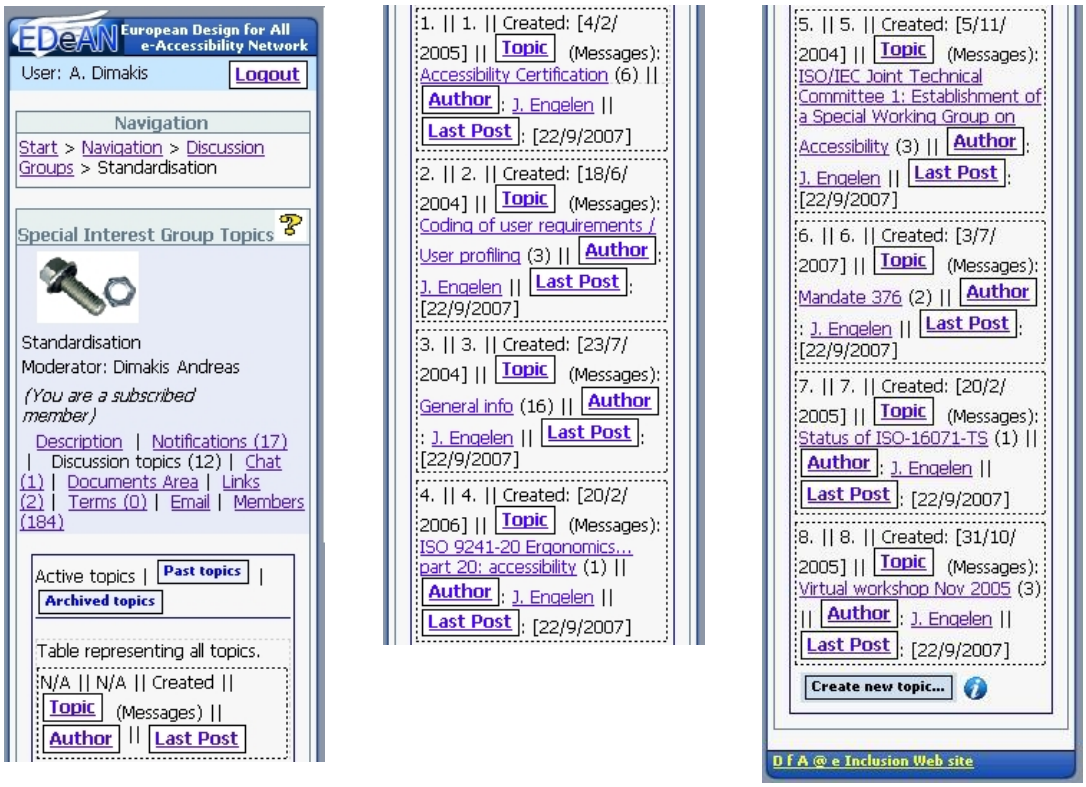


Figure 5.12: Discussion topics of Standardization SIG in mobile EDeAN.

Figure 5.12 shows the alternative mobile style for the displaying of tabs. Moreover, the table appearing in the desktop page (Figure 5.11) is replaced with a linearized style appropriate for the small screen of the PDA. The column names (Topic, Author, Last Post) act as landmarks in the linearized form, while preserving their functional role (sorting of table elements).

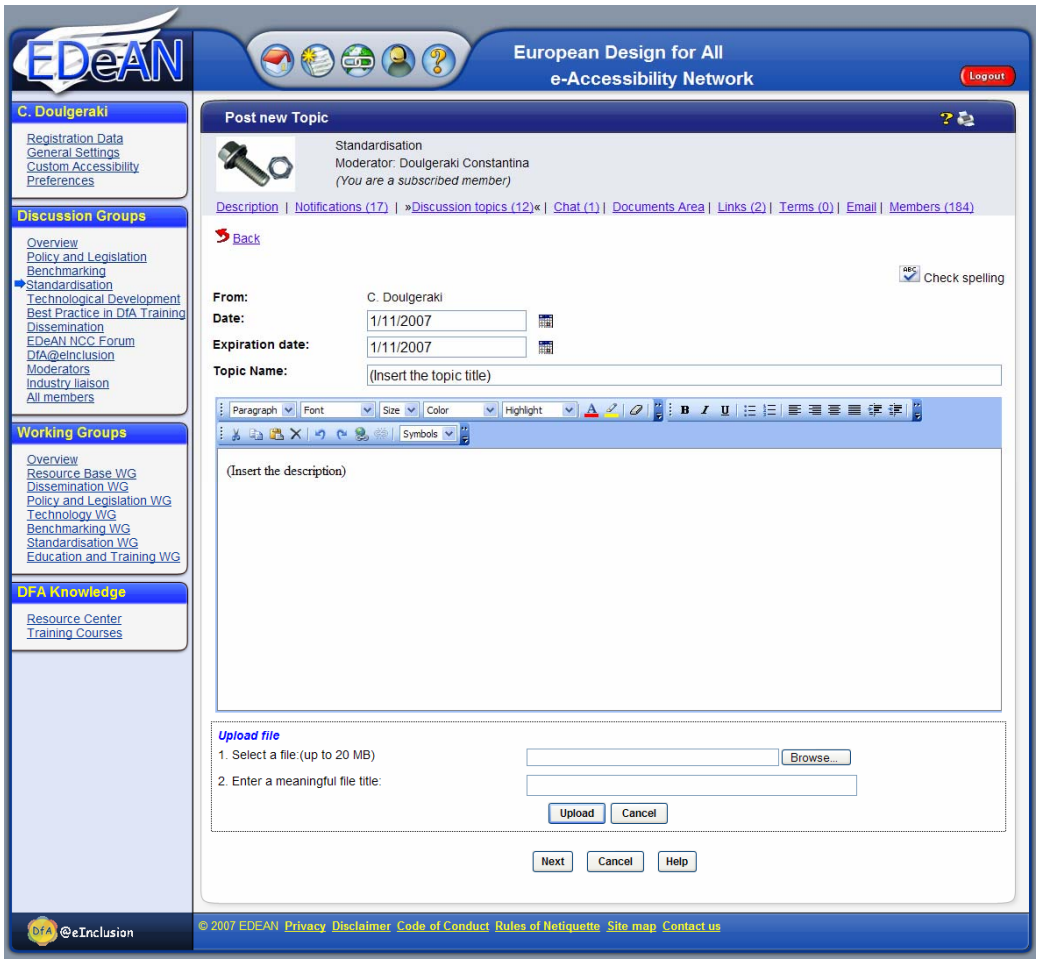


Figure 5.13: Create new topic in Discussion topics of Standardization SIG in desktop EDeAN.

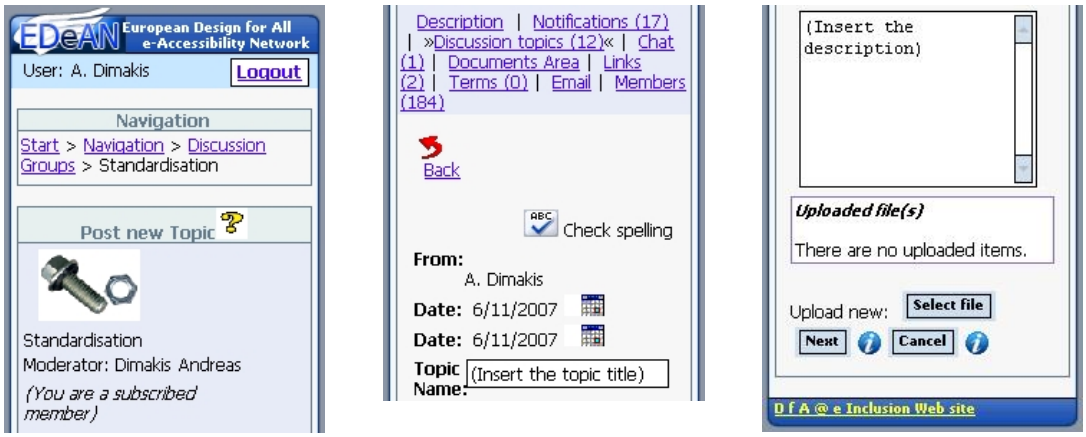


Figure 5.14: Create new topic in Discussion topics of Standardization SIG in mobile EDeAN.

The Create New Topic page (Figure 5.13) contains a series of interactive artifacts that are replaced during the mobile adaptation process with the alternative interactive artifacts of the *UniMob* system. From top to bottom in Figure 5.14, the desktop date picker style is replaced with the equivalent mobile date picker style, the text editor is

replaced with a plain text area style and finally the desktop file uploader style is replaced with a more compact mobile style. The mobile file uploader displays also the alternative style for the displaying of fieldsets.

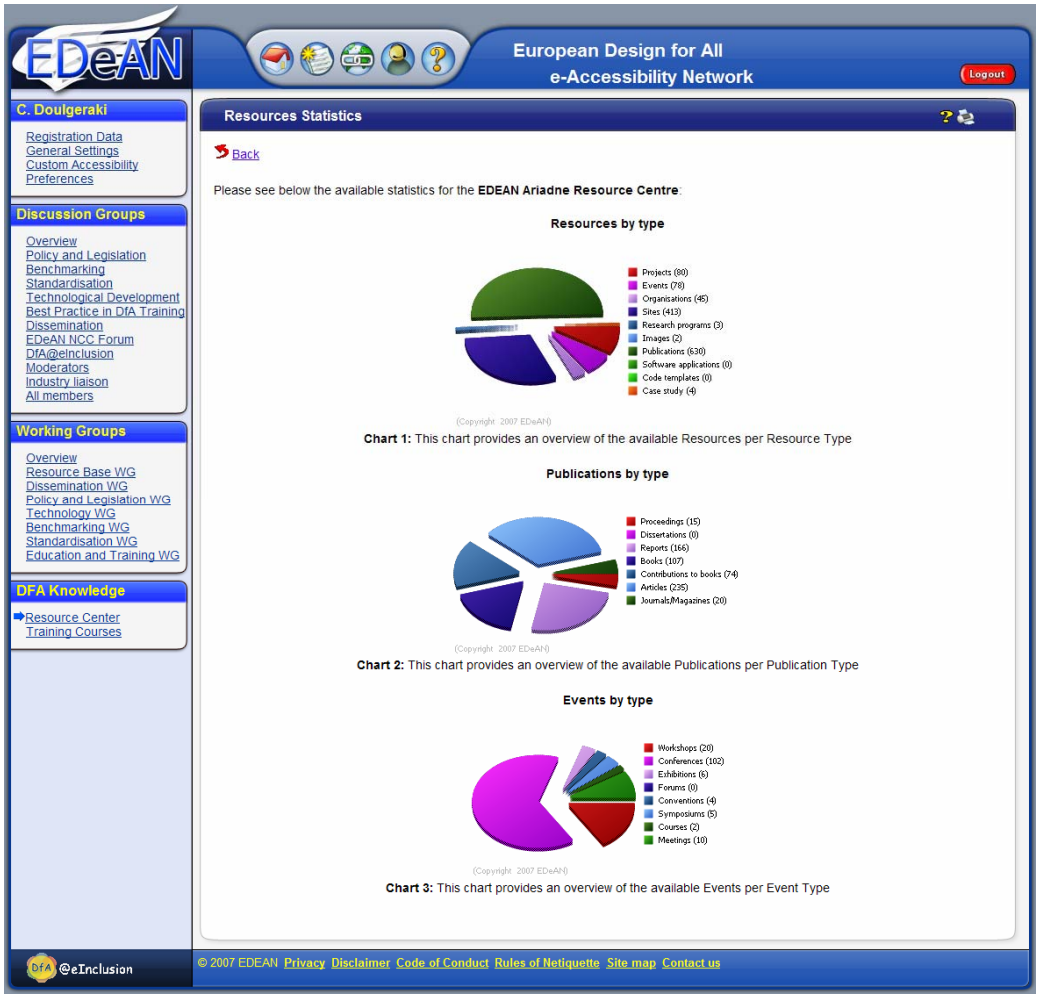


Figure 5.15: Statistics of the Resource Center in desktop EDeAN.

Figure 5.15 shows three pie charts as they are displayed in the desktop page. The alternative mobile version has a resized image for the chart and a legend placed below this image (Figure 5.16). This style utilizes all the available width of the screen and provides an easily readable legend representation.

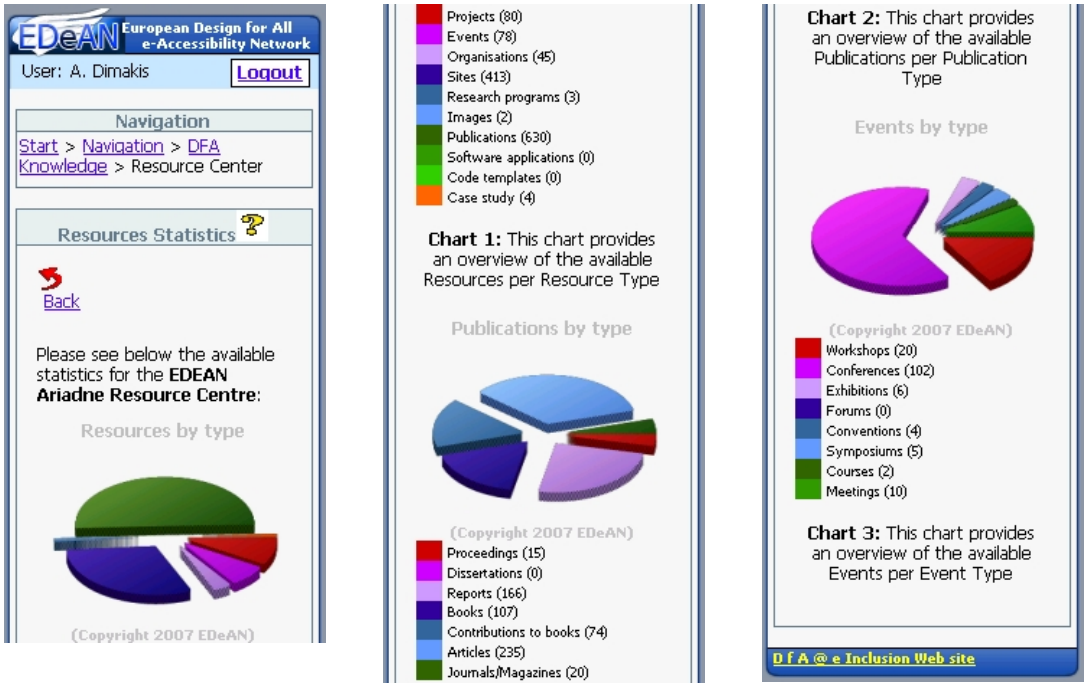


Figure 5.16: Statistics of the Resource Center in mobile EDeAN.

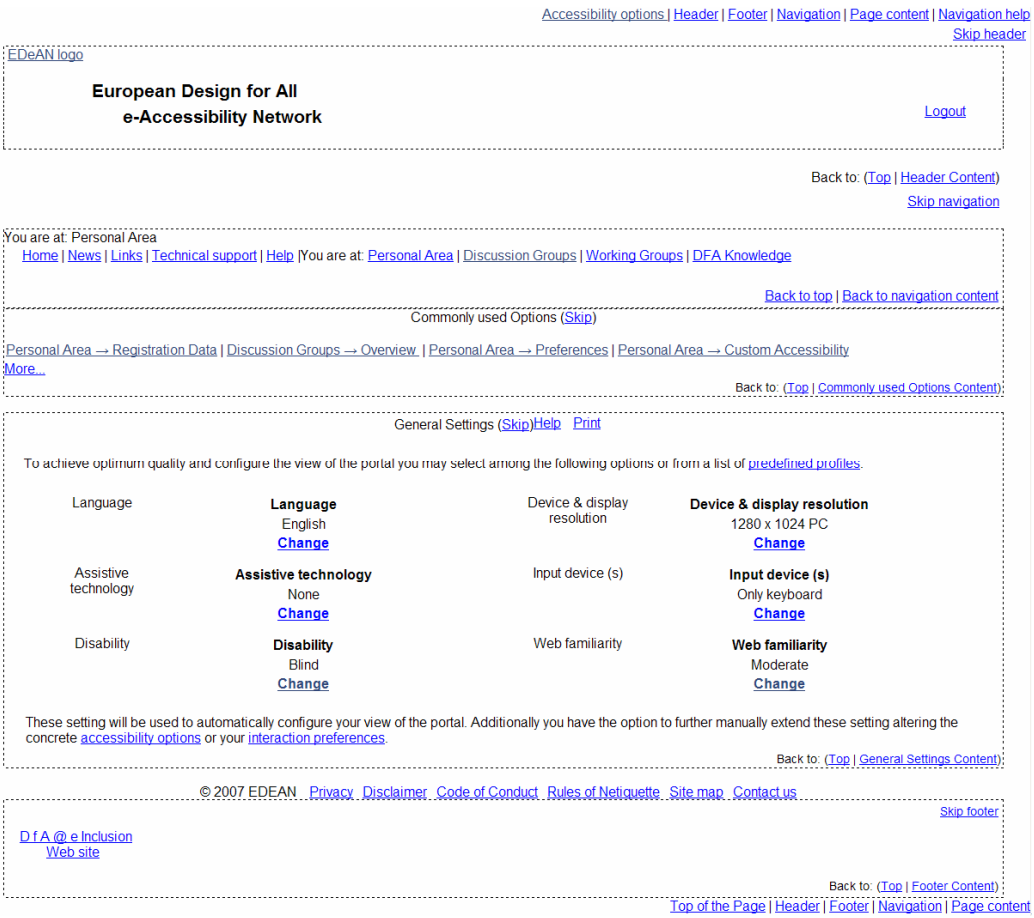


Figure 5.17: General Settings area in desktop EDeAN with disability Blind enabled.

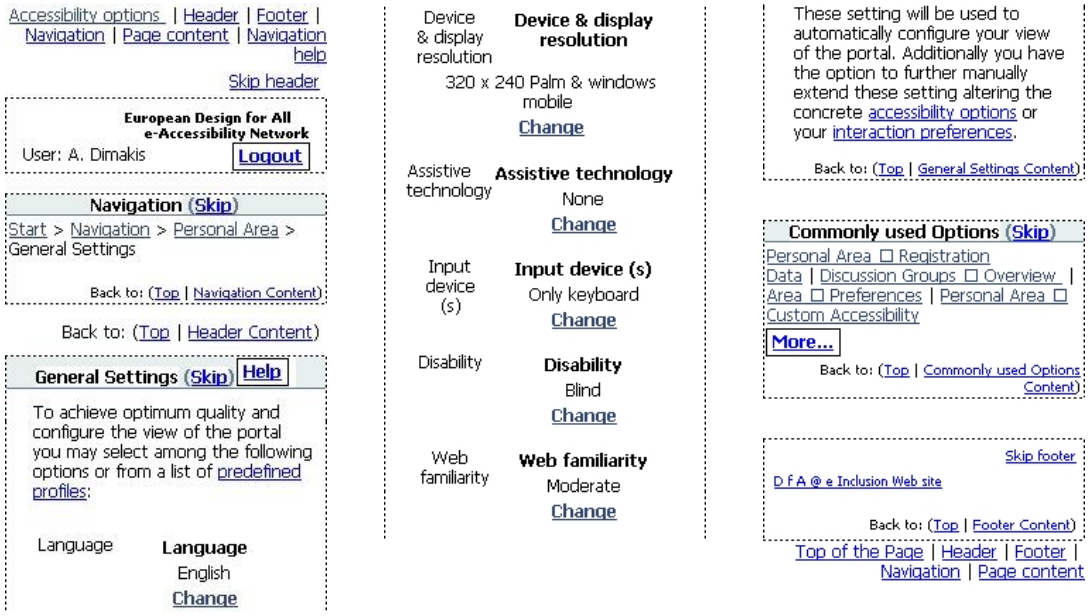


Figure 5.18: General Settings area in mobile EDeAN with disability Blind enabled.

An aim of the proposed adaptation system is to provide mobile adapted interfaces suitable for users with disabilities. Figure 5.18 displays the mobile interface adapted for blind users. It is a textual alternative of the original interface that fits in the dimensions of a mobile device. Also, the Quick Access Links are enabled, the graphical navigation is replaced with a pure textual alternative and section breaks ("skip" and "back to" links) are added before and after each section. A blind user can make use of a screen reader tailored to mobile devices in order to navigate and use this interface.

Figure 5.19: General Settings area in desktop EDeAN with disability Color Blind - Deuteranope enabled.

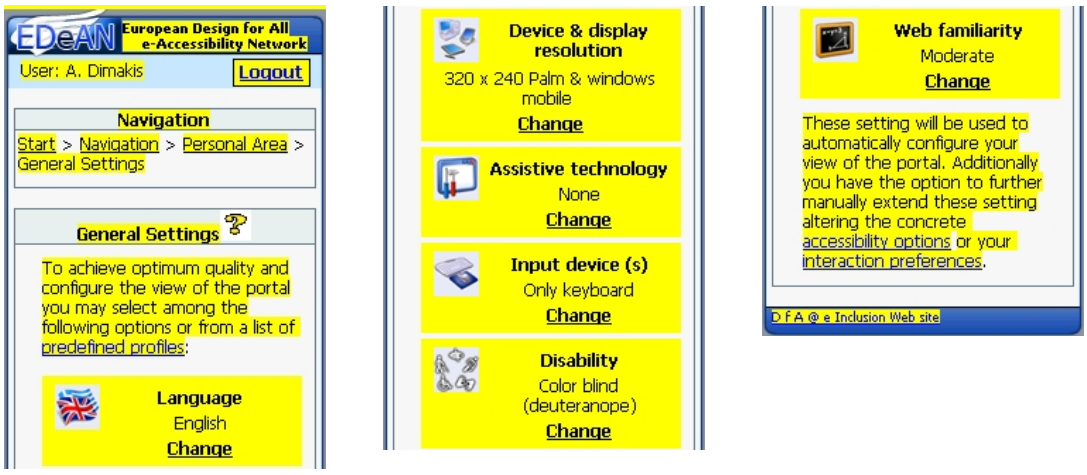


Figure 5.20: General Settings area in mobile EDeAN with disability Color Blind - Deuteranope enabled.

The interface displayed in Figure 5.20 is suitable for Deuteranope color blind users. The textual elements of the page are displayed with contrast suitable for this type of color blindness. This also applies to the other supported types of color blindness (Protanope and Tritanope).

5.3 Discussion

This section presented a case study regarding the unified development of the mobile version of a web portal using the *UniMob* tool. The conducted case study has confirmed the usefulness and fit-for-purpose of the developed tool, leading to a high quality mobile prototype trough graceful adaptation of the desktop version, developed through *EAGER*.

The EDeAN prototype was initially developed in order to support desktop adaptations. After the extension of the portal with *UniMob,* some design decisions that were in favor of the desktop adaptation rise the need for fine-tuning of some areas of the portal. These alterations in the code were implemented in short time and allowed *UniMob's* interface artifacts to be properly displayed on the mobile device.

The conducted case study indicated that:

- The addition of mobile adaptation functionality is feasible even after the development of an only desktop-adaptable portal. In this case minimal changes are required to enhance the presentation of the final interface.
- The best possible results can be achieved if both tools (*EAGER, UniMob*) are available from the start of the development process. The required development effort will be less and there will be no need for additional changes.
- Areas of the portal that are common or will be used in both desktop and mobile versions must be carefully designed and coded in order not to become the one version a burden to the other.
- The EDeAN portal supports advanced functionality. It contains areas that enable users to execute specific tasks and areas that only provide content. Sites that provide only content and have minimal functionality (e.g., online newspapers) will have even better results regarding the quality of their mobile interface.

Overall, this case study has demonstrated that *UniMob* constitutes a powerful extension of *EAGER* for the development of mobile unified web interfaces.

Chapter 6

Evaluation & Results

This Chapter reports the usability evaluation of the *UniMob's* interactive artifacts, and in particular of their instantiation in the EDeAN mobile portal. Additionally, the Chapter discusses the compatibility of the *UniMob's* interactive artifacts and adapted interface with the W3C recommendations about accessibility and mobile web.

6.1 Methodology

The usability evaluation is a necessary part of the development lifecycle of any type of software. It is conducted in several stages of the development lifecycle in order to detect usability problems, and to capture the satisfaction of end users. Through the evaluation process, some primary design decisions are rejected and some others are confirmed. New user requirements and problems rise and can be faced prior to the final release of the product.

Many evaluation methods exist, each one with its own requirements and results of different type. There are no "good" or "bad" evaluation methods, it depends on the situation.

The evaluation of the *UniMob's* interactive artifacts and adapted interface was target to assess consistency and efficiency of the adapted interface, and was divided in two stages. The first one was a **heuristic evaluation** conducted by two experts in the field of interface design. This evaluation phase was executed after the creation of the first functional prototype of the system. The second was a **subjective evaluation** conducted with five end users. This experiment was executed in a fully functional prototype, the EDeAN portal. Sections 6.2 and 6.3 present the details of these two methods and their results.

In both evaluation phases, the mobile device used was the HP iPAQ hx2700 Pocket PC (624 MHz, 64MB, Integrated Wi-Fi). The operating system was the

Microsoft Windows Mobile 5.0 and the browser the Internet Explorer Mobile[8] operating in "Default" display mode, with zoom level "Medium" and Javascript enabled.

6.2 Heuristic Evaluation

Heuristic Evaluation is a Usability Inspection method [93]. It is the less formal method among the eight Usability Inspection methods, and requires a small number of evaluators that examine the interface and judge its compliance with recognized usability principles ("heuristics"). The ten heuristics cover the following areas:

1. Visibility of system status
2. Match between the system and the real world
3. User control and freedom
4. Consistency and standards
5. Help users recognize, diagnose and recover from errors
6. Error prevention
7. Recognition rather than recall
8. Flexibility and efficiency of use
9. Aesthetic and minimalist design
10. Help and documentation

The background of the evaluators is not necessarily the same with the background of the end users. The whole process does not require too much time and it costs less than other methods. Heuristic Evaluation can be conducted by only one evaluator, but optimal results are achieved with more evaluators. The majority of the usability problems are usually common for all evaluators. However, a small number of problems may be found only by some evaluators. Therefore, it is recommended the number of evaluators to be greater than three and less than five [94]. The benefit from using more than five evaluators is not significant. In order to ensure the independence of the evaluators' opinions, each one inspected the interface alone.

Two members of the HCI Laboratory of the FORTH (Foundation for Research and Technology - Hellas) specialized in interface design performed the heuristic

[8] http://www.microsoft.com/windowsmobile/software/iemobile.mspx

evaluation of the *UniMob* system. Prior to the initiation of the process, the evaluators were informed about the procedure that will be followed and the aims of the evaluation, and they were briefly introduced to the functionality and scope of this first prototype. Each evaluator was given a heuristic evaluation checklist. This checklist provided an indicative analysis of the ten heuristics in order for the evaluators to focus on several aspects of each heuristic that were important to check. However, the evaluators were allowed to comment on any encountered usability problem. The checklist is provided in Appendix A. Each evaluator was asked to find any possible usability problem by consulting the heuristics checklist. The results (problems) of the heuristic evaluation are presented next.

- The initially fast-loading designed home page lacks descriptive content, and, more importantly, does not provide direct access to basic functions (login, register, etc.) to the logged out user. The same applies to the home page of the logged in user. Apart from the navigation menu, it does not contain any information about the contents and the role of the site.
- In the pages that contain the preferences of the user (custom accessibility, general settings, interaction preferences, etc.) the areas of the options take up too much space. This results in additional vertical scrolling. A possible solution is to decrease the size of the icons in each area.
- The help icon located near the current area title does not seem to be clickable.
- The “Registration Data” page contains computer jargon (e.g., “Your email is set to “visible to other users”” instead of “Your email is visible to other users”). Novice users could misunderstand this information.
- The navigation path provides access to categories, subcategories and options areas. However, some pages have further navigation in the content. The added “Back” buttons provide backward navigation. It would be preferable for the navigation path to contain also the in content navigation links.
- The contact details of the SIG Moderators are not sufficient. It would be better to provide an email address, telephone or fax.
- When user’s subscription in a SIG is pending for acceptance, a system message appears in red color. Usually, red color is used in errors. Consider using another color (e.g. blue, green). Furthermore, a “Subscribe” button appears while another subscription is pending for acceptance.

- Not all pages have a descriptive title, especially pages with only navigation options. Consider providing a title or a description for the navigation area. By combining the path information with the page title the user can understand where he is and where can he go next.
- When a user enters information through the virtual keyboard and presses the "Enter" key, the related action in the page is not automatically executed. For example, if a user types a keyword in a search function and presses the "Enter" key in the virtual keyboard the focus is not automatically transferred to the "Search" button in the page in order to execute the search.
- When selecting either "Favorites" option or "Navigation Search" option from the start page there is no way to close the selected option. It is possible to select the other option but you can not close it.
- The "Navigation Search" does not return the Top Menu, Navigation, Bottom Menu areas in the results. Moreover, users can misunderstand the meaning of "T", "N" and "B" labels that are used to mark results from the Top Menu, the Navigation and the Bottom Menu and use them as search keywords.
- Most mobile browsers when in full screen mode do not provide the "Back" functionality. Although this is a limitation of the browser it will be helpful to provide a fixed back button inside the web page.

Most of these problems were solved before the release of the final prototype that was used in the Subjective Evaluation with the end users. The rest were either a matter of tradeoff between page size, compactness and usability, or were technically impossible to bypass them.

6.3 Subjective Evaluation

Subjective evaluation aims to provide techniques for collecting, analyzing and measuring the subjective opinion of users when interacting with a software system. This type of evaluation requires a fairly stable prototype, and it tries to capture the satisfaction of end users when using the system. Therefore it does not rely upon expert's opinion. When conducting a subjective usability evaluation, the sample users should be carefully selected in order to be representative of the target user group.

There are several techniques available for subjective evaluation. The IBM Computer Usability Satisfaction Questionnaires [95] constitute the technique selected for use in the evaluation of the EDeAN prototype. These questionnaires measure the users' subjective opinion in a scenario-based situation. Two types of questionnaires are used, the first, namely ASQ (After Scenario Questionnaire), is filled in by each participant at the end of each scenario, while the other one, namely CSUQ (Computer System Usability Questionnaire) is filled in at the end of the evaluation. Both these questionnaires are attached in Appendix B.

The result of the subjective evaluation with the IBM Computer Usability Satisfaction Questionnaires is a set of metrics which can be summarized as follows:

- *ASQ* score, for a participant's satisfaction with the system for a given scenario.
- *OVERALL* metric providing an indication of the overall satisfaction score.
- *SYSUSE* metric providing an indication of the system's usefulness.
- *INFOQUAL* metric providing the score for information quality.
- *INTERQUAL* metric providing the score for interface quality.

End User Selection

Five users where selected to participate to the subjective evaluation of the system. These users where either undergraduate or graduate students of the University of Crete. It was important the end users to have variable web expertise and not to be all of them very experienced with PDA devices. It must be stated that the web expertise level of a user do not always correspond to his mobile web expertise level. This factor was made clearer during the scenario execution process. Although the target user group of the EDeAN portal is mainly scientists and researchers the mobile adaptation mechanism is not targeted to a specific user group. This is the reason why users with moderate web expertise level are included in the evaluation process.

User demographics

The characteristics of the end users were collected with a questionnaire that was filled in prior to the initiation of the evaluation process. This questionnaire contains statistical characteristics (age, gender, education), but also user characteristics related technology use and expertise. Users were asked to provide information regarding how frequently they use the Internet, how do they describe their familiarity with the web and whether they have used a PDA device in the past. These parameters are important

for the analysis of the questionnaires' scores and the user comments. Table 6.1 summarizes the user characteristics collected through this questionnaire.

Table 6.1: End user characteristics.

No	Age	Gender	Education	Internet Usage (frequency)	Internet Usage (expertise)	PDA Usage
1	23	male	higher	3/4 times a week	expert	yes
2	31	male	higher	everyday	expert	no
3	20	male	higher	everyday	moderate	no
4	23	female	higher	1/2 times a week	moderate	yes
5	21	male	higher	everyday	expert	no

Scenario Structure

Two usage scenarios were developed to facilitate the evaluation process. The purpose of the first scenario was to evaluate the "external" navigation facilities of the portal. The "external" navigation refers to the navigation in the portal's pages until the requested content is found. The aim of the second scenario was to evaluate the "internal" navigation of the portal. The "internal" navigation refers to the navigation inside the content of a page. Alternatively, "internal" navigation can be described as the usage of the content facilities in order to execute the requested task.

In the first scenario, the user navigates to a "preferences" area in the portal and alters some pre-selected options. Then the "Customizable table" control is used to show and hide some columns of a table. In the second part of this scenario, the user utilizes two navigation controls (Favorites, Navigation Search) that provide shortcuts in different navigation areas. In the second scenario, the user views the statistics of an area of the portal, executes a search and then navigates to the results. Both scenarios are attached in Appendix B.

Results

The results of the Computer Usability Satisfaction Questionnaires are summarized in Tables 6.2 and 6.3. OVERALL is the average of the responses to items 1 through 19. SYSUSE is the average of the responses to items 1 through 8. INFOQUAL is the average of the responses to items 9 through 15. INTERQUAL is the average of the responses to items 16 through 19. It must be stated that low scores are better than high

scores due to the anchors used in the 7-point scales. If a participant does not answer an item or marks "N/A", this item is omitted from the calculation of the average values.

Table 6.2: Summary of ASQ results.

ASQ Results	**User 1**	**User 2**	**User 3**	**User 4**	**User 5**
Scenario A	1.66	2	3	1.66	2
Scenario B	2	2	2.5	1.33	1.66

Table 6.3: Summary of CSUQ results.

CSUQ Results	**User 1**	**User 2**	**User 3**	**User 4**	**User 5**
OVERALL	1.42	2.1	3.21	1.61	1.57
SYSUSE	1.25	2	3.12	1.57	1.5
INFOQUAL	1.57	2	3.57	1.85	1.85
INTERQUAL	1.5	2.5	2.75	1.25	1.25

From the above results, the overall conclusion is that the subjective opinion of users regarding the mobile interface is very good. The basic pass threshold is 4 and the scores of the most metrics are around 2. This means that users are very satisfied with the interface. The best scores are observed in the interface quality metric (INTERQAL). This could be related to the conservation of the look-and-feel of the desktop site in the mobile version. Also, the minimal graphics of the site and the proper alignment of elements in the page may have contributed to this factor. The weakest scores are observed in the information quality metric (INFOQUAL). In the small screen, only a specific portion of the information available in the desktop version of the site is displayed. The limited content of the mobile interface may create a sense of information lack or content irrelevance. Although the INFOQUAL metric is the weakest among the others, still its value is under the pass threshold.

One important final note is that User 3 (moderate expertise, no previous PDA usage) rated the highest scores among the other users. This means that he faced the greatest difficulties in executing the two scenarios and using the adapted interface. In order to generally validate the positive evidence of this evaluation, a lager evaluation experiment involving a significant number of participants with variable levels of web expertise and mobile device familiarity should be conducted.

6.4 W3C Recommendations Conformance

One of the aims of the proposed adaptation mechanism is the provided interactive artifacts and the final mobile interface to be in conformance with the available recommendations about web content accessibility and mobile web. The first topic is covered by the **Web Content Accessibility Guidelines 1.0**[9] (WCAG) and the second one by the **Mobile Web Best Practices 1.0**[10] (MWBP).

The WCAG 1.0 is a recommendation of the W3C since 1999. It provides a series of guidelines for all web content developers that explain how to make web content accessible to people with disabilities. Following these guidelines also makes web content more available to all users, whatever user agent they are using (e.g., desktop browser, mobile browser, etc.). The compliance with this recommendation was tested with Watchfire Bobby [11]. Bobby is a web accessibility desktop testing tool designed to help expose barriers to accessibility and encourage compliance with existing accessibility guidelines, including Section 508 of the US Rehabilitation Act and the W3C's Web Content Accessibility Guidelines. All interactive artifacts of the *UniMob* library were tested with Bobby for compliance with the WCAG AAA accessibility level.

The MWBP 1.0 was developed by the Mobile Web Best Practices Working Group as part of the Mobile Web Initiative[12]. It is a proposed recommendation that specifies best practices for delivering web content to mobile devices. The best practice guidelines are intended to improve the delivery and display of content to mobile and other portable small screen devices. These practices are not directed to certain types of devices (PDA, mobile phone, etc.) or devices with specific characteristics. However, the “Default Delivery Context” (DDC) (Table 6.4) sets the minimum delivery context specification necessary for a reasonable experience of the web on mobile devices. Many devices exceed the capabilities defined by the DDC specification, and in such cases content providers are encouraged not to diminish the user experience on those devices by developing only to the DDC specification.

[9] http://www.w3.org/TR/WCAG10/
[10] http://www.w3.org/TR/mobile-bp/
[11] http://www.watchfire.com/products/webxm/bobby.aspx
[12] http://www.w3.org/Mobile/

Table 6.4: Default Delivery Context[13].

Usable Screen Width	120 pixels, minimum
Markup Language Support	XHTML Basic 1.1 delivered with content type `application/xhtml+xml`
Character Encoding	UTF-8
Image Format Support	JPEG, GIF 89a
Maximum Total Page Weight	20 kilobytes
Colors	256 Colors, minimum
Style Sheet Support	CSS Level 1. In addition, CSS Level 2 `@media` rule together with the `handheld` and `all` media types.
HTTP	HTTP/1.0 or more recent
Script	No support for client side scripting.

The UniMob Library is targeted to PDA devices. In most cases, the specifications of these devices exceed the DDC specifications. The adapted interface produced by means of the *UniMob* system is targeted to an upgraded delivery context. However, efforts were made for the requirements of the *UniMob's* adapted interface to remain close to the DDC specifications. This means that the scope of the proposed adaptation system is beyond MWBP 1.0. Some practices proposed in MWBP 1.0 were violated in purpose in order to provide an experience designed to take advantage of the extra capabilities of the PDA devices.

UniMob's adapted interface complies with most practices of the MWBP 1.0. The reason for violating some practices is to provide a better user experience. For example, in mobileOK Basic Tests 1.0[14], that check the conformance with the MWBP 1.0, if the size of the document exceeds 10 kb, or 20 kb in case of document that has included resources then this check fails. The average size of a page produced with *UniMob* is 160 kb (6kb template graphics, 10kb scripts, 100kb stylesheets, 20kb content graphics (may vary), 20kb markup (may vary)). Although this is a fairly small page size, convenient for a PDA device, the test fails. A web page of 20kb size with tiny graphics may seem inappropriate for devices with capabilities of a PDA.

6.5 Summary

[13] http://www.w3.org/TR/mobile-bp/#ddc
[14] http://www.w3.org/TR/mobileOK-basic10-tests/

This chapter presented the evaluation conducted in order to measure the level of usability, consistency and efficiency of the adapted interface developed through UniMob. The evaluation highlighted some interesting issues about the usability of the system. The results indicated that the overall user experience and the interface quality are quite satisfying for the end users. The problems that were found were addressed and solved. Most importantly, the evaluation process gave indications for potential improvements, which will guide future research efforts.

Chapter 7

Conclusion & Future Work

7.1 Conclusion

This thesis presented the design and implementation of *UniMob* system. This system provides mobile adaptation capability based on several user and context attributes. The review of currently published literature revealed that, concerning the automated adaptation mechanisms, they provide transparent adaptation for the mobile user, but they lack control of the presentation of the final interface. The non-automated adaptation mechanisms provide mobile interfaces of high quality, but they require additional development and usually offer a subset of the functionality of the desktop site.

A user centered design process was followed in order to produce the interface artifacts of the *UniMob* system. The Unified User Interfaces Design methodology and Architecture were used in the development of the system. A fully functional prototype was built and used as a testbed in the evaluation process. The heuristic evaluation conducted by usability experts revealed some initial usability problems of the interface that were solved. The subjective evaluation conducted by end users indicated the high quality of the mobile interface. All metrics used concerning the interface quality (*INTERQUAL*), the information quality (*INFOQUAL*) and the system's usefulness (*SYSUSE*) scored high in the evaluation with end users.

One of the major objectives of this thesis was to develop a mobile adaptation mechanism providing the capability to control the presentation of the adapted interface. In *UniMob,* this is guaranteed as long as the final interface is built of specific interactive artifacts, whose presentation styles are predefined. The adaptation process is transparent to the end user, and provides graceful transformation of the navigation mechanism and the content of the portal to be adapted. Furthermore, the produced mobile interface preserves all the content and the functionality of the desktop version. All the services of the desktop site are also available in the mobile version.

UniMob's library of adaptable interface artifacts is fully documented and spans a total of 58 alternative presentation styles. These styles cover different contexts of use, including, among others, accessible alternatives for blind and color blind users. This library can be easily expanded with new styles, new parameters for the context of use and with new rules about the selection of styles that build the adapted interface. Such changes and expansions are feasible since the architecture of the system comprises of independent intercommunicating components with well-defined roles and behaviour.

7.2 Future Work

An evaluation of the accessible versions of the mobile interface has been planned as part of future work. Although the system complies with the major standards and rules about web content accessibility, a focused evaluation with specific target user groups will reveal possible usability problems that even usability experts can not easily detect. Moreover, end users could provide useful information that will help the fine tuning of the existing accessible styles or indicate the development of new ones.

The conducted usability evaluation indicated that a greater sample of end users with diverse capabilities and characteristics could potentially improve the existing non accessible styles and lead to the development of new ones. This evaluation will be focused mainly to novice end users and users with no prior experience with PDA devices.

The proposed adaptation mechanism is tailored mainly to PDA devices. More specifically, any device with a browser capable of sending and receiving http requests and displaying "XHTML 1.0 Strict" markup can display the adapted interface. Several PDA devices and emulators were used in the development process. However, the mobile environment includes also mobile phones. The difficulties and challenges of creating adapted interfaces for mobile phones are exceptionally high. Different types of markup languages are used (WML[15], cHTML[16]) with different types of connection. Also, mobile phones have display screens smaller than the PDA devices, and their input methods are far more restricted (usually one four-way button, one select button and one cancel button). New styles need to be designed and implemented in different markup languages. All the architectural components of the

[15] http://www.wapforum.org/what/technical.htm
[16] http://www.w3.org/TR/1998/NOTE-compactHTML-19980209/

UniMob system can be extended to meet the new requirements and the new context of use.

Appendix A: Heuristic Evaluation Checklist

1. Visibility of system status			
Question	**Y**	**N**	**N/A**
1. Does every display begin with a title or header that describes screen contents?			
2. Is there a consistent icon design scheme and stylistic treatment across the system?			
3. Do instructions, prompts and error messages appear consistently throughout the web site?			
4. After the user completes an action (or group of actions), does the feedback indicate that the next group of actions can be started?			
5. Is the naming terminology consistent with the user's domain background?			
6. Does the system provide *visibility*: that is, by looking, can the user tell the state of the system (where s/he is) and the alternatives for action (where can s/he go next)?			
2. Match between the system and the real world			
Question	**Y**	**N**	**N/A**
1. Are graphics concrete and familiar?			
2. If there is a natural sequence to menu choices, has it been used?			
3. Do the selected colors correspond to common expectations about color codes?			
4. On data entry screens, are tasks described in terminology familiar to users?			
5. Is the use of metaphors easily understandable by the user?			
3. User control and freedom			
Question	**Y**	**N**	**N/A**
1. Is it easy for users to switch between tasks?			
2. Can users easily reverse their actions, where appropriate?			
4. Consistency and standards			
Question	**Y**	**N**	**N/A**
1. Does each page have a title?			
2. Do choices (links) appear consistently (location & color)?			
3. Are there no more than four to seven colors, and are they far apart along the visible spectrum?			
4. Have pairing of high chroma (color vividness), spectrally extreme colors been avoided?			
5. Is the most important information placed at the beginning of each page?			
6. Are user actions named consistently across all pages in the site?			
5. Help users recognize, diagnose and recover from errors			
Question	**Y**	**N**	**N/A**
1. Are prompts stated constructively, without overt or implied criticism of the user?			
2. Are prompts brief and unambiguous?			
3. Are error messages grammatically correct?			
4. Do all error messages in the system use consistent grammatical style, form, terminology and abbreviations?			

5. Do error messages suggest the cause of the problem?			
6. Error prevention			
Question	**Y**	**N**	**N/A**
1. Are choices logical and distinctive?			
2. Does the system prevent users from making errors whenever possible?			
3. Do fields in data entry screens and dialogue boxes contain default values when appropriate?			
7. Recognition rather than recall			
Question	**Y**	**N**	**N/A**
1. Are prompts, cues and messages placed where the eye is likely to be looking on the screen?			
2. Are currently inactive elements grayed out or omitted?			
3. Are field labels close to fields, but separated by at least one space?			
4. Are borders used to identify meaningful groups?			
5. Has the same color been used to group related elements?			
6. Does the system provide *mapping*: that is, are the relationships between controls and actions apparent to the user?			
8. Flexibility and efficiency of use			
Question	**Y**	**N**	**N/A**
1. Does the system provide function keys (e.g. shortcuts) for high-frequency commands?			
2. Can a visitor receive a minimum of information about the portal without going through the registration procedure?			
9. Aesthetic and minimalist design			
Question	**Y**	**N**	**N/A**
1. Are all graphic icons in a set visually and conceptually distinct?			
2. Are meaningful groups of items separated by white space?			
3. Does each page have a short, simple, clear, distinctive title?			
4. Is information broken into meaningful chunks?			
10. Help and documentation			
Question	**Y**	**N**	**N/A**
1. Are on-line instructions visually distinct?			
2. If menu choices are ambiguous, does the system provide additional explanatory information when an item is selected?			
3. Is the help function visible?			
4. Is there context-sensitive help?			
5. Is the help information descriptive? (What is this thing for?)			
6. Can the user change the level of detail available?			
7. Can users easily switch between help and their work?			

Appendix B: Usability Evaluation Scenarios and IBM Questionnaires

Scenarios

A. Navigation and usage of "Customizable Table" Scenario
Part (a) 1. From the portal's start page select "Navigation". 2. Select "Personal Area". 3. Select "Custom Accessibility". 4. Select "Change" from the "Display Links as Buttons" area. 5. Choose "Automatic". 6. Select "Save". 7. Select "Change" from the "Quick Access Links" area. 8. Enable Quick Access Links. 9. Select "Save". 10. Return to the portal's start page. 11. Select "Top Menu". 12. Select "News". 13. View all the available tables' columns. 14. Select to display the "Description" column. 15. Select "hide". Part (b) From the portal's start page use either "Favorites" or "Navigation Search" in order to replace steps 1 to 3 in Part (a). It is not necessary to execute the rest steps (4-15).

B. Search and View Field Statistics Scenario
1. From the portal's start page select "Navigation". 2. Select "DFA Knowledge". 3. Select "Resource Center". 4. View the Resource Center's "Statistics". 5. Return to previous page. 6. Select "Search". 7. Insert keyword "dfa" (dfa: design for all). 8. Perform the search.

Questionnaires

The After-Scenario Questionnaire (ASQ)
For each of the statements below, circle the rating of your choice. For items that are not applicable use: N/A.

1. Overall, I am satisfied with the ease of completing the tasks in the scenario.

Strongly agree ←——————————→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

2. Overall, I am satisfied with the amount of time it took to complete the tasks in this scenario.

Strongly agree ←——————————→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

3. Overall, I am satisfied with the support information (messages, documentation) when completing the tasks.

Strongly agree ←——————————→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

The Computer System Usability Questionnaire (CSUQ)

For each of the statements below, circle the rating of your choice. For items that are not applicable use: N/A.

1. Overall, I am satisfied with how easy it is to use this system.

Strongly agree ←→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

2. It is simple to use this system.

Strongly agree ←→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

3. I can effectively complete my work using this system.

Strongly agree ←→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

4. I am able to complete my work quickly using this system.

Strongly agree ←→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

5. I am able to efficiently complete my work using this system.

Strongly agree ←→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

6. I feel comfortable using this system.

Strongly agree ←——————————→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

7. It was easy to learn to use this system.

Strongly agree ←——————————→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

8. I believe I became productive quickly using this system.

Strongly agree ←——————————→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

9. The system gives error messages that clearly tell me how to fix problems.

Strongly agree ←——————————→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

10. Whenever I make a mistake using the system, I recover easily and quickly.

Strongly agree ←——————————→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

11. The information (such as on-screen message, feedback and other documentation) provided with this system is clear.

Strongly agree ←——————————→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

12. It is easy to find the information I need.

Strongly agree ←——————————→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

13. The information provided with the system is easy to understand.

Strongly agree ←——————————→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

14. The information is effective in helping me complete my work.

Strongly agree ←——————————→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

15. The organization of information on the system screen is clear.

Strongly agree ←——————————→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

16. The interface of this system is pleasant.

Strongly agree ←——————————→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

17. I like using the interface of this system.

Strongly agree ←——————————→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

18. This system has all the functions and capabilities I expect it to have.

Strongly agree ←——————————→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

19. Overall, I am satisfied with this system.

Strongly agree ←——————————→ Strongly disagree N/A

1 2 3 4 5 6 7

Comments:

.....

.....

Bibliography

1. Y. Chen, W.Ma and H.Zhang, "Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices", Proc. WWW-12 Conference, 2003, Hawaii
2. Stephanidis C, Savidis A, Akoumianakis D (1997) "Unified user interface development: tools for constructing accessible and usable user interfaces". Tutorial No.13 of the 7th International Conference on Human-Computer Interaction (HCI International '97), San Francisco, CA, 24–29 August 1997
3. Kofod-Petersen, A. & Aamodt, A. (2003). "Case-based situation assessment in a mobile context-aware system". Proceedings of Artificial Intelligence for Mobile Systems (AIMS 2003), Seattle, USA, October, 2003.
4. Stephanidis C., Savidis A., Towards Multimedia Interfaces for All: a new Generation of Tools Supporting Integration of Design-time and Run-time Adaptivity Methods, Workshop "Adaptive Multimedia for People with Disabilities", ACM Multimedia '95.
5. http://my.yahoo.com
6. http://www.google.com/ig
7. http://excite.com
8. Lieberman H. "Letizia : An Agent That Assists Web Browsing". In: Proceedings of the International Joint Conference on Artificial Intelligence, Montreal, CA, 1995
9. Joachims T, Freitag D, Mitchell T. "WebWatcher: A Tour Guide for the World Wide Web". In Proceedings of the International Joint Conference on Artificial Intelligence, 1997
10. Wexelblat A, Maes P. "Footprints: History-rich Web browsing". In: Proceedings of the Conference on Computer-Assisted Information Retrieval (RIAO), 1997, pp. 75-84
11. Stephanidis C, Paramythis A, Karagiannidis C, Savidis A. "Supporting Interface Adaptation: the AVANTI Web-Browser"
12. http://cocoon.apache.org/
13. Claudia Canali, Valeria Cardellini, Riccardo Lancellotti. Squid-based proxy server for content adaptation. Technical Report, Dept. of Computer Engineering, University of Roma, Jan 2003.
14. Freire, J., Kumar, B., and Lieuwen, D. "WebViews: Accessing Personalized Web Content and Services", in Proceedings of WWW10 (Hong Kong, May 2001), ACM Press, pp. 576-586.
15. Steinberg, J. and Pasquale, J. "A Web Middleware Architecture for Dynamic Customization of Content for Wireless Clients", in Proceedings of WWW 2002 (Honolulu, May 7-11, 2002), ACM Press, pp, 639-650.

16. Liljeberg, M., Alanko, T., Kojo, M., Laamanen, H. and Raatikainen, K. "Optimizing World-Wide Web for Weakly Connected Mobile Workstations: An Indirect Approach", in Proceedings of SDNE '95 (Whistler, June 1995), pp. 132-139.
17. Loon, T.S and Bharghavan, V. "Alleivating the Latecy and Bandwidth Problems in WWW Browsing", in Proceedings of the USENIX symposium on Internet technology (Monterey, 1997).
18. Bickmore, T.W. and Schilit, B.N. "Digestor: Device-independent Access to the World Wide Web", in Proceedings of WWW6, Santa Clara, 1997.
19. Fox, A., Gribble, S.D., Chawathe, Y. and Brewer, E.A. "Adapting to Network and Client Variation Using Infrastructural Proxies: Lessons and Perspectives", IEEE Personal Communications, Special Issue on Adaptation, vol. 5, issue 4 (August 1998).
20. James J. Kistler and Mahadev Satyanarayanan. Disconnected operation in the Coda file system. ACM Transactions on Computer Systems, 10(1):3–25, February 1992.
21. Mark Yarvis, Peter Reiher, and Gerald Popek. Conductor: A framework for distributed adaptation. In Proceedings of the Seventh Workshop on Hot Topics in Operating Systems (HotOS), Rio Rico, Arizona, March 1999.
22. Armando Fox, Steven D. Gribble, Eric A. Brewer, and Elan Amir. Adapting to network and client variability via on-demand dynamic distillation. SIGPLAN Notices, 31(9):160–170, September 1996.
23. Mahadev Satyanarayanan, J. Flinn, and K. R. Walker. Visual proxy: Exploiting OS customizations without application source code. Operating Systems Review, 33(3):14–18, July 1999.
24. L.B. Mummert, M.R. Ebling, and Mahadev Satyanarayanan. Exploiting weak connectivity for mobile file access. In Proceedings of the 15th ACM Symposium on Operating Systems Principles, Copper Mountain, Colorado, December 1995.
25. Dushyanth Narayanan, Jason Flinn, and Mahadev Satyanarayanan. Using history to improve mobile application adaptation. In Proceedings of the 3rd IEEEWorkshop on Mobile Computing Systems and Applications, Monterey, California, December 2000.
26. Armando Fox, Steven D. Gribble, Yatin Chawathe, and Eric A. Brewer. Adapting to network and client variation using infrastructural proxies: Lessons and perspectives. IEEE Personal Communications, 5(4):10–19, 1998.
27. Anthony D. Joseph, Joshua A. Tauber, and M. Frans Kaashoek. Building reliable mobile-aware applications using the Rover toolkit. In Proceedings of the 2nd ACM International Conference on Mobile Computing and Networking (MobiCom '96), Rye, New York, November 1996.
28. Lance Bloom, Rachel Eardley, Erik Geelhoed, Meera Manahan, and Parthasarathy Ranganathan. Investigating the relationship between battery life and user acceptance of

dynamic, energy-aware interfaces on handhelds. In Stephen A. Brewster and Mark D. Dunlop, editors, Mobile HCI, volume 3160 of Lecture Notes in Computer Science, pages 13–24. Springer, 2004.

29. Gregory F. Welch. A survey of power management techniques in mobile computing operating systems. Operating Systems Review, 29(4):47–56, 1995.
30. Lawrence S. Brakmo, Deborah A. Wallach, and Marc A. Viredaz. Sleep: a technique for reducing energy consumption in handheld devices. In MobiSys ’04: Proceedings of the 2nd international conference on Mobile systems, applications, and services, pages 12–22, New York, NY, USA, 2004. ACM Press.
31. Lin Zhong and Niraj K. Jha. Energy efficiency of handheld computer interfaces: limits, characterization and practice. In MobiSys ’05: Proceedings of the 3rd international conference on Mobile systems, applications, and services, pages 247–260, New York, NY, USA, 2005. ACM Press.
32. L S. Brakmo, HP Labs; D. A. Wallach, "uSleep: A Technique for Reducing Energy Consumption in Handheld Devices," in the Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services (MobiSys), Boston, MA, 2004
33. Steve Jones, Matt Jones, Gary Marsden, Dynal Patel, Andy Cockburn: An evaluation of integrated zooming and scrolling on small screens. International Journal of Man-Machine Studies 63(3): 271-303 (2005)
34. Bill N. Schilit, Jonathan Trevor, David M. Hilbert, and Tzu Khiau Koh. Web interaction using very small internet devices. IEEE Computer, 35(10):37–45, 2002.
35. I. Cooper and R. Shufflebotham. PDA Web Browsers: Implementation Issues (University of Kent at Canterbury Computing Laboratory WWW Page, November 1995).
36. H. Lie and Bert Bos. Cascading Style Sheets. WWW Consortium, (September 1996).
37. http://edition.cnn.com/mobile/avantgo/
38. http://cnnmobile.com/services/
39. www.bbc.co.uk/mobile
40. Google palm. http://www.google.com/palm/
41. E. Kirda., C. Kerer and G. Matzka, “Using XML/XSL to build adaptable database interfaces for Web site content management”. XML in Software EngineeringWorkshop, 23rd International Conference on Software Engineering, Toronto, Ontario, Canada 2001
42. Unwired Planet, Inc. UP.Link Developer's Guide Version 1.0. Redwood Shores, California, July 1996
43. M. Hori, R. Mohan, H. Maruyama and S. Singhal, "Annotation of Web Content for Transcoding". E3C Note. http://www.w3.org/TR/annot.

44. Kinshuk & Goh T. T. (2003). Mobile adaptation with multiple representation approach as educational pedagogy. Wirtschaftsinformatik 2003 - Medien - Märkte – Mobilität. Heidelberg, Germany: Physica-Verlag, 747-763.
45. Monash, R., Smith, J.R. & Li, C.-S. (1999). Adapting multimedia internet content for universal access. IEEE Transactions on Multimedia, **1**, 1, 104-114.
46. Wireless Markup Language, 2002. WML specification, http://xml.coverpages.org/wap-wml.html.
47. Handheld Device Markup Language, 2002. HDML specification, http://www.w3.org/TR/NOTE-Submission-HDML-spec.html
48. Kam98 Kamada, T., Compact HTML for Small Information Appliances. World Wide Web Consortium, 1998. http://www.w3.org/TR/1998/ NOTE-compactHTML-19980209/
49. George Buchanan, Sarah Farrant, Matt Jones, Harold Thimbleby, Gary Marsden, and Michael Pazzani. Improving mobile internet usability. In WWW ’01: Proceedings of the 10th international conference on World Wide Web, pages 673–680, New York, NY, USA, 2001. ACM Press.
50. Hwang, Y., Kim, J. & Seo, E. (2003). Structure-aware web transcoding for mobile devices. IEEE Internet Computing, September-October. Los Alamitos: IEEE Computer Society, 14-21.
51. AvantGo. http://www.avantgo.com/
52. The Apache Cocoon Project. http://cocoon.apache.org/
53. WebSphere Transcoding Publisher. http://www-306.ibm.com/software/pervasive/transcoding_publisher/
54. M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu. System Demonstration - Strudel: A Web-site Managemnet System. In ACM SIGMOD Conference on Management of Data, 1997.
55. Pocket IE. http://www.microsoft.com/windowsmobile/
56. Opera. http://www.opera.com/
57. NetFront™ Browser v3.5. http://www.access-company.com/products/netfrontmobile/browser/index.html
58. The Web adapted to every screen size: Opera solves the problem of rendering web content on medium-sized screens. Opera press release. Jan 20, 2004. http://www.opera.com/pressreleases/en/2004/01/20/
59. Virpi Roto, Browsing on Mobile Phones, Nokia Research Center.
60. A. Berger and V. Mittal. OCELOT: A System for Summarizing Web Pages. Proceedings of the 23rd Annual Conference on Research and Development in Information Retrieval (ACM SIGIR), pp. 144-151, 2000.

61. Beard, D. B. and Walker, J. Q. 1990. Navigational Techniques to Improve the Display of Large Two-Dimensional Spaces. Behaviour and Information Technology, 9, 6, 451-466.
62. Plaisant, C., Carr, D., and Shneiderman, B. 1995. Image browsers: Taxonomy, guidelines, and informal specifications. IEEE Software, 12, 2, 21-32.
63. Shneiderman, B. 1998. Designing the User Interface. Reading, MA: Addison-Wesley.
64. P. Baudisch, X. Xie, C. Wang, and W.-Y. Ma. Collapse-to-zoom: viewing web pages on small screen devices by interactively removing irrelevant content. In UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology, pages 91{94, 2004.
65. S. Bjork, L. E. Holmquist, J. Redstrom, I. Bretan, R. Danielsson, J. Karlgren, and K. Franzn. West: a web browser for small terminals. In UIST '99:Proceedings of the 12th annual ACM symposium on User interface software and technology, pages 187{196, 1999.
66. Y. Chen, W.-Y. Ma, and H.-J. Zhang. Detecting web page structure for adaptive viewing on small form factor devices. In WWW '03: Proceedings of the twelfth international conference on World Wide Web, pages 225{233, 2003.
67. N. Milic-Frayling and R. Sommerer. Smartview: Enhanced document viewer for mobile devices. Microsoft Technique Report, MSR-TR-2002-114, November 2002.
68. T.W. Bickmore and B.N. Schilit, “Digestor: Device-Independent Access to the World-Wide-Web”, Proc. of 6th World-Wide-Web Conf. WWW’6 (Santa Clara, 7-11 April 1997), Elsevier Science Pub., Amsterdam, 1997.
69. O. Buyukkokten, H. Garcia-Molina, A. Paepcke, and T. Winograd. Power browser: efficient web browsing for PDAs. In CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems, pages 430{437, 2000.
70. J. O. Wobbrock, J. Forlizzi, S. E. Hudson, and B. A. Myers. Webthumb: interaction techniques for small-screen browsers. In UIST '02: Proceedings of the 15th annual ACM symposium on User interface software and technology, pages 205{208, 2002.
71. Xiangye Xiao, Qiong Luo, Dan Hong, Hongbo Fu, Slicing*-Tree Based Web Page Transformation for Small Displays. ACM Fourteenth Conference on Information and Knowledge Management (CIKM 2005), Bremen, Germany, 2005.
72. Furnas, G.W., Generalized Fisheye Views. In: Proc. of ACM SIGCHI '86, pp. 16-23, ACM Press, New York, 1986.
73. Sarkar, M., and Brown, M., Graphical Fisheye Views of Graphs, Proc. ACM CHI 1992, 83-91.
74. Robertson, G., and Mackinlay, J., The Document Lens, Proc. ACM UIST 1993, 101-108.

75. Rao, R., and Card, S., The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus+Context Visualization for Tabular Information, Proc. ACM CHI 1994, 318-322.
76. J. Hsu, W. Johnston, and J. McCarthy. Active Outlining for HTML Documents: An X-Mosaic Implementation. Second International World Wide Web Conference (Chicago, October 1994).
77. T. Kamba, S. Elson, T. Harpold, T. Stamper, and P. Sukaviriya. Using small screen space more efficiently. CHI 96 (Vancouver, BC Canada, April 1996).
78. E. Bier, M. Stone, K. Pier, W. Buxton, T. DeRose. Toolglass and Magic Lenses: The See-through Interface. SIGGRAPH '93 Conference Proceedings (1993).
79. Schaffer, D., Zuo, Z., Greenberg, S., Bartram, L., Dill, J., Dubs, S., and Roseman, M., Navigating Clustered Networks through Fisheye and Full-Zoom Methods, ACM ToCHI, 3,2, 1996, 162-188.
80. Bederson, B., Fisheye Menu Selection, Proc. ACM UIST 2000, 217-225.
81. Gutwin, C., and Skopik, A., Fisheye Views are Good for Large Steering Tasks, Proc. ACM CHI 2003.
82. Gutwin, C., Improving Focus Targeting in Interactive Fisheye Views, Proc. ACM CHI 2002, 267-274.
83. Hornbæk, K.; Bederson, B.; Plaisant, C., “Navigation Patterns and Usability of Overview+detail and Zoomable User Interfaces for Maps”, ACM Transactions on Human-Computer Interaction, Volume 9, Issue 4, p.362-389 (2002)
84. P. Baudisch, B. Lee, and L. Hanna. Fishnet, a Fisheye web browser with search term popouts: a comparative evaluation with overview and linear view. In AVI '04: Proceedings of the working conference on Advanced visual interfaces, pages 133{140, New York, NY, USA, 2004.
85. Benderson, B., Hollan, J., “Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics”, UIST'94, pp. 17-26, 1994
86. Oscar de Bruijn, Robert Spence, Min Yih Chong, “RSVP Browser: Web Browsing on Small Screen Devices”, Personal and Ubiquitous Computing, vol.6, issue 4, pp 245-252, 2002.
87. Potter, M.C. “Rapid serial visual presentation (RSVP): A method for studying language processing.” In Kieras, D.E. and Just, M.A. (eds.), New Methods in Reading Comprehension Research. Lawrence Erlbaum, Hillsdale, NJ, 1984, 91-118.
88. Patrick Baudisch, Xing Xie, Chong Wang, Wei-Ying Ma, Collapse-to-Zoom: Viewing Web Pages on Small Screen Devices by Interactively Removing Irrelevant Content, 17th Annual ACM Symposium on User Interface Software and Technology (UIST 2004), TechNote, Sante

89. Doulgeraki, C., Partarakis, N., Mourouzis, A., Antona, M., Stephanidis, C., (2007). Towards unified Web-based User Interfaces. FORTH-ICS Technical Report, TR-394. Retrieved from: http://www.ics.forth.gr/ftp/tech-reports/2007/2007.TR394_Towards_Unified_Web-based_UI.pdf
90. Hoare, C.A.R. (1978). Communicating Sequential Processes, Communications of the ACM, 21 (8), 666-667.
91. Mowbray, T.J., Zahavi, R. (1995). The Essential CORBA: Systems Integration Using Distributed Objects. New York: John Wiley & Sons.
92. Savidis, A., Antona, M., Stephanidis, C. (2005). A Decision-Making Specification Language for Verifiable User-Interface Adaptation Logic. International Journal of Software Engineering and Knowledge Engineering. Vol. 15, No. 6 (December 2005), World Scientific Publishing, pp 1063-1095.
93. Jakob Nielsen, Robert L. Mack, Usability inspection methods, John Wiley & Sons, Inc, New York, NY, 1994
94. Nielsen, Jakob, and Landauer, Thomas K.: "A mathematical model of the finding of usability problems," Proceedings of ACM INTERCHI'93 Conference (Amsterdam, The Netherlands, 24-29 April 1993), pp. 206-213.
95. IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use / Lewis, James R. International Journal of Human-Computer Interaction 1995 v.7 n.1 p.57-78