

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Προσβάσιμα διαδικτυακά εργαλεία για την κατασκευή
προσβάσιμων και προσαρμόσιμων φορμών αλληλεπίδρασης

Accessible online tools supporting the development of
accessible and adaptable web forms

Γεώργιος Καρτάκης

Μεταπτυχιακή Εργασία

Ηράκλειο, Οκτώβριος 2007

Πανεπιστήμιο Κρήτης
Σχολή Θετικών Επιστημών
Τμήμα Επιστήμης Υπολογιστών

**Προσβάσιμα διαδικτυακά εργαλεία για την κατασκευή προσβάσιμων
και προσαρμόσιμων φορμών αλληλεπίδρασης**

Εργασία που υποβλήθηκε από τον
Γεώργιος Χρ. Καρτάκης

ως μερική εκπλήρωση των απαιτήσεων για την απόκτηση
ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ

Συγγραφέας:

Γεώργιος Χρ. Καρτάκης
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Εισηγητική Επιτροπή:

Κωνσταντίνος Στεφανίδης
Καθηγητής, Επόπτης

Γρηγόρης Αντωνίου
Καθηγητής, Μέλος

Αντώνης Σαββίδης
Αναπληρωτής Καθηγητής, Μέλος

Δεκτή:

Παναγιώτης Τραχανιάς
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Ηράκλειο, Οκτώβριος 2007

Acknowledgements

I am grateful to my supervisor Professor Constantine Stephanidis for his assistance, encouragement and support.

For their patience and thoroughness in reading my thesis, and an unexpectedly enjoyable viva, I am indebted to Professor Grigoris Antoniou and Professor Anthony Savidis.

I would also like to thank the Department of Computer Science of the University of Crete and the Institute of Computer Science (Human Computer Interaction Laboratory) of the Foundation for Research and Technology – Hellas (FORTH) for providing me with the means for setting and achieving my research goals.

I would like to thank Mr. Socrates Kartakis and Miss Anna-Maria Gertsou, for their support during the evaluation stage of this thesis.

For her inspiration and support, I would like to thank in particular Dr. Margherita Antona.

Finally, I would like to thank Mr. Ioannis Basdekis for helping me overcome emerging difficulties and problems by providing his valuable scientific insight, inspiration and support.

Το master αφιέρωσα στην οικογένειά μου,
που 'ναι το μόνο στήριγμα σ'όλα τα βήματά μου.

Στο Χρήστο
Στη Ράνια
Στο Σωκράτη
Στον Άγγελο

Περίληψη

Ο Παγκόσμιος Ιστός γίνεται ολοένα και πιο σημαντικό εργαλείο για τις καθημερινές μας δραστηριότητες. Ταυτόχρονα, οι τεχνολογικές πλατφόρμες και συσκευές πρόσβασης στο διαδίκτυο πληθαίνουν και γίνονται όλο και πιο σύνθετες (π.χ., PDAs, φορητές συσκευές, iTV, κλπ). Επομένως, είναι εξαιρετικά σημαντικό οι εφαρμογές και οι υπηρεσίες του διαδικτύου να είναι προσβάσιμες από όλους τους πολίτες της Κοινωνίας της Πληροφορίας, συμπεριλαμβανομένων και των ατόμων με αναπηρία, μέσα από μία ποικιλία διαφορετικών συσκευών.

Η προσβασιμότητα του διαδικτύου αντιμετωπίζεται κυρίως μέσα από οδηγίες και "de facto" πρότυπα ανάπτυξης λογισμικού, και απαιτείται βάσει νόμου σε πολλές χώρες. Επίσης είναι διαθέσιμα εργαλεία που ελέγχουν ότι οι εφαρμογές που αναπτύσσονται ακολουθούν αυτές τις οδηγίες. Ωστόσο, είναι κοινά αποδεκτό ότι οι περισσότερες εφαρμογές και υπηρεσίες του διαδικτύου δεν είναι προσβάσιμες από άτομα με αναπηρία, ως αποτέλεσμα των περιορισμένων γνώσεων των κατασκευαστών λογισμικού για τον Παγκόσμιο Ιστό σε θέματα προσβασιμότητας και στα σχετικά εργαλεία. Επιπλέον, για την ανάπτυξη διεπαφών του διαδικτύου, κατάλληλων για διαφορετικές τεχνολογικές πλατφόρμες και συσκευές, απαιτείται πρόσθετο κόστος υλοποίησης και συντήρησης εναλλακτικών εκδόσεων των διεπαφών αυτών.

Επομένως, είναι ιδιαιτέρως σημαντικό να αντιμετωπιστούν οι παραπάνω προκλήσεις με εργαλεία που με εύκολο τρόπο βοηθούν στην ανάπτυξη διαδικτυακών διεπαφών ικανών να προσαρμόσουν τη συμπεριφορά τους με ευφυή τρόπο, δηλ. να προσαρμοστούν στα ποικίλα χαρακτηριστικά των χρηστών, στις απαιτήσεις και τις προτιμήσεις τους, καθώς και στα τεχνικά χαρακτηριστικά των διάφορων συσκευών, όπως η διάσταση οθόνης, αποφεύγοντας την ανάγκη για πολλαπλές υλοποιήσεις. Με αυτό το σκοπό, η διατριβή αυτή προτείνει ένα διαδικτυακό εργαλείο συγγραφής φορμών αλληλεπίδρασης οι οποίες:

- βρίσκονται σε πλήρη εναρμόνιση με το τρέχον " de facto" πρότυπο προσβασιμότητας του διαδικτύου WCAG 1.0,
- μπορούν να προσαρμοστούν σύμφωνα με τις ανάγκες και τις προτιμήσεις του εκάστοτε χρήστη,
- είναι προσβάσιμες μέσα από διάφορες πλατφόρμες και συσκευές
- μπορούν να κατασκευαστούν χωρίς να απαιτείται εξειδικευμένη γνώση της προσβασιμότητας του διαδικτύου, ή ακόμα και της ανάπτυξης εφαρμογών διαδικτύου

Καθώς το εργαλείο που αναπτύχθηκε είναι πλήρως προσβάσιμο, αποτελεί ένα βήμα προς την παροχή ενός επαρκούς μέσου για τα άτομα με αναπηρία ώστε να είναι σε θέση τα ίδια να παράγουν προσβάσιμο περιεχόμενο για τον Παγκόσμιο Ιστό.

Abstract

The Web is becoming an increasingly important vehicle for many types of every day activities. At the same time, the technological platforms and devices used to access the Web are increasing both in number and complexity (e.g., PDAs, mobile devices, iTV, etc). Therefore, it is of crucial importance that web applications and services are accessible to all citizens in the Information Society, including people with disabilities, through a variety of different devices.

Currently, Web accessibility is addressed mainly through development guidelines and de facto standards, and is required by law in many countries. Tools for assessing conformance to such guidelines are also available. Still, it is widely recognized that most web applications and services are not accessible by disabled people, due to limited expertise of web developers in web accessibility issues and related tools. Additionally, the development of web interfaces appropriate for different technological platforms and devices often requires the additional cost of maintaining alternative versions.

Therefore, it becomes particularly important to face the above challenges in a cost effective way, by producing web interface elements capable of intelligent adaptation behaviour, i.e., capable of adapting to users, diverse characteristics, requirements and preferences, as well as to different devices, screen resolution, and aspect ratio, avoiding the need of multiple implementation. Towards this end, this thesis proposes an online tool which supports the development of fully accessible web resources, and in particular web forms, that:

- Fully comply with current web accessibility de facto standard.
- Can be personalised according to users, needs and preferences.
- Can be accessed through various platforms and devices.
- Can be constructed without specific knowledge of web accessibility, or indeed web development.

As the developed tool is itself fully accessible, it also constitutes a step towards providing adequate means for people with disabilities to author web resources themselves.

Table of Contents

| | |
|---------------------------------------------------------------------------------------|----|
| Acknowledgements..... | iv |
| 1 Introduction..... | 1 |
| 0.1 The World Wide Web and people with disabilities..... | 1 |
| 0.2 Universal Access to the World Wide Web | 2 |
| 0.3 Universally Accessible Web Forms..... | 3 |
| 0.4 Objectives of this thesis | 4 |
| 0.5 Structure of this thesis..... | 7 |
| 1 Web Accessibility: current practices | 8 |
| 1.1 Problems with Web Access | 8 |
| 1.2 Benefits to People without Disabilities..... | 9 |
| 1.3 Problems in Relation with the Use of Browsers and Devices | 10 |
| 1.3.1 Variety of Web Browsers..... | 10 |
| 1.3.2 Devices with Different Screen Sizes..... | 10 |
| 1.3.3 Different Font Sizes | 10 |
| 1.3.4 Invalid HTML Code | 10 |
| 1.4 Limited Provision of Web Accessibility..... | 11 |
| 1.5 Accessibility and Device Independence | 11 |
| 1.6 Evaluating web pages accessibility..... | 12 |
| 1.6.1 Basic Checkpoints..... | 12 |
| 1.6.2 Manual checkpoints and content accessibility..... | 17 |
| 1.7 Browser compatibility..... | 22 |
| 1.8 Web Accessibility evaluation tools..... | 23 |
| 1.8.1 Bobby Watchfire..... | 23 |
| 1.8.2 A-Prompt..... | 23 |
| 1.9 Discussion..... | 24 |
| 2 Creation of Accessible Web Forms: Methodologies and Tools | 26 |
| 2.1 Methodologies | 26 |
| 2.1.1 Accessibility by design | 26 |
| 2.1.2 Filter and transformation tools..... | 27 |
| 2.2 Platforms for the creation of web services..... | 28 |
| 2.2.1 Microsoft Office SharePoint Server..... | 29 |
| 2.2.2 Adobe ColdFusion | 30 |
| 2.2.3 eZ Publish | 31 |
| 2.3 Authoring tools | 32 |
| 2.3.1 Macromedia Dreamweaver MX | 32 |
| 2.3.2 Accessible Form Creator (HiSoftware) | 33 |
| 2.4 Web forms development requirements | 33 |
| 2.4.1 Separating presentation from content | 33 |
| 2.4.2 Full compliance with W3C Accessibility guidelines..... | 38 |
| 2.4.3 Device independence | 39 |
| 2.4.4 Personalisation capabilities..... | 45 |
| 2.5 Requirements for tools supporting the development of universally accessible forms | 53 |
| 3 Accessible Online Tools supporting the accessibility of Web elements | 55 |

| | |
|---------------------------------------------------------------------------------|----|
| 3.1 The Web Harmonia Platform..... | 55 |
| 3.2 Requirements analysis | 58 |
| 3.2.1 Context analysis..... | 58 |
| 3.2.2 System goals specification..... | 59 |
| 3.2.3 Software requirements | 59 |
| 3.2.4 Interface requirements | 59 |
| 3.3 Preliminary Prototyping..... | 59 |
| 3.4 Tool’s Architecture..... | 60 |
| 4 Functionality and User Interfaces of the Designer and Web Creator tools | 62 |
| 4.1 “Designer” Tool | 62 |
| 4.1.1 Create / Edit Design..... | 62 |
| 4.1.2 Design Elements | 63 |
| 4.1.3 Preview Design..... | 64 |
| 4.2 “Form Creator” | 65 |
| 4.2.1 Create a Form..... | 66 |
| 4.2.2 Form Elements..... | 67 |
| 4.2.3 Preview Form..... | 70 |
| 4.3 Device Independence and Personalization Support..... | 70 |
| 4.4 Cost-Effectiveness Analysis | 72 |
| 4.4.1 Testing the Cost-effectiveness of the Form Creator | 73 |
| 4.5 Discussion..... | 75 |
| 5 Evaluation | 76 |
| 5.1 Methodology..... | 76 |
| 5.2 Evaluation Results | 78 |
| 6 Conclusions and Future Work | 80 |
| 6.1 Contribution of this Thesis..... | 80 |
| 6.2 Future directions | 81 |
| 7 References..... | 83 |

Table of Figures

| | |
|--------------------------------------------------------------------------------------------------------------------------------------|----|
| Figure 1: Illustration of Web roadmap displaying markup technologies introduced and main standards for creation of web content | 6 |
| Figure 2: Web accessibility tutorial participant using computer with headstick | 8 |
| Figure 3: Steps involved in “Accessibility by design” | 27 |
| Figure 4: Three-tier architecture | 36 |
| Figure 5: Adaptive display of Web content with reconfigurations..... | 42 |
| Figure 6: A DOM-tree navigation of the MSN home page via Mobile Web. | 43 |
| Figure 7: Analysis Results | 44 |
| Figure 8: Using Link personalization in www.amazon.com | 46 |
| Figure 9: Structure customization in my.yahoo.com..... | 47 |
| Figure 10: Personalization in WAP Portals | 48 |
| Figure 11: Structure Customization according to users’ roles..... | 49 |
| Figure 12: Node content personalization in the ATL intranet | 51 |
| Figure 13: General architecture of Web Harmonia..... | 57 |
| Figure 14: Supporting tools of the Web Harmonia platform..... | 58 |
| Figure 15: Example of mock-ups produced during the iterative prototyping of the developed tools | 60 |
| Figure 16: Create Design Step 1 | 62 |
| Figure 17: Create Design Step 2 | 63 |
| Figure 18: Design Elements list..... | 64 |
| Figure 19: D Designer Pre-viewer | 65 |
| Figure 20: Form Creator - forms list..... | 65 |
| Figure 21: Creating a Form..... | 66 |
| Figure 22: Add free text..... | 67 |
| Figure 23: Add open-ended question..... | 67 |
| Figure 24: Multiple choice question creator | 68 |
| Figure 25: Multiple choice answers..... | 69 |
| Figure 26: Previewing Form | 70 |
| Figure 27: Previewing a demo form (Desktop pc)..... | 71 |
| Figure 28: Previewing a demo form (PDA)..... | 72 |

Figure 29: Dreamweaver requires experience in web development, accessibility standards, and produces a WYSIWYG solution.....74

Figure 30: Form Creator requires minimum experience in web development74

Figure 31: The Form Creator tool complies with WCAG 1.0 Level AAA (automatic check)75

1 Introduction

1.1 *The World Wide Web and people with disabilities*

The Web is an increasingly important resource in many aspects of life. Therefore, it is essential that the Web is accessible in order to provide equal access and equal opportunity to all citizens in the Information Society, including people with disabilities. The basis of web accessibility is that every user, including people with disability, should have access to the information and experiences available online.

The diversity of the human population encompasses a vast range of physical, sensory and intellectual differences. Web accessibility addresses all disabilities that affect access to the Web, including visual, auditory, physical, speech, cognitive, and neurological disabilities. For example:

- Some people cannot use their arms or hands to type or move a mouse
- Some people with tremors and older people with diminishing fine motor control can use a keyboard, but not a mouse.
- Some people cannot see at all and use a screen reader that reads aloud the information in the web page.
- Some people have blurry vision and cannot read text unless it is very large.

Numbers signify that large portions of population have some kind of disability. The level of impairment that determines the criteria for whether a person is considered to be disabled varies from country to country, and in some cases within a country. Notwithstanding these difficulties, it is pretty safe to say that somewhere between 10% and 20% of the population of a developed country like Australia has a disability that will affect their ability to use the Web. The proportion of the population with a disability appears to be increasing, due mainly to the aging population profile of these countries.

The incidence of disabilities (and limiting illness) that restricts a person's ability to function in everyday life, as recorded by government agencies in some countries (Hudson, R., 2005)

- United Kingdom, 18% of the population (National Statistics, 2001).
- Australia, 17% of the population (Australian Bureau of Statistics, 2003).
- United States, 19.3% of the population (US Census Bureau, 2000).
- Canada, 12.1% of the population (Statistics Canada, 2001).
- New Zealand, 20% of the population (Statistics New Zealand, 2001).
- European Union, across the 15 EU countries in 2001, 19.3% of the population was hampered by physical or mental health problem, illness or disability, with 9.3% severely hampered. (Eurostat, 2003)

In the light of these numbers, still people with disabilities use the Web. A Survey on Income and Program Participation or SIPP (Bureau of the Census, 2000) estimated the number of people with certain disabilities and “access” to the Internet. According to this study, the following percentages of disabled people have Internet access:

- 21.1% of people with “vision problems”
- 27.2% of people with “hearing problems”
- 22.5% of people with “difficulty using hands”
- 42.2% of people with a learning disability

To allow people with disabilities to make use of content available on the Web, important regulations and guidelines in the accessibility area have been developed.

However, in addition to providing opportunities for people with disabilities to get information from the Web and to interact through the Web, there are opportunities for people to provide information through the Web. Still, in most cases, the Web tends to be inaccessible to those with visual, auditory, or other physical impairments, and there are currently no accessible tools for web content development.

1.2 Universal Access to the World Wide Web

As every day activities become more and more dependent on the Web, the impact of the Digital Divide caused by differences in accessibility to the new technologies (e.g., PDAs, mobile devices, iTV, etc) is growing in terms of widening differences in all aspects of life, and affects all people without discrimination. As a result, the internal structure of a web interfaces is becoming increasingly complex, as there is a continuous demand to support these new devices, user preferences, tasks and environments. These increasingly rapid technological changes are likely to significantly change to the way web interfaces and content are been created.

Users want to view Internet content and use web applications on a variety of devices. Current devices support a variety of different content types partly determined by their underlying hardware capabilities. In order to support device independence, web providers must be able to deliver content in a format compatible with a device. For example if a handheld device can read GIF images but not JPEG images it is necessary to convert one format to another. In addition the content must reflect the underlying hardware capabilities of the device so we may need to do some additional image processing if the target device can only display four level grey scale output.

So it becomes clear that the Web has evolved into an amazing multimedia environment, and users desire to access it by several channels. However in order this environment to be reachable by several devices controlled by “standard” users, the complexity of web pages has increased and created new barriers for people with disabilities. One definition of web accessibility states: accessible web sites are perceivable, operable and understandable by the broadest possible range of users and compatible with their wide range of assistive technologies, now and in the future (W3C-WAI, 2003). With this definition a link between accessibility and device independence is been made, although the concept of accessibility is far older. Because there is no such thing as a “standard” user, hardware, or

navigation software, it is important that Web is designed to support all users, all computer systems, and all web browsers.

It is also important to note that, while access to people with disabilities is the primary focus of web accessibility, it also benefits people without disabilities. For example, a key principle of web accessibility is designing web content that is flexible to meet different user needs. This flexibility also increases general usability and let people without disabilities use websites according to their preferences, such as using whichever browser they want and using keyboard shortcuts. In particular, the following groups will benefit from accessible web content: people with low literacy level, users using new devices such as mobile phones, Web-TV and kiosks, low bandwidth users, users in a noisy environment, users who are driving and users with different learning styles.

It is therefore also important to look into new challenges that web accessibility is facing, such as:

- Provision of interface alternatives for new devices and technologies that implement the paradigm of ambient Intelligence and ubiquitous computing to allow access to information in different environments.
- Intelligent adaptation behaviour and personalisation to address different display sizes, differentiation of input devices and environment in use
- Constant evolution of web technologies specifications. For example, XHTML 1.0 was created shortly after HTML 4.01 to help the transition of hypertext to a new generation of mark-up languages for text, and other mark-up versions are likely to follow soon.

Therefore, access to the web needs to be considered in the wider perspective of Universal Access, ([Stephanidis et al., 1998a], [Stephanidis et al., 1999], [Stephanidis, 2001a]), aiming at the provision of access to anyone, from anywhere and at anytime, through a variety of computing platforms and devices, to diverse products and services..

1.3 Universally Accessible Web Forms

According to Wikipedia¹, a Web form on a web page allows a user to enter data that is, typically, sent to a server for processing and to mimic the usage of paper forms. Forms can be used to submit data to save on a server (e.g., ordering a product) or can be used to retrieve data (e.g., searching on a search engine). Web forms are included in almost all web sites since they allow transactions and data gathering from users. And as the Web grows, more and more transactions will be done online bringing benefits to all Web users, as they are increasingly going online to do their banking and shopping, request information, book travel, pay bills and participate in education and training courses.

Forms are one of the most difficult aspects of web development, largely because it means stepping out from simply presenting information to the user of a site. An especially challenging task is creating accessible online forms, particularly forms that are accessible

¹ Wikipedia, the free encyclopedia available at: http://en.wikipedia.org/wiki/Main_Page

to screen reader users. This is due to the fact that there are a variety of form control types — text, checkboxes, radio buttons, menus, etc. —each with its own distinct accessibility challenges. Additionally, different screen readers handle form control types in different and somewhat unpredictable ways.

For example, a JAWS® user may attempt to complete an online form by activating the "Forms Mode," where the user navigates among form controls using the TAB key or combinations of TAB, SHIFT, and CONTROL. Upon tabbing to a form element, that element gains focus, and JAWS announces certain information that it perceives to be relevant. Without specific accessible mark-up, however, JAWS is forced to make judgments about what information is relevant. For text boxes and text area boxes, JAWS announces the nearest text preceding the form control; for checkboxes, JAWS reads the nearest text following the form control; for radio buttons, JAWS reads the text preceding the complete set of radio buttons, plus (for each button) the text that immediately follows that button. If text labels are positioned differently from what JAWS expects, JAWS will provide either incorrect or insufficient information.

Other screen readers behave somewhat similarly, but there are significant differences across screen readers in how forms are processed, particularly concerning which information about a form control the screen reader judges to be relevant. Therefore, in order to make HTML forms reliably accessible to screen readers, web developers must provide sufficient information to screen readers regarding what to say when they encounter specific form controls.

Currently, the creation of accessible web forms is addressed in two major sets of standards and guidelines for web accessibility: Section 508 of the Rehabilitation Act, and the Web Content Accessibility Guidelines (WCAG 1.0) developed by the W3C. The 1998 amendments to Section 508 of the Rehabilitation Act mandate that only accessible information technology can be acquired and used by the federal government. These amendments also establish accessibility standards for websites. These standards must be met when federal departments and agencies procure, develop, use, maintain or upgrade electronic and information technology. The WCAG 1.0 might be better suited for all kinds of situations dealing with the creation of accessible content, and web forms as well. These guidelines include checkpoints for compliance and coding examples.

1.4 Objectives of this thesis

There are several common challenges encountered when working with web developers to address web accessibility. The true art of web accessibility is not so much in conforming to the guidelines but in implementing them effectively and efficiently. Conforming to the well establish web accessibility standard (WCAG 1.0) can be achieved in many ways, but it is often particularly effective to consider the accessibility requirements of the end-users from the start of a web project. The introduction of accessibility into a web form that is already fully developed can involve significant redesign and recoding, which may be considered outside a web page's scope and budget. Therefore, addressing the issue of accessibility as early as possible (during user-interface design and specification) reduces the risk of redevelopment, but does not eliminate the need for significant manual testing

and recoding, which requires proper knowledge for applying accessibility and usability guidelines.

Methods and techniques can be learnt for applying this standard early, effectively and efficiently for a construction of an accessible web form. However, training for the Web development team remains a key aspect for the successful implementation of all accessibility features issue. Apparently, there is a basic reason why web developers are reluctant to introduce accessibility engineering into their development process. Most of them claim that depending on the lifecycle phase of a web product, incorporating accessibility may vary from simply altering some HTML files and HTML tags to (re-) designing from scratch accessible forms with specific auxiliary elements that are quite difficult to handle. The latter is undoubtedly a nontrivial task that may require active participation of accessibility experts and trained Web developers rather than using for a couple of seconds an authoring tool or a form wizard for this purpose.

In addition, to ensure device independence, web forms must be designed so that, when rendered by browsers, the user's preferred input and output devices are supported. A user may prefer any combination of mouse, standard and non-standard keyboards, voice input or output, head wand, standard and non-standard pointing devices, or Braille devices. In order to achieve this, a web developer must cope with different mark-up versions (e.g., HTML 4.01, XHTML 1.0, XHTML 1.1, cHTML, WML, CSS 1, CSS 2) and validate the outcome of his/hers work through the use of various graphical user interface (GUI) browsers (e.g., Internet Explorer, Netscape Navigator, Firefox, and Opera) and devices (e.g., desktop, PDAs, mobile phones) while adjusting the browsers' settings, or managing multiple devices with mechanisms to define alternative behaviour (see Figure 1). With the use of CC/PP (Composite Capabilities/Preferences Profile) for instance, a user with a specific preference or disability-related need can clarify that even though their browser handles millions of colours, they personally can only distinguish certain colours.

In this context, the construction of universally accessible web form implies addressing diversity through the provision of web interface elements capable of intelligent adaptation behaviour. Such challenge should also be addressed in a cost effective way: web forms must offer fully accessible and usable components, capable of adapting to devices, screen resolution and aspect ratio, as well as in their input and output capabilities according to users' preferences, without the need to maintain several implementations for each situation. Due to the nature of web accessibility, it is improbable that web developers can attain designs which adhere to all users issues.

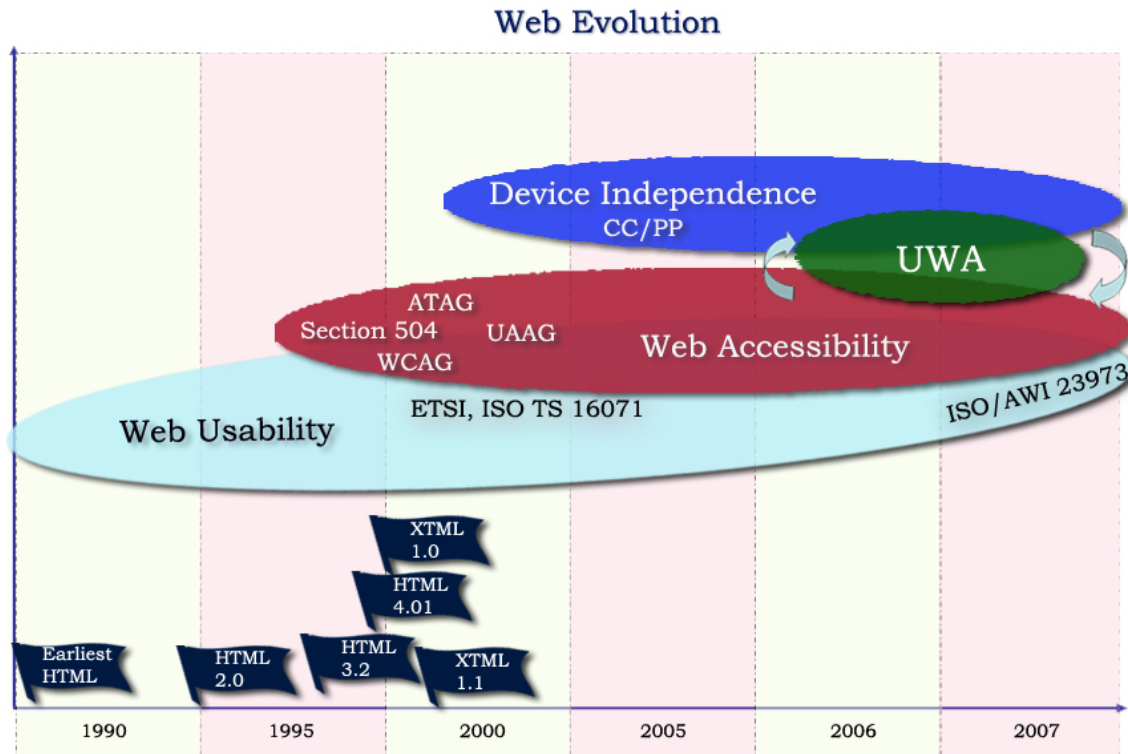


Figure 1: Illustration of Web roadmap displaying markup technologies introduced and main standards for creation of web content

Guidelines and tools exist that support web developers for the production of accessible web forms. However, as W3C states² “we were unaware of any single authoring tool that fully supports production of accessible Web sites”. To this end, conformance with accessibility guidelines is being performed “manually”, due to the fact that until now relevant provisions are not embedded in programming platforms and tools. Therefore, the need arises to provide tools able to:

- deliver fully accessible web forms suitable for all users, anywhere and at anytime
- improve the cost-effectiveness and quality of the production of web forms, with no previous knowledge for web accessibility or and usability guidelines

As a consequence, it is important to look into new ways for the provision of flexibility in online transactions, i.e., to provide accessible tools for the construction of universally accessible web forms. Especially now when Web services have evolved as a practical, cost-effective solution for uniting information distributed between critical applications over operating system, platform, and language barriers that were previously impassable. In this respect, this thesis aims to contribute towards the growing demand for development of universally accessible web interaction forms that will run on and adapt to different user characteristics, as well as to the characteristics of multiple display devices, and in particular:

² Selecting and Using Authoring Tools for Web Accessibility: <http://www.w3.org/WAI/impl/software>

- Fully comply with WCAG 1.0
- Can be accessed by various devices through a web browser
- Can be personalised according to user preferences
- Can be constructed without prior knowledge of web development

1.5 Structure of this thesis

Chapter 2 of this thesis discusses current approaches and practices in web accessibility, focusing on web forms construction, and Chapter 3 put forwards a number of requirements that an accessible tool for the development of accessible forms should exhibit. Chapter 4 presents briefly Web Harmonia, an interaction platform that facilitate the actual production of universally accessible web interfaces in a cost-effective and standards-compatible way, used as a basis for the construction of the Designer and Form Creator, online tools developed to satisfy the requirements elaborated in Section 3. The architecture and implementation of the tools in the context of Web Harmonia are reported in details. Chapter 5 presents the tools functionality and user interfaces, and describes step-by-step an example scenario of use of the tools for generating a portal's registration pages and their forms, focusing on their adaptation capabilities for users with different types of disabilities and for use with several different types of interaction platforms. . Chapter 6 presents the accessibility and usability evaluation of the tools an dof generated forms through their automatic evaluation (e.g., using Bobby Watchfire) and user tests with a blind user and a web developer. Finally, section 7 concludes the thesis and discusses potential future work.

2 Web Accessibility: current practices

According to World Wide Web Consortium (W3C)³, “Web accessibility means that people with disabilities can use the Web”.



Figure 2: Web accessibility tutorial participant using computer with headstick

In a number of countries, Web accessibility is nowadays required by law (e.g., U.S. Code, 1998) and policies (e.g., European Parliament, 2002). As a result, a number of standards, guidelines, checklists and techniques for Web accessibility have been proposed worldwide. For instance, according to the World Wide Web Consortium - Web Accessibility Initiative (W3C-WAI), websites must at least conform to all Priority 1 checkpoints of the Web Content Accessibility Guidelines (WCAG 1.0). In the USA, websites are additionally required to comply with the provisions of Section 508 of the US Rehabilitation Act. However, results of recent surveys show a very low conformance to any of these guidelines and the majority of Web-based communication and information for Web tools remain inaccessible to a large number of people. For example, a survey on websites from Ireland, United Kingdom, France and Germany regarding their conformance to WCAG 1.0 and HTML standards indicated an average of 40% Priority 1 diagnostic violations (Marincu & McMullin, 2004). W3C provides more information about Policies Relating to Web Accessibility⁴.

2.1 Problems with Web Access

Although Web accessibility is important for all users, still most of the web sites are inaccessible. As a result, for many disabled people it is not just difficult but impossible to access web content because of the way this content is designed and implemented. Many blind or visually impaired surfers access the Web using software that reads the content

³ Introduction to Web Accessibility <http://www.w3.org/WAI/intro/accessibility.php>:

⁴ Policies Relating to Web Accessibility: <http://www.w3.org/WAI/Policy/>

out to them using a computer generated voice (e.g. screen reader) in the case of graphics with no labels attached to them graphics explaining their purpose. Everyone to a greater or lesser degree has to put up with access problems - but the problems of lack of access are more acute for many disabled people. The following sketch summarily the problem for each disability category.

Visually disabled users range from the colour blind, to the fully blind. These users can have problems understanding images that are not accompanied by a text description of what they show. Without a text description, a user who can not see an image has no way of knowing what it is, or what it represents. Colour blind users may also have trouble discerning design elements (including text) whose colors are not sufficiently different from the elements around them (including the background or page colour). Visually impaired users may also have problems understanding sites that are not built to accommodate "viewing" non visually, for example through a screen reader. A screen reader is a Web browser that reads Websites out loud, thereby making them accessible to visually disabled users. Often a Website that looks nice visually can not be accessible when it is listened to through a screen reader.

Similarly to the issues facing the visually impaired, are those that face Web users with hearing problems. Users with hearing disabilities have no way of understanding information that is communicated with sound. The simple solution is to provide an alternative that does not use sound, such as a text description or an image.

Physically disabled users are often incapable of using a mouse. Unless these users' needs are taken into account when creating Website navigation and input methods (see Figure 2), physically disabled users may find a site completely inaccessible.

Websites can be complex, and finding the information can be difficult for the most able of us. The situation is not helped by sites that use an overly complex design, inconsistent navigation, and distracting, repetitive animation. These design elements can compound problems for users with Cognitive and Neurological Disabilities, and can make some sites completely inaccessible for them.

2.2 Benefits to People without Disabilities

Web accessibility also benefits people without disabilities. As an example, alternative text that is used for images and links, which provide descriptions for those who can not view the image, let users know what images are about and provides more information on pages linked too before they click. Web content providers should be aware of auxiliary benefits of Web accessibility as fully accessible pages work better for everyone who uses the web, not just those with a special need or limited ability. Making web content accessible can increase its usability dramatically. One way and another, users may confront with Web access problems due to user individual factors (e.g., medical, age, low literacy, etc) as well as by hardware, and software approaches.

2.3 Problems in Relation with the Use of Browsers and Devices

2.3.1 Variety of Web Browsers

A Web browser is a translation device. It takes a electronic file that is been produced by a Web server written in HTML mark-up language, and translates it into a “visually formatted” Web page. Although web browsers build in order to perform essentially the same thing according to W3C specifications, they go about it in slightly different ways, and can produce significantly different displays. Even the same name-of-browser and version number, on different systems will produce different viewing results.

For example, the HTML standards say that the TABLE tag should support a CELLSPACING attribute to define the space between parts of the table. But standards don't define the default value for that attribute, so unless you explicitly define CELLSPACING when building your page, two browsers may use different amounts of white space in your table. Another example is that each browser and each computer platform display fonts slightly differently. Such differentiations affect the accessibility level of the resulted pages as each produced component may be translated differently by assistive technology devices.

Designing and maintaining web content for different versions of a web browser causes problems for Web designers, because rushing to build content with the features supported by a newest version of a browser will have to wait.

2.3.2 Devices with Different Screen Sizes

Many experienced Web designers use techniques to control their web page layout, yet they design their pages on large, 1024x768 pixel screens. When these pages are displayed on smaller screens (e.g. PDAs, old desktop displays), the browser may not be able to fit all the content onto the screen. In these cases, the content will scroll of the right of the page. While this may not sound like much of a problem, users hate scrolling left and right to view a page.

2.3.3 Different Font Sizes

Most browsers allow users to customize their default font size. Many users who work on computers all day do this to reduce eye strain. As a result, user preferences may cause the typeface that you used to design your Web page to increase as much as 50% larger in a user's browser. This increase in font size can hurt many carefully-planned page designs.

2.3.4 Invalid HTML Code

Invalid mark-up code, which is the use of HTML code without following the proper mark-up specification (e.g. HTML 4.01, XHTML 1.0), can affect accessibility. It seems strange to consider that a few coding errors could cause all those problems, but it's true. Even the simplest errors can cause big problems - particularly in browsers which adhere

more stringently to W3C standards. Not to mention that screen readers often have problems with mark-up code errors - particularly missing attributes.

2.4 Limited Provision of Web Accessibility

Apparently, there are quite some reasons why providers are reluctant to comply with policy and legislation requirements and introduce accessibility engineering into their development process. Most web service providers claim that creating accessible web services requires, among others, a considerable initial investment for equipment and recruitment of a proper team (e.g., training, technical assistance, purchase of authoring software and assistive technologies, “translating” web accessibility guidelines). Depending on the lifecycle phase of a web tool (e.g., new concept, under development tool, deployed and under maintenance product), incorporating accessibility may vary from simply altering some HTML files and HTML tags to (re-) designing from scratch accessible interfaces. The latter is undoubtedly a nontrivial task that may require active participation of accessibility experts and trained Web developers, but even then there are considerable long-term benefits, well demonstrated by a number of experts. Sierkowski (2002) claims that “incorporating accessibility takes time, planning and research however accessibility allows long-term savings”, and according to Clark (2003), Web accessibility induces an increase of only 2% of the original budget. However, providers appear to have low awareness of these benefits.

2.5 Accessibility and Device Independence

A common definition of web accessibility is access to the Web by everyone, regardless of disability. Such definition implies that accessibility is only about people with disabilities (Nyman, 2006).

However, in the context of this thesis, accessibility is intended in a wider sense (Nylander and Bylund, 2002) that also includes device independence – regardless of disability, user agent or platform. As previously stated, the web is evolving towards a situation where it has the potential to make its content available to everyone, regardless of the device, platform, network, culture, geographic location, or physical or mental ability of those using it. Unfortunately, however, existing tools do not provide solutions for all existing combinations of user-platform currently in use or likely to emerge in the future.

Users may view Internet content and use web applications on a variety of devices, including PCs, electronic book readers, PDAs, phones, interactive TVs, voice browsers, printers and embedded devices such as cameras. Although most of them view the Web with a graphical Web browser on a desktop or laptop computer with a colour screen display of 800 by 600 pixels or larger, the Web is not just for those Web browsers and display resolutions. Nowadays, other ways to use the Web are available, including Web browsers of PDAs and mobile phones, text-based Web browsers, screen readers, handheld and subcompact computers, aural browsers, refreshable Braille devices, and more. Due to this device proliferation, web content providers can no longer deliver one version of their content to the user as they need to deliver an appropriate form of content depending on the capabilities of the viewing device. Re-authoring content, in order to support different mark-up languages or the different capabilities of each device, is clearly

impractical, whereas providing content for a single device or browser excludes large numbers of users.

2.6 Evaluating web pages accessibility

Accessibility evaluation is a rather complex process, especially for inexperienced web designers. This section presents a summary of the main guidelines for accessibility evaluation, grouped into two main categories:

- **Basic Checkpoints** concerning accessibility aspects that can be tested with automated tools, as well as some relatively easy manual checks.
 - Valid HTML and CSS
 - Frames
 - Use of automated accessibility checking tools
 - Images and alternative text
 - JavaScript
 - text size
 - semantic markup
 - CSS
 - Screen reader simulators
- **Manual checkpoints** that are more difficult to test with automated tools, and require more time and experience to evaluate manually.
 - Colour contrast
 - Document titles
 - Link text
 - Non HTML formats
 - Platform discrimination
 - Keyboard navigation
 - Data tables
 - Form controls and labels
 - Use a screen reader
 - Content accessibility

Running through the checkpoints described below does not make testing with actual disabled persons using screen readers and other assistive devices unnecessary. However, it can provide a very good indication of the accessibility status of the website under evaluation.

2.6.1 Basic Checkpoints

Valid HTML and CSS

The first step towards assessing accessibility is to validate HTML and Cascaded Style Sheet (CSS) code. However, validity does not equal accessibility. Plenty of sites use valid markup, but still are far from being accessible. Valid markup is also important to ensure

device independence, one of the fundamental building blocks of the web. Using valid markup is as close possible to guaranteeing that the information can be interpreted correctly by as many browsing devices as possible.

W3C validators can be used to check if the HTML and CSS used is valid. It is very convenient to use the Web Developer⁵ extension toolbar for this task (Tools - Validate CSS and Tools - Validate HTML).

The W3C validator sites can also be used:

- W3C Markup validation service⁶
- W3C CSS validator⁷

Note that the HTML of many badly built websites is difficult to validate because there is no DOCTYPE or character encoding specified. For these sites, it may be necessary to manually override in the validator interface.

Some sites completely block access from the validators. For those cases, the use of the Tools ‘Validate Local CSS’ and ‘Validate Local HTML’ in the Web Developer extension toolbar will be necessary. Depending on how invalid the HTML is, the Tool ‘Validate Local CSS’ will also be necessary for some sites that do not block the validators – certain markup errors will make the CSS validator refuse to do its job.

The HTML Validator⁸ extension will also allow to validate sites that block the W3C Markup validator. It does this without sending anything to a server, so this is a good option for sites that are behind a firewall or require a login.

The HTML Validator extension automatically alerts user to any errors and warns client of possible accessibility issues for every page you load in the browser. We can customise the level of the accessibility warnings in the “Options” dialog. We should keep in mind that the HTML Validator extension is based on Tidy, which means that there are cases where it will not report certain errors that the W3C markup validator catches.

Use of frames

While frames (and iframes) are not necessarily completely inaccessible, they generally do make a site less accessible and usable, and should be avoided. Frames are also an indication that the site was built by a web developer with a lack of understanding of both accessibility and usability (Who framed the web - frames and usability⁹).

⁵ <http://chrispederick.com/work/firefox/webdeveloper/>

⁶ <http://validator.w3.org/>

⁷ <http://jigsaw.w3.org/css-validator/>

⁸ <http://users.skynet.be/mgueury/mozilla/>

⁹ http://www.456bereastreet.com/archive/200411/who_framed_the_web_frames_and_usability/

If the page contains frames, the contextual menu in 2007 web browsers (e.g., Firefox 2.0, IE7) will contain an option called “This Frame”. Iframes are a little trickier to find since the user has to context click within the area occupied by the iframe for the “This Frame” option to appear in the contextual menu.

Automated accessibility checking tools

Automated accessibility checking tools (see section 2.8) tend to be used mainly by accessibility novices. The main problem is that the existing automated accessibility evaluation tools are far from perfect, and should not be relied upon.

These tools will help reveal some issues, and can be used as a guide to get a quick and very general overview of the accessibility of a website.. However, there are many possible accessibility problems that these tools will not be able to find, and they will occasionally report problems that are not really problems. A manual follow-up is always needed. A site that passes all automated accessibility checks is not necessarily accessible. Automated tools help to spot accessibility problems that would take much longer to find by manually going through the markup. The Developer has to evaluate the automatically generated report and manually inspect the areas that are flagged as problematic.

Images and alternative text

When no alternative text is specified, or if it is specified improperly, anyone who can not see the images will either miss out on information or get flooded by useless information.

The HTML validator and the accessibility checking tools mentioned above will report if any images or image maps are missing alternative text since the alt attribute is required. If the validator does not report any missing alt attributes, the developer knows that any images in the document have an alt attributes. However, there may still be problems, so it is important to check **how** the alt attribute is used.

By using the browser extensions (such as Web Developer) to show the alternative text of any images in the document, it is necessary to check that the site overall makes sense. Also, it needs to be checked that the alternative text is actually visible when images are turned off – many browsers use the text colour specified for the image (or one of its ancestors) to display the alternative text. If that colour is too close to the background colour that appears when the image is missing, the alternative text will be very hard or impossible to read. This is mostly encountered on sites that use a lot of background images.

Many people misunderstand the alt attribute. It is meant to provide meaningful, informative text that should be used **as an alternative** when an informational image or other graphical object cannot be displayed. It is **not** intended to provide text that is displayed in a tooltip. The alt attribute is **required** for img and area elements.

Informational images should have a short description of the image. Decorative images should have an empty alternative text (Glazkov, 2005). There are sites that in a misguided attempt to be helpful take great care to describe every decorative image and spacer GIF in detail, which is very annoying when someone visits a site with a screen reader or text-only browser (Johansson, 2004).

Use of JavaScript

Not everyone will be able to execute a javascript script or a java applet. For this reason, alternatives need to be provided and scripts should still be usable when javascript is disabled:

- ◆ Many people mistakenly believe that screen readers don't support JavaScript. Technically they may not, but they run on top of browsers that do. Further, screen reader software appears to attempt to infer what a Javascript is trying to do, and then make some sense of it.
- ◆ The theory behind Web Standards methodologies is that we can remove either or both the presentation and behaviour layers and still have plain content. If that is true (and it is true), then shouldn't we be able to leverage that, and even encourage it, if it makes an experience better for some people?
- ◆ Using these methodologies, we can deal with JavaScript on or off, but we can't deal with in between. This is a key problem - screen readers appear to be trying to infer what is supposed to be happening in the page, but they aren't actually interpreting the JavaScript (in the strict sense of the word).

Text size

Many people need or want larger text, in order to be able to read it comfortably. They may have a visual impairment, be over the age of 40 or like to lean back in their chair while browsing the web.

Text size depends on how font size is specified. Font size specified in a relative unit like em or percent lets text resizing work in all browsers. If font size is specified in pixels, users of Internet Explorer for Windows can not change the font size without first changing settings in their browser, which very few people do.

So, when load the site in IE/Win and try resizing the text (Ctrl + Scroll wheel or View - Text Size - Largest), if nothing happens, the site is probably using pixels to specify font size. Even if font size is specified in a relative unit, there could still be problems related to changing the text size. The layout needs to be designed to take into account the possibility of the visitor increasing text size considerable. Many layouts quickly become unusable as text size increases. Obviously, all layouts will break eventually, but a good layout should be able to hold up reasonably even if text size is increased a few hundred percent. It does not have to look as good as it does at the normal text size, but no content should disappear or become unreadable.

If font sizing widgets are added to the site (which is not recommended - Johansson, , 2006), it is important to make sure the largest setting is really large. Otherwise, what's the point in spending time on building those widgets if the end result isn't much different from the default?

Semantic markup

Semantic markup is important, since it gives browsing devices a chance of interpreting and presenting the content in a way that is suitable for the meaning it has. As an additional benefit, search engines also tend to favour semantic markup.

Proper use of headings will also let assistive devices create a document outline which can be very useful for screen reader users.

Even though not all assistive devices use all semantic information they are provided with, if a website does not use semantic markup, it is impossible for any device to derive any meaning from the text.

To determine if the site uses semantic markup, we view source and look for headings (<h1> - <h6>), lists (, , <dl>), quotations (<blockquote>, <q>), and emphasis (,). We can usually spot this pretty quickly since sites that don't use semantic markup tend to have constructs like

```
<span class="bigboldreadheading">Heading</span>
```

where a semantically marked up document would have <h1>Heading</h1>.

Cascaded Style Sheets

An accessible document should be well-structured, meaningful, and readable without CSS.

Disabling CSS will allow seeing the structure of the underlying HTML with your browser's default styling.

If very little happens when the user disables CSS, probably the document uses tables for layout and lots of presentational markup. Usually, a document of this type presents many other accessibility problems.

Screen reader simulators

Screen readers are an important assistive technology. Knowing how a screen reader would speak the contents of the site we are testing will help us determine things like if the order of the content makes sense, if links and headings are used well, and if alternative text is used properly.

When no screen reader application is available, a good way of getting a reasonable feel for how the site under evaluation will sound to someone using a screen reader is to use Fangs¹⁰.

Fangs is a Firefox extension that emulates one of the most widely used screen readers, displaying the output as text instead of using speech.

Results and process iteration

After going through to the previous checklist, any very serious accessibility problem on a page should have been already identified. At this stage, problems can be fixed and the checking process iteratively repeated.

The next section discusses accessibility issues that are difficult to test with automated tools, and require manual evaluation, as well as issues related to web content.

2.6.2 Manual checkpoints and content accessibility

Colour contrast

If there is not enough difference in hue and brightness between foreground and background colours many people will have trouble reading the text. This can be because they have a colour deficiency or because they are using a monochrome or greyscale screen, or they have perfect eyesight, a really good monitor displaying 24-bit colour, and still have problems because the site is using light grey text on a white background. For the same reason it is important not to rely on colour alone to convey information. Links, for instance, should differ from surrounding text not only in colour, for example by also being bold or underlined.

An easy way to check that the difference in hue and lightness between foreground and background colours is good enough is to use Jonathan Snook's Colour Contrast Check¹¹ tool, either by entering the hexadecimal numbers for the foreground and background colours or by pulling the sliders and get live feedback on the contrast and colour difference. Another one similar tool is Gez Lemon's Colour Contrast Analyser¹². Both of these tools use an algorithm provided by the W3C¹³ to calculate colour visibility.

¹⁰ <http://www.standards-schmandards.com/fangs>

¹¹ http://www.snook.ca/technical/colour_contrast/colour.html

¹² <http://juicystudio.com/services/colourcontrast.php>

¹³ <http://www.w3.org/TR/AERT#color-contrast>

One can also change the display to greyscale or monochrome to verify that all text is still readable. Using Mac OS X¹⁴ there are great options for this in the “Universal Access” preference panel, which lets invert the colours, set the display to monochrome or greyscale, and change the contrast.

Document titles

Document titles are important for several reasons: it is often the first thing an assistive device will render when loading a new page, it is used in the browser’s title bar, in bookmarks, and when printing a document. Descriptive document titles are very useful for everybody. They are also extremely important for search engine visibility.

Therefore, it is important to check that each document has a unique and descriptive title and that the title does not use excessive punctuation.

There are no clearly defined rules for which characters to use as title separators. However, document titles should definitely not contain punctuation used for decorative purposes, for instance “:: Title ::” or “...== Title ==...”. Each punctuation character may be read out loud by screen readers, which will make listening to the titles very tedious.

Another screen reader, Apple’s VoiceOver, reads “»” as right pointing double angle quotation mark. That rules out that character for use not only in document titles, but also in links, where it unfortunately is quite popular.

Link text

Links are more useful to everybody if they consist of descriptive text. Most sighted people scan web pages to get a quick idea of their content, and links are often an important part of that content. Clear links make scanning faster. Blind and severely vision impaired people can not scan a page in the same way, and instead tab from link to link or bring up a list of links, so using descriptive link text really helps. Descriptive links are also important for search engine visibility, so this is one more case of accessibility being good SEO.

Links should make sense when read out of context. “Click here” or “here” does not make for good link text since there is no information at all about the link’s target. Link text that consists of an entire paragraph, which is common on newspaper websites, contains too much information and should also be avoided.

In general, the same link text should not be used for links that lead to different destinations. Ideally each link should be able to stand on its own and make sense out of context. Some automated accessibility checking tools will warn you of this.

14

http://www.amazon.com/exec/obidos/redirect?path=ASIN/B0002G71T0&link_code=as2&camp=1789&tag=456bereastree-20&creative=9325

There are exceptions to this. Link texts like “Read more” or “Continue reading” may be ok in a news listing or similar, as long as the title attribute is used to differentiate the links (Clark, 2005).

To get an overview of all links in a document, one can either load the document in Opera and open its Links window, or use the “Links list” feature of the Fangs extension.

Non HTML formats

While it is possible to make the content of PDF files reasonably accessible to people who have the required software, HTML is still the most widely supported format and should always be the preferred choice. Additionally, far from everybody has Microsoft Word or Excel or any means to open documents in those formats. If information is only available in a proprietary Microsoft format (which is often the case), many people are excluded. If the site makes important information available in PDF, Microsoft Word, Microsoft Excel, or other proprietary formats, HTML alternatives should also be provided. There is nothing wrong with making information available in non-HTML formats as long as it is also available as HTML.

Platform discrimination

One of the fundamental properties of the web is that it should be independent of the hardware or software used to access it. The web should be for anyone, everywhere and at anytime. Web accessibility is a much broader issue than catering for disabled people. Platform discrimination is an excellent example of that.

Platform discrimination means partially or completely limiting access to information or functionality for users of minority operating systems or web browsers. To address this checkpoint, one should ideally have access to several different operating systems with multiple browsers installed. That kind of setup is not practical, so an alternative solution is to fake the user agent string of the browser used for testing.

it needs to be checked if the site lets user agents other than Internet Explorer for Windows enter. This action checks only whether the site uses some kind of browser sniffing that is based on the User Agent string. If the discrimination is based on features that only exist on a specific platform, faking the user agent string will not help.

Platform discrimination may be unintentional, but all too often it is intentional. It is extremely rare to find a site that discriminates against users of Internet Explorer for Windows. Most Mac and Linux users, on the other hand, have probably experienced being denied access to several sites because they are using an “unsupported” operating system or browser, which is completely unacceptable.

Keyboard navigation

Screen reader users do not use a mouse to navigate the web, so this is very important to them. There are also many people who are keyboard users by choice, because they find it faster and more convenient than using a mouse.

Depending on the source order and size of the document, it may also be very beneficial to non-mouse users if skip links are available. If the document contains a large number of navigational links before the main content, an in-page link that leads to the start of the main content will save keyboard users a lot of key-presses.

An accessible site is required to be device independent and not rely on visitors to use a particular input device, in this case a mouse.

This point can be checked by trying to navigate the site using only the keyboard, and tabbing through links and form controls. It may be needed to adjust the settings of the browser to be able to do this. Neither Firefox nor Safari has this enabled by default. If there are dropdown or fly-out menus, it needs to be checked if they can be activated without using a mouse.

Data tables

There should be no tables used for layout, and tabular data should be properly marked up with tables that make use of the available accessibility enhancing elements and attributes.

When tables are used to mark up actual data, they are not just a layout grid. Sighted people can get a feel for the relationship between header and data cells by looking at the layout and visual presentation of the table. Blind or severely vision impaired people can not do that. For a table to be accessible to people using a screen reader or some other non-visual user agent, it needs to tell the user agent how the information it contains is related.

HTML provides plenty of elements and attributes for that. However, it can be pretty difficult to understand how to use some of these accessibility features¹⁵.

Form controls and labels

Proper labels help everybody since they makes the clickable area larger. Each visible form control should be explicitly associated with a label element.

Automatically submitting a form when the selected option in a select box is changed causes problems for keyboard users. Requiring JavaScript to submit the form obviously makes it impossible to submit it when JavaScript is not available. Relying on JavaScript for input validation may lead to unexpected values being submitted and stored in the database.

¹⁵ http://www.456bereastreet.com/archive/200410/bring_on_the_tables/

To assess this checkpoint, one needs to check that each form control has a label associated with it, that labels are properly marked up with label elements, and that labels and controls are in the right order (label first, then the control, except for radio buttons and checkboxes which should be control first, then the label).

If the form uses select boxes for navigation, it needs to be checked if the form is automatically submitted (with JavaScript) when an option is selected. For cases where JavaScript is not available, the developer has to make sure that the form can be submitted and that any client-side validation of the entered data is handled by the server.

Automated accessibility tools will normally alert of form controls that do not have an associated label.

Use a screen reader

Experiencing what the web is like to somebody who cannot see is important, since it will help realising the importance of many of the various problematic areas listed above. If user is sighted, it is very hard to imagine what it is like to use the web without seeing it.

However, accessibility is not just about screen readers. Far from it, equating web accessibility with “works in screen reader X” is a very common mistake, and accessibility is much more than that.

Most screen readers are very expensive though, so a demo version can be used. The following is a list of screen readers that are available as demo versions:

- JAWS¹⁶
- Window-Eyes¹⁷
- Supernova¹⁸
- IBM Home Page Reader¹⁹

Content accessibility

If the evaluated site has passed all the checkpoints above, it is quite safe to assume that the site is technically accessible. That, unfortunately, does not necessarily mean that its content is understandable to all.

¹⁶ http://www.freedomscientific.com/fs_products/JAWS_HQ.asp#Downloads

¹⁷ <http://www.gwmicro.com/Window-Eyes/Demo/>

¹⁸ <http://www.dolphincomputeraccess.com/downloads/index.asp>

¹⁹ <http://www-3.ibm.com/able/dwnlds/hpr4trial.html>

Writing or otherwise creating and presenting content that is truly accessible to all can be very difficult. It needs to be kept in mind that badly or incoherently written content can be difficult to understand even for highly intelligent people.

Obviously, understanding a website's content can be even more problematic for people with some kind of cognitive impairment or a learning difficulty (Hudson, Weakley and Firminger, 2005).

2.7 Browser compatibility

It is difficult to build a Web page that displays perfectly on every version of every browser running on every computer. Doing so may require to exclude features that a designer would really like to include on a Web page. Building a Web page that is compatible with Version 1.0 of every browser would mean building a bland page filled with plain text.

Therefore, the first step to solving browser compatibility problems is to determine which browsers are to be really considered.

Avoid the Cutting Edge

Many Web designers feel they have to build cutting-edge features into their Web page. That is a bad idea, because cutting-edge features are rife with browser compatibility problems, not to mention the impact they have on your page load time. Most well-known and well-designed = sites on the Internet, which work under all major browsers (e.g., Yahoo, eBay, Amazon.com). do not cutting-edge features (e.g., Java, Dynamic HTML, Flash). However, they do use JavaScript and Cascading Style Sheets.

Including cutting-edge features in a site does not necessarily cause compatibility problems, but it greatly increases the chance of browser display errors. If the designer really feels that these features are needed, it is essential to test the pages under all major browsers.

Browser Compatibility Report

HTML errors are the leading cause of browser display problems. Making sure that Web pages are error free is one of the most important steps to take in order to solve browser display problems.

That means running an HTML validator, such as the HTML Toolbox, over every page in a site.

Next to HTML errors, compatibility problems are the leading cause of browser display errors. Compatibility dangers extend to all aspects of HTML.

The HTML Toolbox includes a Browser Compatibility report that identifies HTML tags and attributes that are not compatible with the three most recent versions of Netscape Navigator and Internet Explorer.

2.8 Web Accessibility evaluation tools

According to W3C, Web accessibility evaluation tools are software programs or online services that help determine if a Web site is accessible (W3C-WAI, 2007). Many web developers are introduced to web access through accessibility tools. All accessibility tools perform automated checks of web pages for accessibility issues and all generally have additional features, but each tool targets different audiences.

However, web accessibility requires more than just accessibility tools; it requires human judgment. It is important to remember that accessibility tools can only partially check accessibility through automation. Two of the most famous tools that are briefly described below, requiring real understanding of the web accessibility standards rather than relying on a tool to determine if a page is accessible or not.

2.8.1 Bobby Watchfire

Bobby is probably the most widely known of Web accessibility tools at present, and possibly the most commonly used. Bobby is only an automated tool that points out where the guidelines appear to be broken. It can very well be that certain guidelines are indeed followed. Every website is unique and therefore, the Bobby accessibility report should not be taken literally. All information contained in the report should act as support in the process of improving accessibility. A human evaluation by the webmaster supported by the Bobby accessibility report are the optimal instrument one can use into identifying accessibility problems and finding solutions.

Because it covers all accessibility guidelines, Bobby can identify problems which are ultimately easy to fix but often overlooked. It's often the case where there was no one to point them out. At the end of the process webmasters will see that by thoroughly going through the report and making minor or less minor changes to their website, the degree of accessibility of their website will have improved significantly.

Bobby has been heavily criticized for issuing misleading messages. For example, Bobby always says, even in "Bobby AAA Approved" messages: "If you can't make a page accessible, construct an alternate accessible version." (At least I think it always says that. The structure of Bobby's messages actually suggests that it would be "triggered" by some features on a page, but Bobby does not tell what those features might be.) How does that apply? Besides, it's basically wrong advice anyway: if you can't make a page accessible, you should try smarter.

2.8.2 A-Prompt

A-Prompt (Accessibility Prompt) is a software tool designed to help Web authors improve the usability of Web pages created in HTML format. A-Prompt first evaluates an HTML Web page to identify barriers to accessibility by people with disabilities. A-

Prompt then provides the Web author with a fast and easy way to make the necessary repairs. The tool's evaluation and repair checklist is based on accessibility guidelines created and maintained by the Web Accessibility Initiative of the World Wide Web Consortium.

By taking this approach, A-Prompt helps Web authors to include HTML features which widen the range of users who can access their website. As well as providing better access for people with disabilities, the resulting Web pages are generally improved for all people and in a larger variety of circumstances. For example, the inclusion of text alternatives for all images makes it possible to understand Web pages in a low-bandwidth text-only situation.

A-Prompt is a very useful program in many ways, but the user needs to understand the basics of accessibility and to learn some specific features and peculiarities in the program. The documentation is well-written and includes a help system that explains the checks and repairs in detail.

Unfortunately, the instructions are subjective and partly rather debatable. There are often good reasons to use alt texts that are longer than ten words. For practical reasons, mainly bugs and deficiencies in current graphic browsers, an 'alt' attribute value works better if it is a one-liner, but for accessibility, it should simply be as long as needed.

2.9 Discussion

This Chapter has reviewed current practices in the domain of web accessibility, and has provided a summary of the accessibility evaluation process and of the role of automatic evaluation tools. The conclusion can be drawn that accessibility is complex issue, which in current practices requires considerable expertise both in development and assessment methods in order to be applied successfully.

Accessibility evaluation tools provide web developers with a very useful first step toward web accessibility. However, it is important to remember that using such tools to check for accessibility is just a first step toward web access. Web developers who understand WCAG 1.0 (or the new WCAG 2) and how they should be implemented into web sites will be able to help a web development team more than any evaluation tool.

Additionally, a high percentage of the web developers are not familiar with the use of such tools. The growing number of authoring tools and platforms, as the ones that will be discussed in the next Chapter, that provide a WYSIWYG environment which hide from the developer many standard tag attributes, has also favoured the emergence of a generation of web developers fully focused on the graphic design, and not very familiar with other subtleties of HTML.

Most web service providers claim that creating accessible web forms requires, among others, a considerable initial investment for equipment and recruitment of a proper team

(e.g., training, technical assistance, purchase of authoring software and assistive technologies). Available resources, the economic cost and the constraints of 'universal accessibility' are problems that need to be resolved. Therefore, in order to promote accessibility and its application by developers, it is necessary to offer to web developers appropriate tools which reduce the need of accessibility know-how and simplify the development process. Two such tools are proposed in the context of this thesis and described in Chapter 4, focussing on the development of universally accessible web forms. By offering proper tools for the development of fully accessible web interaction elements, the overall accessibility of products and environments can be greatly improved. In a universal access perspective, such provision entails increased benefits not only for people with disabilities but also for those users who access the Web through mobile devices, Web-TV, kiosks or low-bandwidth devices.

3 Creation of Accessible Web Forms: Methodologies and Tools

Web forms are an important tool for users to provide information to a web application or service. E-commerce sites use forms to find out what people want to buy, where they want their purchases delivered, and how they are going to pay. Distance and e-learning sites use web forms for a variety of purposes. Students complete a form when they register online to a e-class.

Web forms are powerful tools. But they can pose significant accessibility barriers, especially for people who use screen readers, talking browsers, text browsers, or refreshable Braille displays. In order for these assistive technology devices to work properly, the form controls (e.g. text input fields, buttons, etc.) must be labeled in such a way that assistive technology can associate the correct label with the form control.

Accessibility of web forms usually refers to their access by to people who use screen readers. People with other types of disabilities are generally less affected by faulty forms. It should be noted, however, that everyone benefits from a well-organized, highly usable form, especially those with cognitive disabilities. Web forms are maybe the biggest problem for them because they require them to do a lot of typing, which is understandably difficult. Even to those people using a PDA or a mobile device.

This chapter reviews the main available platforms and tools for the development of web services and mainly web forms, and, based on such an analysis, put forwards set of fundamental requirements for a new tool supporting the creation of accessible web forms.

3.1 Methodologies

Two are the main methodologies for the provision of accessible material: (a) the implementation of accessible web elements as a result of a systematic design and development approach that adhere to well-established accessibility design principles (“Accessibility by design”); and (b) the use of tools that can be installed “on-top” of a Web tool, to provide accessible versions of its Web pages (“Filter and transformation tools”).

3.1.1 Accessibility by design

This approach mainly involves the following steps (see Figure 3):

- **Production** of a web template (interface) that is compliant with existing accessibility standards for web content, such as WCAG 1.0 and those provided by Section 508 regulations.
- **Validation** of the produced accessibility
 - through the use of one or more of the available evaluation tools²⁰, and

²⁰

Evaluation Tools: <http://www.w3.org/WAI/ER/existingtools.html#Evaluation>

- through manual evaluation by experts.
- **User testing** with various assistive technologies, such as screen readers and user agents to assess whether the final product can be usable by the widest possible range of users. In general, any web tool intended for use by people with disability has to involve disabled users during the entire development process.

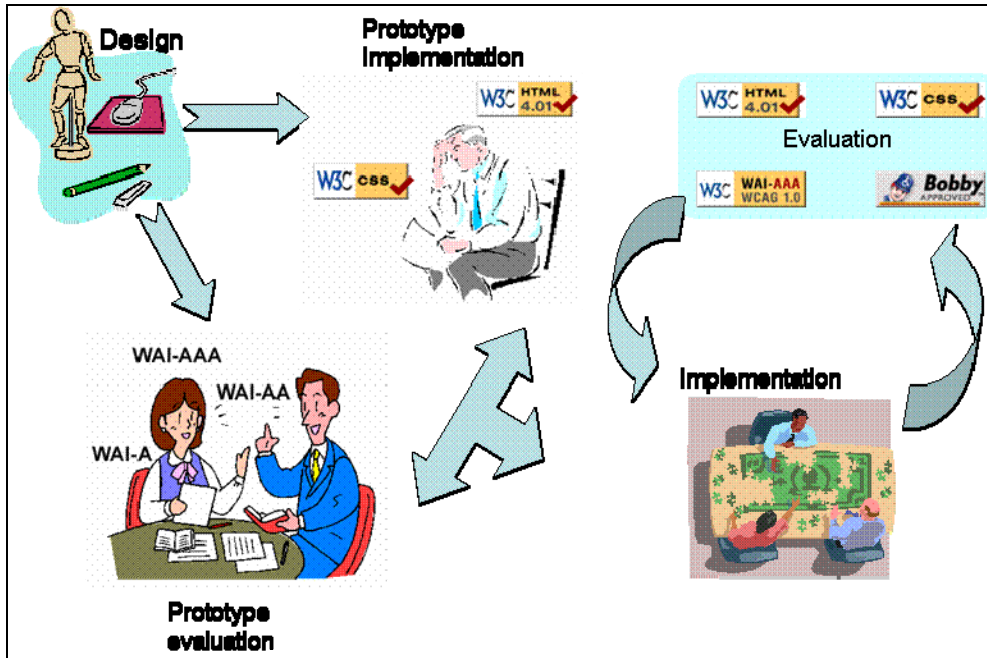


Figure 3: Steps involved in “Accessibility by design”

Despite the high availability of resources that can be of assistance throughout all these steps, Web accessibility still seems to be a complex issue to address for most web designers and developers, as they all require substantial effort, planning, and well trained personnel. Although it is possible to reuse the same principles and practical solutions for the production of accessible web templates, design and implementation of these web templates is a time-consuming procedure, especially when considering the pre-required training of the involved team.

3.1.2 Filter and transformation tools

Reasonably, most providers are not thrilled with the idea of redeveloping from scratch existing web tools only for the purpose of making them accessible. Therefore, suitable alternatives are required for incorporating Web accessibility in such cases, such as that of filter and transformation tools²¹, which allow web users to access pages through a proxy kind of service that adapts pages according to the needs of individual users. Such an example is the *WebFACE* tool which allows the dynamic transformations of web pages,

²¹ Filter and Transformation Tools: <http://www.w3.org/WAI/ER/existingtools.html#Filter>

and automatically introduces web accessibility enhancements to existing web sites (Alexandraki et al., 2004). In general and according to W3C, these tools assist Web users (rather than Web authors) to either modify a page or supplement an assistive technology or a browser. A particular advantage of this approach is the scalability and reusability of such tools. A second and more important reason for choosing this approach is that the control of accessibility features is granted directly to the web user, therefore allowing for different users to precisely define their preferences according to their individual needs. The issue that arises though is: How and to what degree can web service providers utilize such tools effectively?

First of all, it should be noted that such a tool is not be able transform automatically any Web page to an accessible version as, for example, the page may be broken beyond repair (i.e., the page contains invalid HTML and, in most cases, does not look the same in different browsers), which implies that a considerable amount of human effort needs to be put on revising the source code of the existing Web pages. In this process, the existing look-and-feel is not necessarily altered, but proper adjustments must be applied in terms of ‘valid web programming code’ to conform with standards for web-based technologies (e.g., those set by the W3C for HTML, XHTML, CSS, DOM, and SMIL), as well as to integrate some basic aspects of accessible Web design (e.g., to provide appropriate Alt descriptions to all images). In any case, these adjustments are much easier to implement than the task of producing accessible Web pages by design. Finally, once these adjustments are made, the software package that will allow dynamic transformations of the Web pages into more accessible versions needs to be installed on the hosting server and configured appropriately.

Under the light of the above, the option of producing accessible interfaces through filter and transformation tools can be more preferable in cases of existing user interfaces that adhere to established standards for web engineering so that only some low cost adjustments on the mark-up code are further required. But, in addition to this cost, one must consider the price for acquiring the necessary filter / transformation software. Nonetheless, it should also be noted that such tools have the potential to assist developers with no particular background or knowledge in the details of developing sites that are fully accessible, and can potentially save development time.

3.2 Platforms for the creation of web services

In the last few years, a number of software applications have been developed for creating web sites and web-based forms. WebAim provides a review of free, online accessibility tools (WebAIM, 2007). In the next subsections, some of the most popular and successful platforms are discussed, namely Microsoft Office SharePoint Server²², Adobe ColdFusion²³ and eZ Publish²⁴.

²² <http://www.microsoft.com/sharepoint/default.mspx>

²³ <http://www.adobe.com/products/coldfusion/>

²⁴ <http://ez.no/>

3.2.1 Microsoft Office SharePoint Server

Microsoft Office SharePoint Server 2007 (MOSS), commonly abbreviated to MOSS, is the successor to Microsoft Office SharePoint Portal Server 2003. It has a large range of new features not present in the previous version.

MOSS is a licensed enterprise extension to version 3.0 of the no-cost Windows SharePoint Services platform - a component available for Windows Server 2003. Its main strength is enabling information to be organized and aggregated in one central, web-based application. It can be configured to return separate content for Intranet, Extranet and Internet locations. The primary areas of investment that Microsoft has made over the previous version are Excel Services, Infopath Forms Services, the Business Data Catalog, Enterprise Search, web content management, more specialized document management, records management, web 2.0 collaboration functionality like blogs and wikis, delivery of information stored in SharePoint via RSS, Microsoft PowerPoint slide libraries, and the ability to take content and lists offline with Outlook 2007 and Microsoft Access.

The application uses a Microsoft SQL Server back-end for storing data, with Windows SharePoint Services providing the document management functionality. The front-end consists of ASP.NET pages served via Internet Information Services (IIS) on Windows Server 2003. MOSS 2007 requires .NET Framework 3.0 (with Windows Workflow Foundation) to be installed.

MySite is an important feature in MOSS 2007 that enables a user to obtain access to a personalized view of the information that's relevant to them. MySite has a Public view and a Private view. Users are able to throttle the permissions on various pieces of information that are in a MySite, so that only their colleagues, manager, or anyone in the organization can see the information. The Private view of a user's MySite enables to see a number of interesting pieces of information:

- Workspaces - Users can see and access the workspaces to which they have access saving on wasted navigation time.
- MyLinks - A list of personal links that are important to user. As a user is browsing the SharePoint site, they can quickly add a link for a given page to the MyLinks list, by selecting Add Link from a menu in the upper right corner of the page.
- Personalization Sites - Special Sharepoint sites that personalize content based on a users role in the organization can be pinned to the appropriate user's MySite based on their organization role (HR, Facilities, Finanace, etc). Microsoft has released several role-based personalization templates to help people get started with this feature.
- Colleague Tracker - Enables users to track the changes that they have permission to see in their colleague's MySites.
- Outlook email - Web Parts are available for a user's MySite that display their email and calendar information from Exchange.
- Distribution Groups - In the public version of your MySite you can see the distribution groups that you're a member of, and when looking at another user's MySite can see the distribution groups that you have in common with them.

- Standard WSS Site Features - Since a MySite is a WSS site at its core, user's MySites have all of the typical functionality that comes with Windows SharePoint Services (Document Libraries and Lists, Recycle Bin, Version Control, Workflow, etc).
- If the system has the appropriate multi-language packs and templates installed, users can be given the option of creating their MySite in one of the languages available on the system instead of being forced to use the language that governs the more public areas of the SharePoint system. This might be useful in a scenario where a global enterprise is enabling their users in China and Spain to create their MySite in Chinese or Spanish.
- Collaboration with Office 2007Wi. The MOSS 2007 wiki is rather simplistic. It lacks many of the conventions of MediaWiki, but it allows RSS export of content, provides a wysiwyg editor, and is fairly simple to use. As with MediaWiki it produces hyperlinks with a double square bracket.

On an Office SharePoint Server 2007 or Office Forms Server 2007 site, most user interface (UI) elements, such as links, form controls, and buttons, are designed to use Microsoft Active Accessibility (MSAA). MSAA enables people with disabilities to interact with content by using accessibility tools such as screen readers, which are devices that provide a synthesized speech or Braille description of what a blind or low-vision user is unable to see on a computer screen or Web site. However, this tool does not provide compliance with WCAG 1.0 Level A.

3.2.2 Adobe ColdFusion

ColdFusion is an application server and software development framework used for the development of computer software in general, and dynamic web sites in particular. In this regard, ColdFusion is a similar product to Microsoft ASP.NET, Java Enterprise Edition or PHP.

The primary feature of ColdFusion is its associated scripting language, ColdFusion Markup Language (CFML), which compares to JSP, C#, or PHP and resembles HTML in syntax. "ColdFusion" is often used synonymously with "CFML", but it should be noted that there are additional CFML application servers besides ColdFusion, and that ColdFusion supports programming languages other than CFML, such as server-side Actionscript and embedded scripts that can be written in a JavaScript-like language, known as CFScript.

ColdFusion Server includes a subset of its Macromedia Flex 1.5 technology. Its stated purpose is to allow for rich forms in HTML pages using CFML to generate Flash movies. These Flash forms can be used to implement rich internet applications, but with limited efficacy due to the ActionScript restrictions in place on Flash forms by Macromedia.

Flash forms also provide additional widgets for data input, such as date pickers and data grids. In previous versions of ColdFusion, some form validation and additional widgets were available using a combination of Java applets and JavaScript. This option persists for those who do not wish to use Flash, however not all features are supported.

Example:

```
<cfform format="flash" method="post" width="400" height="400">
  <cfinput type="text" name="username" label="Username" required="yes">
  <cfinput type="password" name="password" label="Password" required="yes">
  <cfinput type="submit" name="submit" value="Sign In">
</cfform>
```

ColdFusion also includes some XForms capability, and the ability to "skin" forms using XSLT.

ColdFusion can generate PDF or FlashPaper documents using standard HTML (i.e. no additional coding is needed to generate documents for print). CFML authors simply place HTML and CSS within a pair of cfdocument tags and specify the desired format (FlashPaper or PDF). The generated document can then either be saved to disk or sent to the client's browser.

ColdFusion 8 has now introduced the cfpdf tag which allows for unprecedented control over PDF documents including PDF forms, and merging of PDFs.

According to Adobe,²⁵ the resulted mark-up complies with Priority 1 checkpoints of the WCAG 1.0 (Level A), which does not guaranty the production of a fully accessible web forms.

3.2.3 eZ Publish

eZ Publish is an open source enterprise content management system. It is developed by the Norwegian company eZ Systems and a growing number of users and developers worldwide. eZ Publish is available for free download under the GPL licence, as well as under proprietary licences with commercial support. eZ Publish aims to support the development of professional web applications in PHP.

eZ Publish supports the development of professional, customized web applications. Typical applications range from a personal homepage to a multilingual corporate website including role-based multi-user access, e-commerce functions and online communities.

According to eZ Systems, eZ Publish is used for tens of thousands of web applications of varying type and size worldwide, among them MIT (specifically the controller's office), Vogue magazine, NASA, the US Navy DASN and the Swiss public broadcasting organisation Schweizer Fernsehen. Further examples eZ Publish-based projects are listed on the eZ Publish reference page²⁶.

eZ Publish is managed via a Web browser, thus additional local software is not necessary. It also features a rich text editor that allows formatting content similar to a word processor (e.g. Word). This enables content editing and contribution without HTML skills. Content management can also be done through the eZ Publish frontend.

²⁵ Voluntary Product Accessibility Template:
http://www.adobe.com/resources/accessibility/tools/vpat/coldfusion_8_508.html

²⁶ <http://ez.no/customers/references>

The eZ Publish functional range targets the quick, professional and secure realization of web applications. Functional criteria are (besides standards such as sitemaps, search and printing function):

- A logic for content versioning;
- A media library; and
- Role-based rights management.

Furthermore, custom changes can be made to eZ Publish. For this, the system's architecture provides "Extensions", which are meant to contain individual functions. This allows for the upgrading of the kernel even after customizing new versions. Finally, there are several hundred contributions provided by the community. eZ Systems integrates such contributions into the kernel on a continuous basis. This is done especially to avoid mixed installations of the kernel and custom plugins, which could lead to serious problems (e.g. for migrating an existing installation to new versions of PHP, as such plugins are usually supported unpredictably).

As a LAMP application, eZ Publish is based on PHP. The recommended webserver is Apache. This makes the software independent from the operation system. eZ Publish can be run on Windows as well as on different UNIX derivatives.

One of the strictly applied development principles is a clean implementation of the database abstraction layer, which enables the use of nearly any common database by using drivers, thus rendering changes to the kernel unnecessary. eZ Systems recommends MySQL, but drivers for PostgreSQL, Microsoft SQL Server and Oracle are also available. As eZ Publish supports open standards such as XML and SOAP, it can be flexibly integrated into existing IT infrastructures.

eZ Publish is cluster-ready and enforces the strict separation of information and design through XML storage of all content. This eases media-neutral design in terms of accessibility, e.g. for Braille devices or serving WAP browsers and mobile phones.

eZ Publish produces valid (i.e., proper use of mark-up) web material that complies with Priority 1 checkpoints of the WCAG 1.0 (Level A)

3.3 Authoring tools

3.3.1 Macromedia Dreamweaver MX

Macromedia MX, developed by Adobe, formerly Macromedia, is one of the most popular and powerful web development applications available today. Its main characteristics are:

- Its WYSIWYG (What you see is what you get) interface allows to create a web page without looking at the code. This can be especially valuable for novice web designers.
- The HTML code created in the WYSIWYG interface tends to be cleaner and more compliant with W3C HTML and XHTML specifications - external link than

- code created with other tools. Standards compliant code is usually more accessible more compatible with emerging and assistive technologies.
- Changes can be performed manually, or in the Code view. This feature has really has set Dreamweaver apart from other web development applications.
 - It integrates smoothly with other Macromedia tools, especially Flash.

Dreamweaver does not have functionality to add additional accessibility features to forms. Form label tags, fieldset tags, and legend tags must all be done within the code. It even has a menu function that allows you to add a JavaScript driven, inaccessible jump menu, which uses a drop-down menu item for navigation. In general, Dreamweaver supports both CSS and accessibility with an updated evaluation tool that includes WCAG Priority 2 checkpoints.

3.3.2 Accessible Form Creator (HiSoftware)

The HiSoftware Accessible Form Creator allows to create forms for web sites containing all the additional markup required to make the forms accessible under Section 508 standards and the W3C WCAG 1.0 Priority 1-3 Guidelines.

The HiSoftware Accessible Form Creator is intended to make creating accessible forms easy, regardless of the skill of the developer. It is designed for creating accessible HTML code related to the design, appearance, accessibility and usability for the form. From there, a web developer can then code the back-end functionality of the form, whether CGI programs, FrontPage Extensions or other methods are used to get the form to perform its actions. Please review the documentation for your web editing programs, web server and other web technology you are using in order to complete these "back-end" configurations.

3.4 Web forms development requirements

Based on the analysis of currently available platforms and tools presented in the previous section, this section put forward a set of requirements for the development of universally accessible web forms.

3.4.1 Separating presentation from content

In order to be in compliance with the WCAG 1.0, an author must clearly separate content from presentation. For example, a book title (Inroads) and emphasized text (webpages must be accessible) are both traditionally displayed in italics, i.e., they have the same presentation. But they are different ontologically, since they are different types of things—one is a title, and one is an emphasized word. The two words could both be marked up using an <i> tag, which would italicize them both, but a better solution would be to respect their difference and mark the former with a <cite> tag and the latter with . This difference in markup reflects the difference in content.

Placing an emphasis on accessibility provides developers with a good reason to separate content from presentation. It's easy to imagine screen reading software increasing

loudness when reading text marked with an `` tag, but it's not so clear what you'd want it to do when encountering a new ``. (Does the font change signify change in emphasis? Does it signify a book title? Does it signify a heading? Does it have no logical significance in the document?) With a strong content/presentation distinction in hand, it is much easier to teach students to use a markup language to mark up the logical structure of a document.

The use of XHTML further reinforces this distinction because strict XHTML lacks the stylistic elements and attributes of HTML. The presentation aspects of a document are moved to its associated style sheet. XHTML is simply a reformulation of HTML 4.01 in XML, so a further advantage to this approach is that students are learning an XML application. Finally, since XML is much more strict than HTML, it will be easy for students with a knowledge of XHTML to write (or maintain) HTML documents.

3.4.1.1 Cascaded Style Sheets

CSS benefits accessibility primarily by separating document structure from presentation. Style sheets were designed to allow precise control - outside of mark-up - of character spacing, text alignment, object position on the page, audio and speech output, font characteristics, etc. By separating style from mark-up, authors can simplify and clean up the HTML in their documents, making the documents more accessible at the same time.

CSS allows precise control over spacing, alignment and positioning. Authors can thereby avoid "tag misuse" - the practice of misusing a structural element for its expected stylistic effects. For example, while the `BLOCKQUOTE` and `TABLE` elements in HTML are meant to mark up quotations and tabular data, they are frequently used to create visual effects instead such as indentation and alignment. When specialized browsing software such as a speech synthesizer encounters elements that are misused in this way, the results can be unintelligible to the user.

In addition to preventing element misuse, style sheets can help reduce image misuse. For instance, authors sometimes use 1-pixel invisible images to position content. This not only bloats documents, making them slow to download, but can also confuse software agents looking for alternative text (the "alt" attribute) for these images. CSS positioning properties mean that invisible images are no longer required to control positioning.

CSS provides precise control over font size, color, and style. Some authors have used images to represent text in a particular font when they are uncertain of the availability of the font on the client's machine. Text in images is not accessible to specialized software such as screen readers, nor can it be catalogued by search robots. To remedy this situation, the powerful WebFonts of CSS allows users much greater control of client-side font information. With WebFonts, authors can rely on fallback mechanisms on the client when the author's preferred fonts are not available. Fonts can be substituted with more accuracy, synthesized by client software, and even downloaded from the Web, all according to author specification.

CSS allows users to override author styles. This is very important to users who cannot perceive a page with the author's chosen fonts and color. CSS allows users to view documents with their own preferred fonts, colors, etc. by specifying them in a user style sheet.

CSS provides support for automatically generated numbers, markers, and other content that can help users stay oriented within a document. Long lists, tables, or documents are easier to navigate when numbers or other contextual clues are provided in an accessible manner.

CSS supports aural style sheets, which specify how a document will sound when rendered as speech. Aural style sheets (or "ACSS" for short) allow authors and users to specify the volume of spoken content, background sounds, spatial properties for sound, and a host of other properties that can add effects to synthesized speech analogous to those achieved with styled fonts for visual output.

CSS provides more precise control over the display of alternative content than HTML alone. CSS2 selectors give access to attribute values, often used to provide alternative content. In CSS2, attribute values may be rendered in a document along with an element's primary content.

3.4.1.2 Architectural models

In order to achieve the separation between content and presentation, two major architectural models are available:

- **Three-tier architecture**
- **Model-View-Controller (MVC) model**

Three-tier architecture

A three-tier architecture (see Figure 4) consists of the following:

- **Data access layer**, responsible for storing and communicating data between the database and the application. This layer uses stored procedures for faster retrieval or insertion in the database, reducing the amount of client side processing by looking up data and maintaining key values and internal integrity. Furthermore, using stored procedures, the database server creates for each query a plan that includes all the information required to return the data effectively to the client. This plan is stored in the system's cache, so that it can be reused when needed (Dalton, 1997). Another advantage of the stored procedure is that the database server can create indexes, thus increasing the speed of interaction.
- **Business logic layer**, contains web-services and data transformation functionality providing information to the presentation layer in a meaningful form. Web-services are usually implemented in the C# or JAVA programming language

using XML description files for communicating with the stored procedures. Thus, the business logic layer is totally independent from the implementation of specific parts of the data access layer, and allows the replacement of the data access layer without redeveloping the business layer. For example, the SQL Server database can be replaced by an Oracle database making the appropriate changes only in the XML description files. Another important feature implemented in the business logic layer is the data transformation functionality. This feature is used in order to transform data returned by web services into object types that can be easily manipulated at the presentation level. In order to achieve the transformation, a class was developed to capture the data returned by the web services and subsequently transform the XML data into programming language's objects that can further be elaborated. The main benefit of this procedure is that it provides the ability to rapidly implement the presentation layer of an application without conveying complex structures. The same layer offers the mechanisms to perform all the necessary calculations, and the final data to be displayed are conveyed to the presentation layer.

- **Presentation layer**, responsible for the user interface of the system and incorporates the designs created during the design phase. For the implementation of the presentation layer ASP.NET, PHP, JAVA, ASP, etc is used.

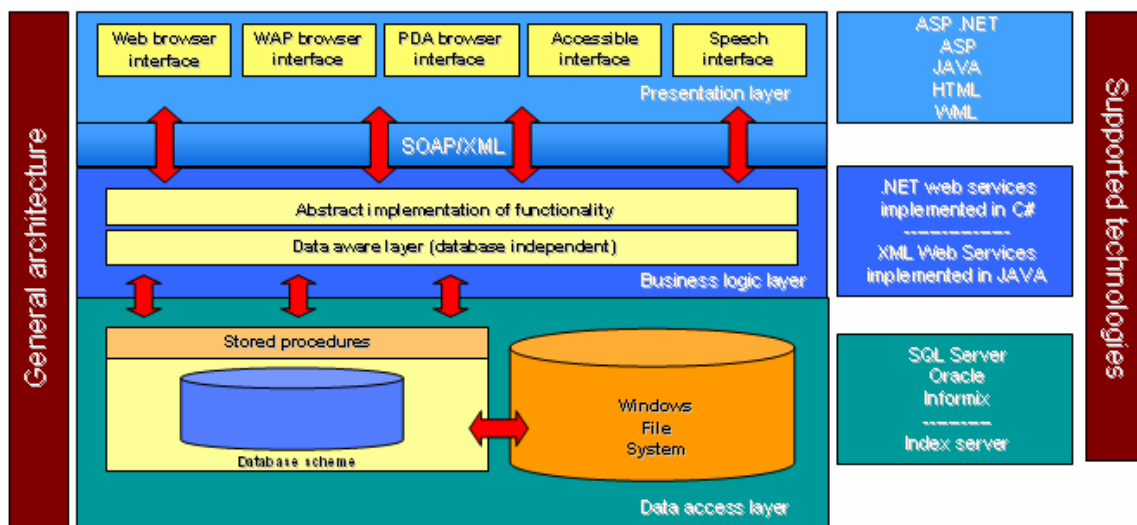


Figure 4: Three-tier architecture

Model-View-Controller (MVC) model

Model-view-controller (MVC) is an architectural pattern used in software engineering. In complex computer applications that present a large amount of data to the user, a developer often wishes to separate data (model) and user interface (view) concerns, so that changes to the user interface will not affect data handling, and that the data can be reorganized without changing the user interface. The model-view-controller solves this problem by decoupling data access and business logic from data presentation and user interaction, by introducing an intermediate component: the controller.

It is common to split an application into separate layers: presentation (UI), domain logic, and data access. In MVC the presentation layer is further separated into [view](#) and [controller](#). MVC encompasses more of the architecture of an application than is typical for a design pattern.

- **Model:** The domain-specific representation of the information on which the application operates. It is a common misconception that the model is another name for the domain layer. Domain logic adds meaning to raw data (e.g., calculating if today is the user's birthday, or the totals, taxes and shipping charges for shopping cart items). Many applications use a persistent storage mechanism (such as a [database](#)) to store data. MVC does not specifically mention the data access layer because it is understood to be underneath or encapsulated by the Model.
- **View:** Renders the model into a form suitable for interaction, typically a [user interface](#) element.
- **Controller:** Processes and responds to events, typically user actions, and may invoke changes on the model.

MVC is often seen in [web applications](#), where the view is the actual [HTML](#) page, and the controller is the code that gathers dynamic data and generates the content within the HTML. Finally the model is represented by the actual content, usually stored in a [database](#) or [XML](#) files.

Though MVC comes in different flavors, control flow generally works as follows:

- The user interacts with the user interface in some way (e.g., presses a button).
- A controller handles the input event from the [user interface](#), often via a registered [handler](#) or [callback](#).
- The controller accesses the model, possibly updating it in a way appropriate to the user's action (e.g., controller updates user's [shopping cart](#)).
- A view uses the model to generate an appropriate user interface (e.g., the view produces a screen listing the shopping cart contents). The view gets its own data from the model. The model has no direct knowledge of the view.
- The user interface waits for further user interactions, which begins the cycle anew.

Comparing the techniques

A fundamental rule in three-tier architecture is the client tier never communicates directly with the data tier; in a three-tier model all communication must pass through the middleware tier. Conceptually the three-tier architecture is linear. This addresses the question of how to pass information between a user and a database. However, the MVC architecture is triangular: the View sends updates to the Controller, the Controller updates the Model, and the View gets updated directly from the Model. This addresses questions of how a user interface manages the components on the screen. (MVC interface components are often used in applications with 3-tier architecture.)

From a historical perspective the three-tier architecture concept emerged in the 1990's from observations of distributed systems (e.g., web applications) where the client, middleware and data tiers ran on physically separate platforms. Whereas MVC comes from the previous decade (by work at [Xerox PARC](#) in the late 1970's and early 1980's) and is based on observations of applications that ran on a single graphical workstation; MVC was applied to distributed applications much later in its history.

3.4.2 Full compliance with W3C Accessibility guidelines

If the user misunderstands how data should be entered, they may not be able to complete the form, or may input data incorrectly, which in turn may affect the result of form interaction. The WCAG guidelines focus primarily on technical accessibility; thus, they can only partially address the highly subjective and contextual issues surrounding optimal usability of a specific digital resource. Despite this, they have become widely accepted as the most authoritative set of guidelines relating to Web accessibility.

Historically, usability and accessibility advocates have met with resistance from Web and software developers involved in graphic design (Cloninger, C., 2002). Without doubt though, there has recently been a noticeable culture shift in the Web design community, where usability, accessibility, and standards compliance are increasingly being seen as important technical requirements and core objectives in Web-design development projects. Demonstrations that W3C technologies such as XHTML and CSS need not hinder aesthetic creativity have encouraged a groundswell among developers who are driven equally by issues such as compatibility with previous browser versions, accessibility/usability, and graphic design.

Some improvements in accessibility have been developed as “add-ons.” For example, IBM has produced Web Adaptation Technology (WAT) to help older users overcome many of the barriers presented by inappropriately designed Web content (Richards, J. et al, 2003). WAT works within Internet Explorer to let the user alter characteristics of Web pages to make them more easily accessible. A central advantage of this system is that it places minimal responsibility on the developer of the content but instead takes existing Web content and allows people to alter it so they can read it more easily; it also allows people to use standard computer equipment to enable useful changes to be made easily and immediately.

However, the existence of tools such as WAT should not lead designers to conclude that the accessibility of Web pages is no longer their responsibility. Indeed, the more accessible a Web page is, the easier it will be for assistive technologies and adaptation tools to render the page according to users' needs. Conversely, the more inaccessible a page is, the harder it will be to adapt, meaning that on those occasions when users need it most, adaptation is least likely to succeed. Therefore, although the continued development of assistive technologies and adaptation tools is necessary and beneficial,

designers have a responsibility to ensure that any Web content they produce is optimally accessible in order to maximize the success of these tools.

To develop mainly accessible web forms that are appropriate for all users and devices (if possible), designers need to know the users and their needs, as well as devices' technical characteristics. Although there are various ways of providing such information to designers, there are many misconceptions about Web usability and accessibility that also have to be considered and addressed. For example, there is a frequent apparent assumption among many designers that the use of automated accessibility checking tools, such as Bobby, provides a sufficient basis for discovering and rectifying accessibility barriers in a site as far as it concerns a typical desktop pc. Whereas such automated tools can be useful, they can only check for a limited range of access barriers, and there is often inconsistency between automated tools in reporting accessibility barriers. Once these barriers are discovered, it is still the responsibility of the designer to resolve the problem in the most appropriate way, and a lack of understanding of the issues will likely lead to a sub-optimal solution.

Furthermore, whereas automated tools can create accessible form templates and check them against simple conditions, such as the absence of text descriptions as an alternative for images, they are incapable of assessing in more depth the impact on accessibility of specific interactive form features.

Although the WCAG guidelines are a constructive and necessary part of the drive toward a more usable Web, clearly, the mere availability of guidelines is not enough to ensure suitably accessible and usable web forms, nor it seems are legal imperatives to adhere to them. Even well-written guidelines can be inherently difficult to implement, as they attempt to summarize some very complicated issues into manageable and memorable sets of instructions. To be of any value, guidelines must present a generalized rule that can be followed in a variety of scenarios. However, the fine details of each design scenario, which ought to dictate the most appropriate solution in each case, are often lost in this process of summarization. A designer is unlikely to derive the best solution for a particular scenario for an interaction form from a generalized guideline in isolation. At the very least it is necessary to investigate any additional information provided alongside the guideline, including sample scenarios and solutions for each device in use.

3.4.3 Device independence

Recent years have witnessed the explosive growth of mobile handheld devices, such as cell phones and personal digital assistants (PDAs) in. Many wireless applications have been developed for those devices, including daily news update, classified advertising, tourist guide, wireless Web portals, and m-commerce applications. The ability to communicate from virtually anywhere and the convergence of Web and wireless technologies offer an unprecedented level of flexibility and convenience, particularly for ubiquitous information access through mobile devices.

However, the unique features of wireless networks (for example, low bandwidth and unreliability) and mobile devices (for example, small screen size, and low memory and processing capability), as well as the mobility of users, present challenges for taking advantage of the convenience of mobile devices for information access. For example, most Web content is designed and optimized for desktop and broadband clients. Web content is poorly suited for mobile devices (Zhang and Adipat, 2005); users who carry those devices around may need different information under different contexts. Therefore, how to adapt content to meet the needs of users, fit the characteristics of individual mobile devices, and adjust to dynamic context becomes important.

The World Wide Web Consortium (W3C) defines content adaptation as a process of selection, generation, or modification that produces one or more perceivable units in response to a requested uniform resource. In 2005, W3C announced the launch of a new Mobile Web Initiative (MWI), aiming to address the interoperability and usability problems in order to make “Web access from a mobile device as simple, easy, and convenient as Web access from a desktop device”.

3.4.3.1 Selective Content Delivery

There are two types of content delivery to mobile device users: pull and push. In the pull model, a mobile client sends an information query to a server and the server returns the relevant content back. In the push model, a server automatically delivers content to mobile clients via a push proxy without receiving users’ requests. The push model is very useful in mobile applications such as news alert services, mobile advertising, and real-time traffic updates.

To prevent users from wading through possibly irrelevant content to find a single item of information of interest, selecting and delivering content relevant to users’ interest is essential. This can be achieved by employing user profiles. A push proxy assesses the similarity between a Web page and a user profile to determine if it might be of interest to the user. Furthermore, a Web page can be partitioned into several blocks (for example, sports, business, health, advertisement, among others). Unnecessary or irrelevant blocks can be removed from the original page. A user profile in a mobile information system can consist of the following information about a user’s interests and preferences:

- Demographic information;
- Information interests (for example, represented by keywords);
- Browsing history, such as recently visited Web sites, the time of last access, and visiting frequencies;
- Content presentation preferences, which will be discussed later;
- Quality of Service (QoS) preferences; and
- Access privilege indicating what information a user can access;

User interests can be inferred and updated based on users’ explicit and/or implicit feedback (for example, browsing behaviour on mobile devices) (Billsus, Pazzani and

Chen, 2000). In a mobile computing environment, user profiles can be stored at different locations, including a single centralized server as the only profile server; different profile servers with duplicated or unduplicated user profiles; or local mobile devices.

The push model is generally used for multicasting information to a group of users. It, coupled with user profiles and intelligent agents, can also be used for content prefetching for individual users to minimize the transmission delay (sometimes the required information is not even accessible when it is needed due to a disconnected wireless network). The idea of prefetching is that a system delivers certain content to a local mobile device that will likely be accessed by the user soon.

However, the prefetching decision is contingent upon different network conditions and the likelihood that the user will access this content shortly. Prefetching operations consume the already limited bandwidth of wireless networks and limited storage of mobile devices and users often must pay for the service. It will be particularly expensive if the user does not view the prefetched content eventually.

Therefore, it is vital not only to determine what content should be prefetched and when, but also to build utility models to analyze the trade-off between the potential benefit of prefetching and cost under current circumstances (for example, network conditions).

For instance, (Jiang and Kleinrock, 1998) proposes a prefetching decision scheme that utilizes the access probabilities and prefetch thresholds. The access probabilities indicate how likely certain content will be requested by a user based on the user's previous access history, while the prefetch thresholds computed based on system and network conditions, cost of bandwidth, and transmission time determine whether the performance may be improved by prefetching certain content. The content delivery can also be context-aware.

The term context refers to information that characterizes a situation related to the interaction between users, mobile applications, and the surrounding environment, such as information about a person, a place, or an object. Context information gathered from various sensors, networks, devices, user profiles, and other sources can trigger the content adaptation in order to cope with the dynamic environment.

For example, in a museum, mobile device users should only receive corresponding introductory information when they enter different exhibition rooms. In the GUIDE project (a context-aware electronic tour guide) (Cheverst et al 2000), visitors to a city can access context-aware information and services using their handheld GUIDE units. The information presented to visitors is tailored based on the visitor's interest, current physical location, and attractions already visited.

3.4.3.2 Adaptive Content Presentation

While many of today's mobile devices already feature Web browsers, browsing the Web on a mobile device has not become as convenient as expected. Mobile handheld devices are quite restrictive on the format and length of the received content. They typically display less than 20 lines of text on the screen; they may run different operating systems

and support different markup languages. There is information loss when a Web site uses a presentational mode that a mobile device does not support. Currently, most Web pages are designed only for display on desktop computers, making direct presentation of those pages on small devices aesthetically unpleasant, difficult to navigate, and even completely illegible.

Merely squeezing original Web content into small screens may not work. Traditionally, device-specific authoring and multiple-device authoring approaches were used to adapt Web content presentation for effective display on mobile devices. They either authorized a set of Web pages that were designed and formatted up-front according to a predefined guideline for a specific device, or generated a set of different versions of the same Web page to cover a number of identified target devices. Obviously, those approaches are device dependent and inflexible.

Therefore, automatic re-authoring and client-side navigation approaches seem more effective. Such approaches involve developing software to re-authorize a Web page in real time through a series of transformations, including layout change and content format reconfigurations, so that the page can be effectively displayed on a device and the user can interactively navigate the page.

Similar to a user profile, a device profile can be used to support device-related adaptation. It specifies the MIME media types and physical characteristics of a device including color depth, screen size, memory size, operating system, as well as supported markup languages. The generic Composite Capabilities/Preference Profiles (CC/PP) framework²⁷ provides a mechanism through which a mobile user agent, such as a browser, can transmit information about the mobile device. A user agent profile based on the CC/PP framework includes device hardware and software characteristics, information about the network to which the device is connected, and other attributes (Laakko and Hiltunen, 2005).

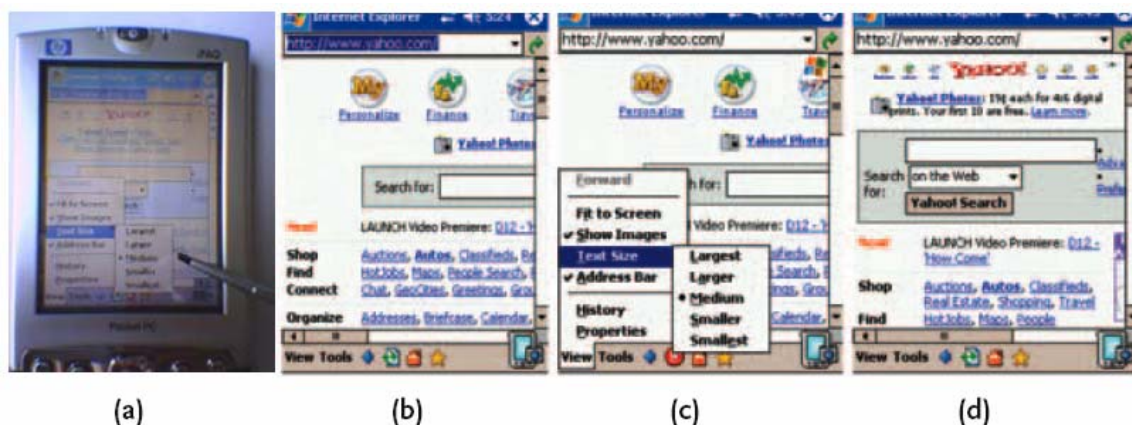


Figure 5: Adaptive display of Web content with reconfigurations.

²⁷

www.w3.org/2001/di/

Figure 5 shows a few screen shots of a Yahoo! Web page displayed on a HP iPAQ h4355 Pocket PC with a 3.5” transfective screen (Figure 9(a)). It has integrated Bluetooth and WLAN 802.11b for wireless communication and access to the Internet. Figure 9(b) shows the page displayed on that device without any adaptation. A user must use both vertical and horizontal scroll bars in order to see the entire content, which can be cumbersome and may cause loss of context.

There are some simple ways to adapt content presentation based on the features of a device and user’s preference to improve its navigation. For example, as shown in Figure 9(c), a “Fit to Screen” function can be activated to produce a scaled-down version (Figure 9(d)) of the original Web page to better fit the width of a device through syntactic translation (without removing any content from the original page); images embedded in a Web page and the Internet address bar can be removed to reduce the page size and increase display space; and the font size of textual content can be adjusted by the user. Such adaptation preferences, once defined, can be stored in the user profile and automatically used for future adaptation.

Various summarization and visualization techniques have also been used in content presentation adaptation for mobile devices (Lank and Phan, 2004, Yang and Wang, 2003). Web pages often contain many sections and not all of them are of interest to users. It is argued that the document summarization on handheld devices should make use of “hierarchical display,” which is suitable for navigation of a large document and ideal for small area display (Yang and Wang, 2003). A Web page can be organized into a multi-level hierarchy with a thumbnail representation at the upper level for providing a global view and an index to a set of sub-pages at the lower level for details (Adipat and Zhang, 2005). A user can select a desired portion of a Web page to zoom in for further details.

A Mobile Web system has been developed that automatically adapts the presentation of a Web page on a mobile device based on a three-tier architecture (Buyukkokten, Garcia-Molina, and Paepcke, 2001). In addition to adaptation functions that users can activate, as shown in Figure 9(c), Mobile Web also features DOM (Document Object Model)-tree navigation, summarization, personalized topic spotting, and fisheye view, aiming to make Web browsing and information seeking on handheld devices more effective.

By following the principle of “overview first, then zoom in for details,” Mobile Web first automatically parses a Web page and generates a DOM-tree hierarchical view of its content.

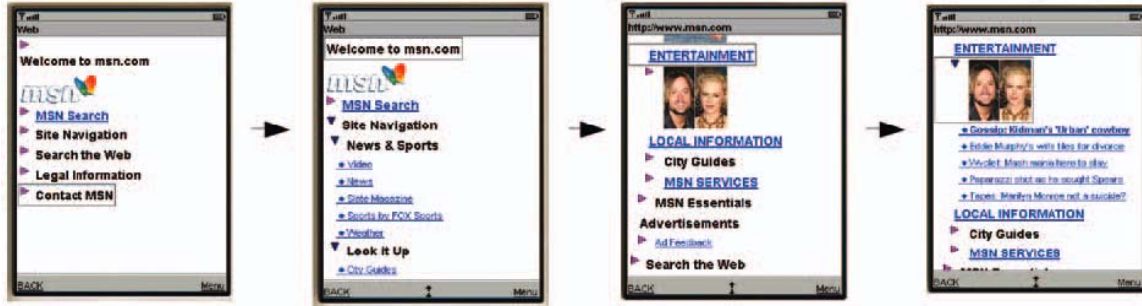


Figure 6: A DOM-tree navigation of the MSN home page via Mobile Web.

The main task of generating a DOM-tree is to identify content blocks and their relationships in a Web page and to extract labels that can represent those content blocks. A user can either expand any branch of the tree, or view the summary of a selected section that is dynamically generated based on heuristic rules. Based on the summary, the user can determine if he or she wants to see the full content of that section. Mobile Web stores users' personal interest as keywords in their user profiles. When a Web page is displayed, those specified keywords appearing in the page will be automatically highlighted with different colors for easy spotting, and the page display is automatically anchored by those keyword occurrences (see Figure 6).

Focus and context visualization techniques are designed to integrate a high-detail focus area and a low-detail periphery in order to maximize display space. Via a customized fisheye view component in Mobile Web, users can navigate content through a focus context view, with the focal content displayed in a larger font size, while peripheral content is still shown in the surrounding area with reduced granularity of detail, such as a smaller font size (see Figure 7). Environmental context can also be used to adapt the content presentation. For example, when the light level is low or the device is on the move (for example, the user is walking), then the font size of textual content can be automatically enlarged to ease reading.



Figure 7: Analysis Results

3.4.4 Personalisation capabilities

Building personalized Web applications, i.e., those applications that are responsive to the individual needs of each user, is a challenging task. It involves a myriad of different technologies that range from simple database views to software agents and collaborative filtering algorithms.

Personalization has become hype in areas such as electronic commerce, and hundreds of applications claim to be fully customizable to different user profiles or individuals. The number of possible personalization variants seems countless.

As with other Web features, a great variety of technologies and systems have been developed and are available in the market (Communications of the ACM, 2000), but little or no attention has been paid to the process of modeling and designing personalized Web applications (an interesting exception can be found in Ceri, Fraternali, Paraboschi, 1999).

3.4.4.1 Scenarios of Personalization

Although it seems impossible to classify all the existing approaches to personalization, using a simple conceptual framework allows to show the main differences between most of them. In this context, Web applications are considered as hypermedia applications (Schwabe, Rossi, 1998) in the sense that users navigate a hypermedia information space composed of nodes connected by links.

The main difference between a “traditional” static hypermedia application and most Web applications is that the latter may involve some business logic (application functionality). In addition, users may alter information while navigating – adding products to a cart for example. There are thus two approaches to characterize personalization: analyzing how the underlying application logic may change for each user or analyzing what may change in the information space the user perceives.

Each of them is discussed in a separate sub-section below. However, both kinds of variability can be, and usually are, combined.

Personalizing content and links in a Web application is by far the most popular way of individual customization currently found in the Web, with many different variants. For instance, recommendations in Amazon.com are based on the history of the user; links to specific Purchase groups are built from user personal data.

Many Web sites allow the user to select which contents he wants to see from a repertoire of options (most of them also allow customizing the interface lay-out as well). For example, in my.yahoo.com there are two levels of personalization: first, which modules the user will get in his site (e.g., Weather, Headlines, Financial, etc) and then which information he wants to see within each module (cities, types of news, particular stock quotations, etc).

3.4.4.2 Link Personalization

This strategy involves selecting the links that are more relevant to the user, changing the original navigation space by reducing or improving the relationships between nodes. E-commerce applications use link personalization to recommend items based on the clients buying history or some categorization of clients based on ratings and opinions. Users who give similar ratings to similar objects are presumed to have similar tastes, so when a user seeks recommendations about products, the site suggests those that are most popular for his class, or those that best correlate with the given product for that class.

Link personalization is widely used in www.amazon.com (see Figure 8) to link the home page with recommendations, new releases, shopping groups, etc. that are personalized. Amazon.com has taken this approach to an extreme by building a “New for you” home page and presenting it to each user, with those new products in which he may be interested.

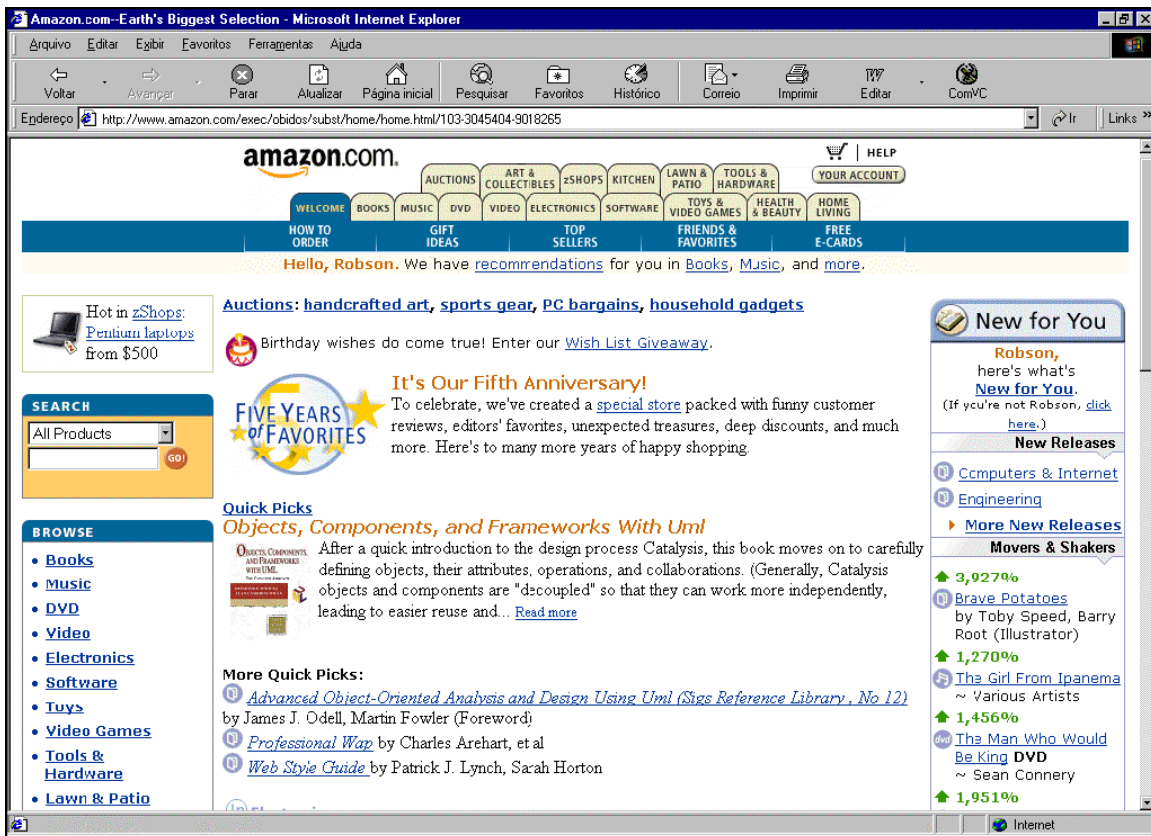


Figure 8: Using Link personalization in www.amazon.com

3.4.4.3 Content Personalization

Content is personalized when nodes (pages) present different information to different users. The difference with link customization is subtle since when links are personalized, part of the contents (the link anchors) present different information. Therefore, content personalization refers to cases where substantive information in a node is personalized, other than link anchors.

Content personalization can be further classified into two types: node structure customization and node content customization. Structure personalization usually appears in those sites that filter the information that is relevant for the user, showing only sections and details in which the user may be interested. The user may explicitly indicate his preferences, or it may be inferred (semi-) automatically from his profile or from his navigation activity. For example, in my.yahoo.com or in www.mycnn.com users choose a set of “modules” (from a large set including weather, news, music, etc...) and further personalize those modules choosing a set of attributes of the module to be perceived. Some “automatic” customization may occur by using the zip code of the user, for instance to select which sport events he may be interested in. The approach followed in these applications is that the user should be able to “build” his own page; even layout may be customized. Figure 9 shows an example of structure customization in my.yahoo.com.



Figure 9: Structure customization in my.yahoo.com

WAP portals can be improved with the same approach. In the Infospace application (www.infospace.com), the user can customize the content and the content provider, making a kind of content syndication. In this way each customer navigates only through the information he desires, improving the usability of the site. Figure 10 shows the customization and its result for the WAP phone.

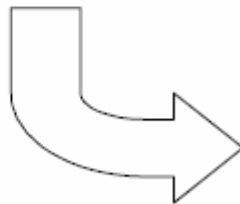


Figure 10: Personalization in WAP Portals

Applications in which different user roles have different access rights or authorizations provide another good example of structure customization. For example suppose an academic application where teachers and students have different tasks to perform; teachers need to access their class schedule to update its contents and students have to access the classes that are available for enrolment, depending on their GPA.

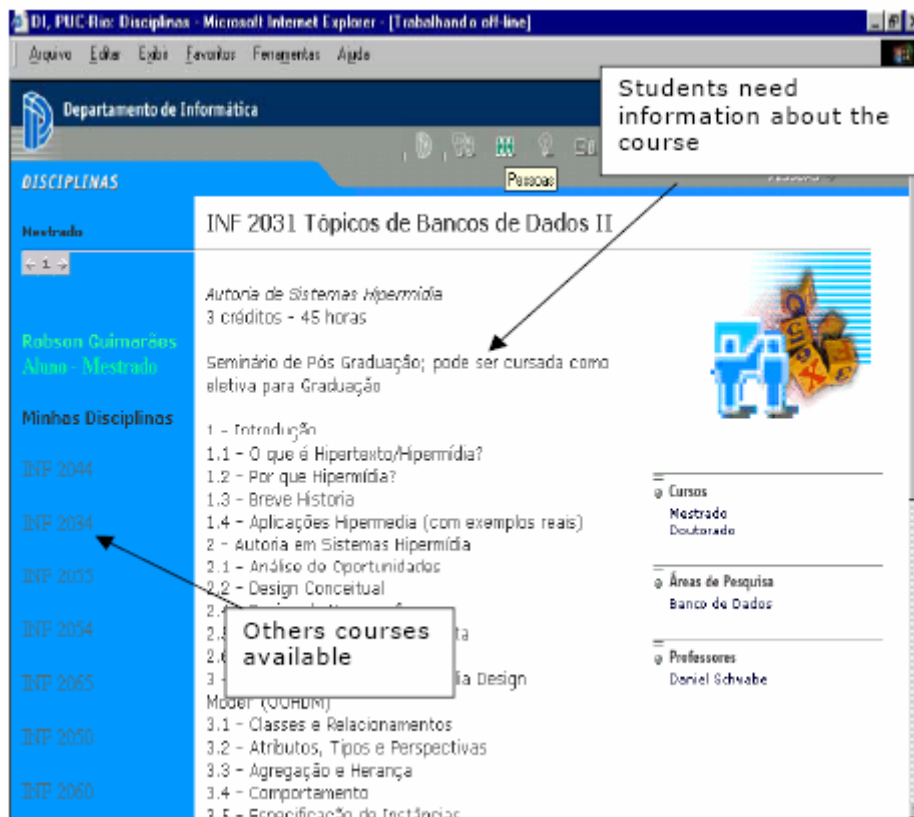
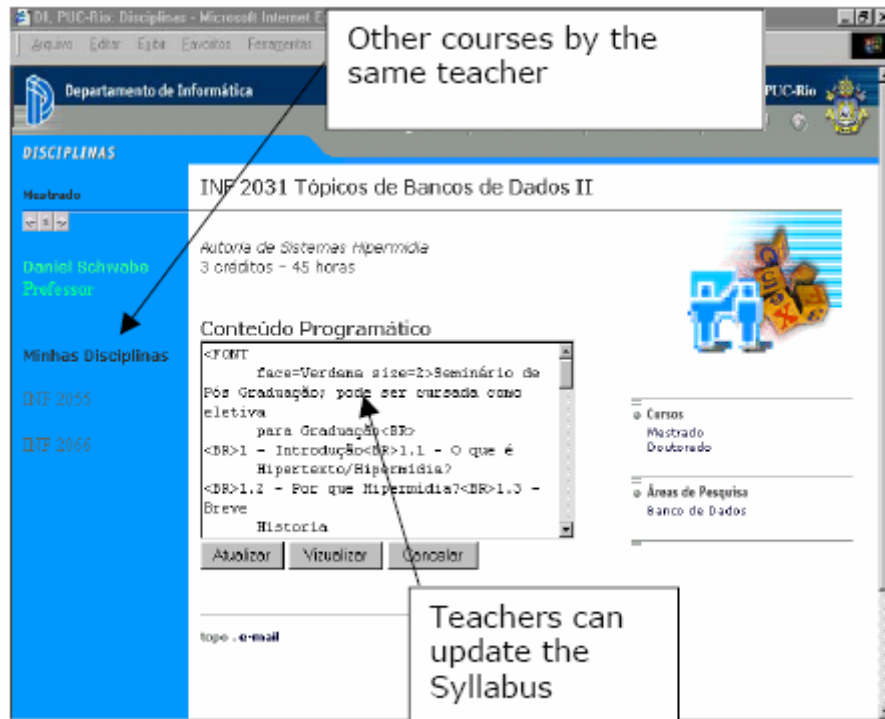


Figure 11: Structure Customization according to users' roles

When a teacher accesses a class node, he can update the class information (e.g., the syllabus), so it is important to make the update button available for the classes for which he is responsible. On the other hand, the student needs to access the syllabus, the course location, the course program, and of course, he cannot modify the site (see Figure 11).

Another difference is the links to related information, in each case. For the teacher it is relevant to provide links to “other courses he teaches”, whereas for the student it is relevant to provide links to other “courses that he may take”.

Node content personalization occurs when different users perceive different values for the same node attribute; this kind of content personalization is finer grained than structure personalization. A good example can be found in online stores that give customers special discounts according to their buying history (in this case the attribute price of item is personalized). There are many good examples of node content personalization in intranet applications, where employees’ role and needs determine the tailoring of information they see.

For example in the ATL (a mobile phone company in Rio de Janeiro) intranet, different sales channels receive different, customized information about business procedures. When a call center attendant looks up information about phone repairs, he will receive the address of a repair center; when a repair center employee looks up information for the same procedure, he will receive repair instructions for the phone, as shown in Figure 12. This type of personalization must be designed from the beginning, capturing the personalization rules for the different user groups that are identified.

Call Center view

Internet Corporativa - Microsoft Internet Explorer

ATL

Para descartá-las use a fragmentadora de papel

Atualize-se

- Clipping
- Notícias ATL
- Agência Estado

Áreas / Informações

Serviços

KM

Serviços SAN

Aparelho com defeito
Passo a Passo(lojas)

1 - MENOS DE 72 HORAS

- encaminhar o cliente a uma loja ATL (exceto DOWNTOWN por não possuir assistência técnica) , com **NOTA-FISCAL de compra** , para análise do defeito do aparelho.

2 - MAIS DE 72 HORAS (

- encaminhar o cliente a uma loja ATL (exceto DOWNTOWN) ou a qualquer assistência técnica autorizada, com a **NOTA fiscal de compra** e a **GARANTIA** para manutenção do aparelho. Consultar a lista de assistência técnica autorizada.

3 - CLIENTE SEM A NOTA FISCAL DE COMPRA / GARANTIA DO APARELHO

- cliente SEM a garantia / nota fiscal não poderá ser atendido, nem pela loja ATL, nem pelas Assinências Técnicas. Caso o cliente não tenha mais a nota fiscal do aparelho, solicitar que peça uma cópia na loja que adquiriu o aparelho.

APARELHO CANCELADO OU COM CADASTRO INCORRETO (acerto do cadastro)

Mostre aqui links e subtemas de acordo com o acesso ao site ou quando não for:

Inicio | Inadara - Ni... | Nicollinea... | ATL - Algar... | Oracle SQL... | Explorando... | Intrenet... | Microsoft W... | 21/03

Instructions to repair a phone

Repair Center view

Internet Corporativa - Microsoft Internet Explorer

ATL

Informação é Patrimônio!

Atualize-se

- Clipping
- Notícias ATL
- Agência Estado

Áreas / Informações

Serviços

KM

Serviços SAN

Aparelho com defeito
Passo a Passo(SAC)

1 - Encaminhar a uma loja ATL ou revendedor solicitando que leve a nota fiscal e o contrato do aparelho.

Instructions to repair a phone

Figure 12: Node content personalization in the ATL intranet

3.4.4.4 User Profiling

In order to personalize a Web site, the system should be able to distinguish between different users or groups of users. This process is called user profiling and its objective is the creation of an information base that contains the preferences, characteristics, and activities of the users. In the Web domain and especially in e-commerce, user profiling has been developed significantly because Internet technologies provide easier means of collecting information about the users of a Web site, which in the case of e-business sites are potential customers.

A user profile can be either static, when the information it contains is never or rarely altered (e.g., demographic information), or dynamic when the user profile's data change frequently. Such information is obtained either explicitly, using online registration forms and questionnaires resulting in static user profiles, or implicitly, by recording the navigational behaviour and/or the preferences of each user, resulting in dynamic user profiles. In the latter case, there are two further options: either regarding each user as a member of a group and creating aggregate user profiles, or addressing any changes to each user individually.

When addressing the users as a group, the method used is the creation of aggregate user profiles based on rules and patterns extracted by applying Web usage mining techniques to Web server logs. Using this knowledge, the Web site can be appropriately customized.

A description of several methods for implicit and explicit collection of user profile data is provided below.

A way of uniquely identifying a visitor through a session is by using cookies. W3C (Web characterization terminology & definitions) defines cookie as "the data sent by a Web server to a Web client, stored locally by the client and sent back to the server on subsequent requests." In other words, a cookie is simply an HTTP header that consists of a text-only string, which is inserted into the memory of a browser. It is used to uniquely identify a user during Web interactions within a site and contains data parameters that allow the remote HTML server to keep a record of the user identity, and what actions he takes at the remote Web site.

The contents of a cookie file depend on the Web site that is being visited. In general, information about the visitor's identification is stored, along with password information. Additional information such as credit card details, if one is used during a transaction, as well as details concerning the visitor's activities at the Web site, for example, which pages were visited, which purchases were made, or which advertisements were selected, can also be included. Often, cookies point back to more detailed customer information stored at the Web server.

Another way of uniquely identifying users through a Web transaction is by using identd, an identification protocol specified in RFC 1413 (Identification Protocol) that provides a means to determine the identity of a user of a particular TCP connection. Given a TCP port number pair, it returns a character string, which identifies the owner of that

connection (the client) on the Web server's system. Finally, a user can be identified making the assumption that each IP corresponds to one user. In some cases, IP addresses are resolved into domain names that are registered to a person or a company, thus more specific information is gathered.

As already mentioned, user profiling information can be explicitly obtained by using online registration forms requesting information about the visitor, such as name, age, sex, likes, and dislikes. Such information is stored in a database, and each time the user logs on the site, it is retrieved and updated according to the visitor's browsing and purchasing behaviour.

All of the aforementioned techniques for profiling users have certain drawbacks. First of all, in the case where a system depends on cookies for gathering user information, there exists the possibility of the user having turned off cookie support on his browser. Other problems that may occur when using cookies technology are the fact that because a cookie file is stored locally in the user's computer, the user might delete it and when she revisits a Web site will be regarded as a new visitor.

Furthermore, if no additional information is provided (e.g., some logon id), there occurs an identification problem if more than one user browses the Web using the same computer.

A similar problem occurs when using identd, inasmuch as the client should be configured in a mode that permits plaintext transfer of ids. A potential problem in identifying users using IP address resolving, is that in most cases this address is that of the ISP, and that does not suffice for specifying the user's location. On the other hand, when gathering user information through registration forms or questionnaires, many users submit false information about themselves and their interests resulting in the creation of misleading profiles.

3.5 Requirements for tools supporting the development of universally accessible forms

Based on the discussion in the previous sections, this section puts forward a set of overall requirements for the development of supporting tools which allow producing fully accessible and adaptable web forms.

A tool that can help the development of a universally accessible web form must support the creation of alternative views according to user preferences and the technical specification of device-browser combination. With this in mind, the first requirement for such tools is that presentation and content in the created forms must be well separated.

Bearing in mind that web forms must be accessible to users regardless of disability, tools must take steps to ensure conformance to existing or prospective mark-up standards (e.g., HTML 4.01, CSS 1.0, HTML 5.0) and web accessibility standards (e.g., WCAG 1.0 Level AAA, WCAG 2.0 working draft), creating form elements with correct labelling in

prior. Since the Web is both a means of receiving information and communicating information, it is important that both the Web form produced and the tool itself be accessible.

In addition to the advancement of mark-up technologies, server-side adaptation offers maximum author control over the delivered content, including the ability to radically change the content amount and its styling, navigation, and layout. In order to produce the most appropriate adaptation, however, the server must have sufficient information about the delivery context, including the delivery device’s capabilities.

From a developer’s point of view, such tools must be easy to use and should not require technical background regarding web accessibility. However, web developer should be able to preview alternative instances of the resulting form and make proper adjustments if needed. To facilitate use, tools should be web-based and easily available, as well as usable.

If web developers must rely on tools to do almost all the work concerning the construction of an accessible web form, the tools they use should be designed to generate standards-compliant code by default. The requirements summarised in Table 1 can be drawn from the discussion above:

Table 1: Supporting tools requirements

| | Requirement |
|---------------------|----------------------------------------------------------------------------|
| Accessibility | Compliance with all Priorities 1, 2 and 3 of WCAG 1.0 |
| Validity | Valid mark-up (W3C Specifications) |
| Availability | Web – based |
| Device independence | Alternative views for other devices |
| Extensibility | Importing interaction resources offered by different interaction platforms |
| Orthogonality | Making use of business logic produced by external software tools |

4 Accessible Online Tools supporting the accessibility of Web elements

As previously documented, forms can present problems for Web users with vision or mobility impairment, as well as for people with cognitive or learning disabilities. It is very easy for someone with impaired vision who relies on an assistive output device such as a screen reader, talking browser or Braille display, to get lost in a form. For these technologies to work effectively, the devices need to be able to associate a form label (request or prompt) with the correct form control, such as a text field or checkbox.

Different devices have different capabilities of user interaction and presentation, and most services cannot adapt their user interfaces to these differences. This means that users often have to use different versions of a service from different providers to access the same functionality. The main approach to making services accessible from multiple devices today is versioning. However, with many different versions of services, development and maintenance work gets very cumbersome, and it is difficult to keep the consistency between different versions.

Another popular method is to use Web user interfaces, since most devices run a Web browser. However, adaptations are still needed, for example, translation between mark-up languages and layout changes for small screens. It is also difficult to take advantage of device-specific features and to control how user interfaces will be presented to end users. Thus, we need new and robust methods for developing services that can adapt to different devices.

This Chapter describes two tools which have been developed to comply with the web forms development requirements discussed in the previous section, as well as with the supporting tools requirements. Following a brief description of the Web Harmonia platform, the design and implementation of the two tools are reported.

4.1 The Web Harmonia Platform

The vast majority of the available accessibility guidelines are formulated either as general design principles, or as low level and platform specific recommendations. They are typically based on past experiences and best practice, while experimental evidence is typically rare (Casali, 1995). For example, accessibility guidelines developed by W3C-WAI mainly focus on three aspects, namely page authoring, user agents, and the structure and presentation of Web documents. By implication, such guidelines without the accompaniment tools to support them do not fully support the accessibility of the Web elements. On the other hand, the proliferation of interaction platforms and their continuous growth (e.g., HTML, XHTML, CSS, XML, DHTML), necessitate an account of key requirements that should be preserved, if these developments and future ones are to comply with the broad accessibility objectives. Moreover, such guidelines offer limited guidance on the process of integrating accessibility into design and development activities.

Currently, web authors typically design Web applications for the desktop and screen. To exert maximum control over the final appearance, they often base Web page layouts on tables, specified using absolute pixel positioning. Because adapting such a site to address different accessibility requirements or a small display's design constrains is effectively impossible, authors in most cases create a parallel site to accommodate these devices. As the variety of Web-connected devices increases, however, creating a separate site for each kind of device is both economically and administratively impractical.

Under a Universal Access perspective, given the diversity and complexity of the factors underlying human interaction with technology, no single design perspective is likely to be adequate for all potential users or computer-mediated human activities (Stephanidis and Savidis, 2001). Instead, designing for Universal Access requires a conscious effort and commitment to analytical and exploratory design, as well as a multidisciplinary effort. For that reason, Universal Access designers need to analyse trade-offs between multiple conflicting design criteria in order to satisfy an increasing range of requirements placed by traditional user requirements and utilisation of new technology. Recent approaches to HCI design under a Universal Access perspective have emphasised two main contributing dimensions towards this end (Stephanidis and Akoumianakis, 2003):

- understanding how interactive tasks are carried out by different users, across different interaction platforms and diverse contexts of use, and
- devising suitable artifacts for each relevant task execution context

The first of these two dimensions implies a user- and context-oriented focus, while the second implies an adaptation-oriented view of interactive artifacts and suitable means for defining and structuring adaptations. As a small contribution towards this goal, a platform is currently being elaborated, named Web-Harmonia (Basdekis et al., 2007), which, along with its supporting tools, facilitates the actual production of universally accessible web interfaces in a cost-effective and standard-compatible way. This undergoing work has the potential to contribute to the wider adoption of design practices that takes into account user and context diversity in a Universal Access perspective. Moreover, it has the potential to deliver web material (e.g. interfaces & content) capable of run-time adaptation behaviour. The tools developed in this thesis constitute supporting tools of Web-Harmonia.

Recent research illustrates that there is a clear trend for using generic approaches to define user interfaces such as XUL²⁸, XIML (eXtensible Interface Markup Language) (Puerta, Eisenstein, 2001), UIML (User Interface Markup Language), RDL/TT and VTT UI Toolkit (Plomp and Mayora-Ibarra, 2002). Some of the most consolidated options are XIML and UIML. In both cases, there is the proposal of frameworks for writing generic widget descriptions, allowing the possibility to develop or incorporate new widgets.

The Web-Harmonia platform offers the possibility to develop a device independent version of mark-up. To accomplish this, the platform uses generic tags, similar to HTML tags, to describe presentation elements in a device-independent way by Server-side adaptation (server selects the appropriate mark-up and style sheet based on the delivery

²⁸ XUL, XML User Interface Language <http://www.xulplanet.com>

context). In addition, access features can be personalised according to user preferences. The Mark-up generator of Web-Harmonia produces “on the fly” the most appropriate mark-up (e.g. HTML, XHTML, cHTML, any future version mark-up) according to browser capabilities as well as to user profile and presenting features that enrich accessibility. The general architecture of Web Harmonia is illustrated in the following image (Figure 13: General architecture of Web Harmonia).

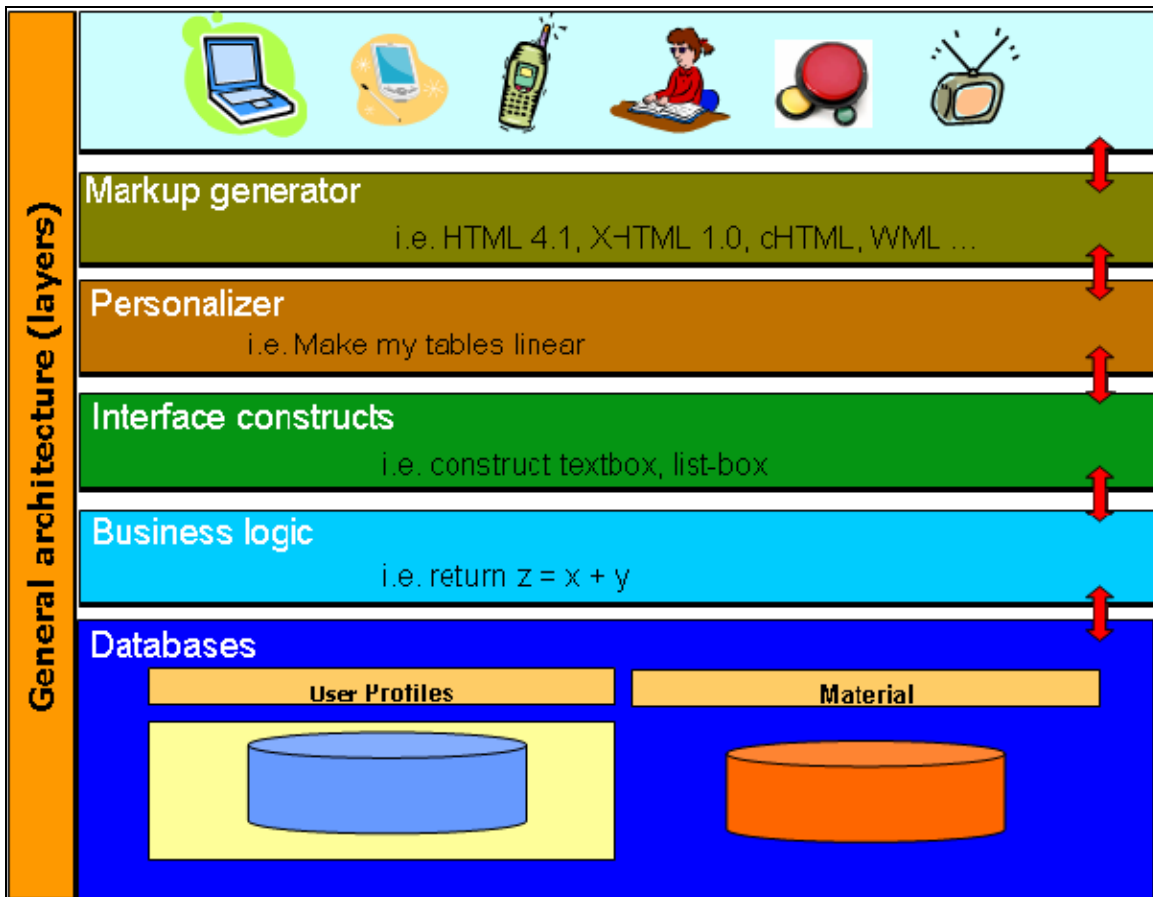


Figure 13: General architecture of Web Harmonia

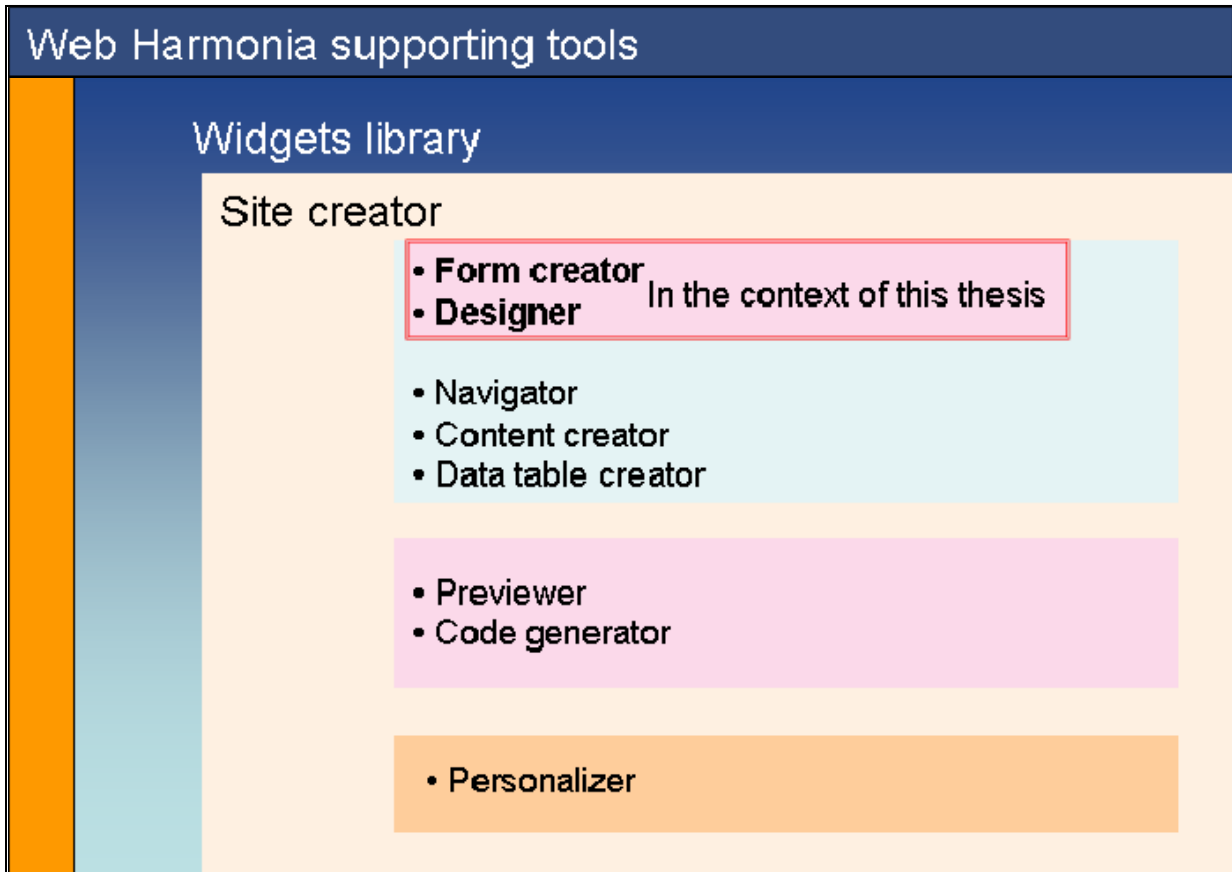


Figure 14: Supporting tools of the Web Harmonia platform

Designer and Form Creator, the two online tools developed in the context of this thesis, utilise the aforementioned platform (see Figure 14). Without requiring any technical background regarding web accessibility issues, these tools are aimed to simplify the creation of fully accessible and adaptive web forms. This makes it possible to develop web services once, and automatically tailor their user interfaces to different user needs and different devices.

4.2 Requirements analysis

In the requirements analysis phase of the developed tools, the model proposed by Bosson and Svensson (2001) was followed. This model, described in detail in Fransson, Bosson, and Svensson (2003), is mainly divided into five phases: *context analysis*, *system goals specification*, *system requirements specification*, *interface requirements specification*, and *technical requirements specification*.

4.2.1 Context analysis

During the context analysis of Designer and Form Creator, the work domain was analyzed and documented in relation to three key areas, namely users, tasks and environment. The main target user group of the two tools includes accessibility researchers and practitioners such as designers, developers, evaluators, etc., and is referred to as the 'Authors' user group. Also, two secondary user groups of the tool were

identified: ‘Users’ and ‘Administrators’. Users constitute people who will actually use the generated web forms themselves. Administrators have the responsibility to administrate and maintain the system.

4.2.2 System goals specification

During the system goals specification phase, the Designer and Form Creator tools were defined in terms of what they should accomplish, what problems they should solve and what user tasks they should support. The main identified goal of the tools is to provide a web-based, comprehensive and easy-to-use environment for designing and deploying web forms that do not require knowledge about accessibility issues and most importantly can utilise existing business logic (e.g. Web services) developed elsewhere. In addition, the tools, like any other online form wizard, should empower the simplicity in the creation of a web form, and the production of valid mark-up code that conforms to W3C standards,

4.2.3 Software requirements

Clearly, all types of form elements should be supported by the tools. In terms of available functionality to each user group, Users should simply be allowed to view, interact with and print the output. Authors should also be able to author, design and administrate web forms. Finally, Administrators are to share the same functionality with Authors, but with full access (create, modify, delete) to all created resources.

4.2.4 Interface requirements

In terms of interface requirements, both tools should incorporate established human-computer interaction guidelines for high quality interaction. Furthermore, as accessibility and usability for people with disability are also of interest, user interfaces should conform to WCAG 1.0 Level AAA. However, conforming to accessibility standards does not necessarily entail that the produced web pages are usable for people with disability (Theofanos & Redish, 2003). For this reason, user tests are planned for the evaluation of upcoming versions of the Designer and Form Creator tools.

4.3 Preliminary Prototyping

Following the analysis of requirements, a number of preliminary paper-based mock-ups and prototypes of limited interactivity were iteratively produced and expert-based reviewed. For the development of these prototypes, Microsoft PowerPoint was used as a facade tool. PowerPoint was chosen as it fulfils the requirements for good prototyping postulated by Szekely (1994): “Ease of use, Fast turn around, Flexibility, Useful throughout the development cycle, Executable and Version control”. The outcome of this phase allowed to work through the details of the system without extensive and time consuming design and programming iterations, leading quickly to a first interactive version of the system. Figure 15 shows two mock-ups of the Form Creator user interface.

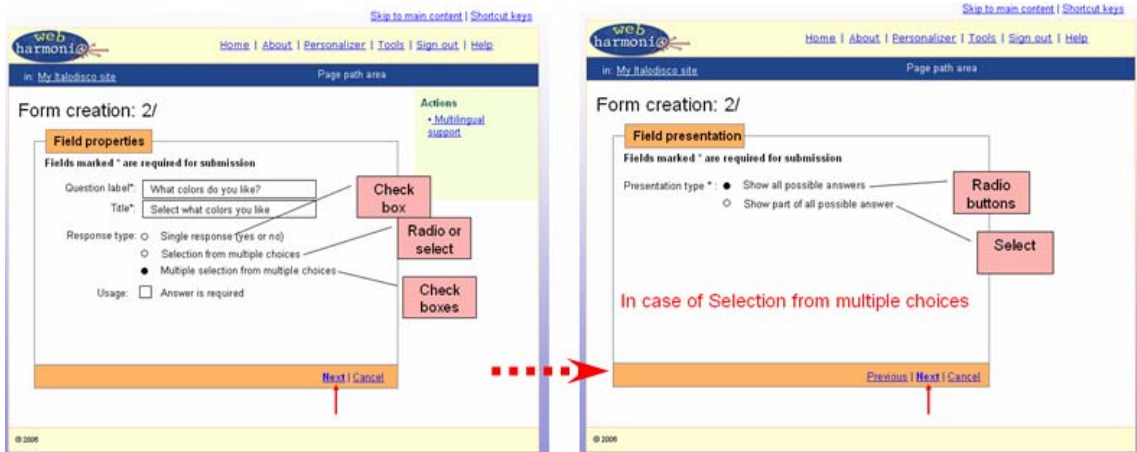


Figure 15: Example of mock-ups produced during the iterative prototyping of the developed tools

4.4 Tool's Architecture

Following the Web Harmonia architecture, both tools are based on a combined **Three-tier** and **WAMP** (variation of **LAMP**) architecture.

The Three-tier architecture contains a Data Layer, a Business Logic Layer and a Presentation Layer, which were developed using MySQL and PHP.

The acronym **LAMP** refers to a solution stack of software, usually free software / open-source software, used to run dynamic Web sites or servers. The original expansion is as follows:

- **L**inux, referring to the operating system
- **A**pache, the Web server
- **M**ySQL, the database management system (or database server)
- **P**HP, the programming language

The combination of these technologies is used primarily to define a web server infrastructure, define a programming paradigm of developing software, and establish a software distribution package.

Though the originators of these open source programs did not design them all to work specifically with each other, the combination has become popular because of its low acquisition cost and because of the ubiquity of its components (which come bundled with most current Linux distributions). When used in combination, they represent a solution stack of technologies that support application servers.

The acronym **WAMP** is used to show that the adopted architecture can be attached also to a Windows operating system, with no further changes. The basic advantages of the above architecture are:

- Open source
- Low cost technologies

- Free and low cost development tools
- Operating system independency
- Interface separation from the logic

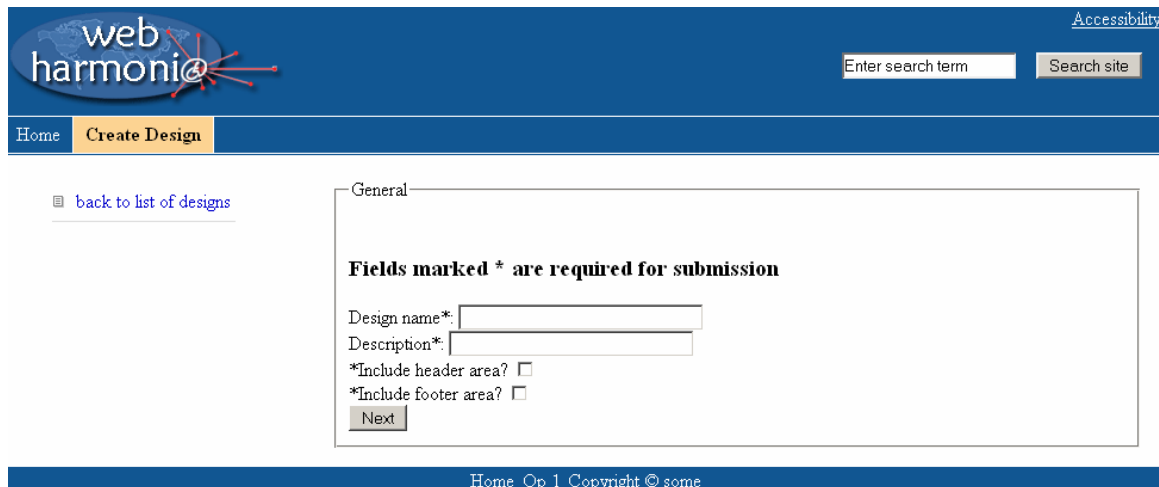
5 Functionality and User Interfaces of the Designer and Web Creator tools

5.1 “Designer” Tool

The Designer tool allows users to create, modify and manage their web page designs. It has a very simple wizard-like user interface facilitating step-by-step form creation. The Designer’s main page provides all the already available designs by the current user or by other users. This helps the user to apply an already created design and skip the design creation procedure. Through the available list of designs, it is possible to preview previously developed designs or edit the current ones. In addition, the main page offers to users the “Create Design” and “Design Elements” modules.

5.1.1 Create / Edit Design

As soon as the user selects to “create a design”, a two-step procedure is initiated. Through the first step of design creation, general information about the design is collected, such as name, description, header and footer areas availability (Figure 16). After completing the first step, information about the design’s navigation is requested. There are three possible options, namely top navigation, side navigation, and top-left navigation. Additionally, the user is asked about the availability of a logo for the item being designed. After completing this step, the user saves the design by clicking on the ‘Next’ button (Figure 17).



The screenshot shows the 'Create Design Step 1' form. At the top, there is a blue header with the 'web harmonio' logo on the left, an 'Accessibility' link on the right, and a search bar with the text 'Enter search term' and a 'Search site' button. Below the header is a navigation bar with 'Home' and 'Create Design' buttons. The main content area is titled 'General' and contains a warning: 'Fields marked * are required for submission'. Below this, there are two text input fields: 'Design name*' and 'Description*'. There are also two checkboxes: '*Include header area?' and '*Include footer area?'. At the bottom of the form is a 'Next' button. A footer at the very bottom of the page reads 'Home Op 1 Copyright © some'.

Figure 16: Create Design Step 1

web harmonia

Accessibility

Enter search term Search site

Home Create Design

[back to list of designs](#)

Design elements

Fields marked * are required for submission

Select navigation placement*:

Top-running navigation

Side-running navigation

Top-and-left navigation (upside-down L)

*My design has logo

Next

Home Op 1 Copyright © some

Figure 17: Create Design Step 2

The basic difference between “create” and “edit” is that although the forms are the same, when modifying an existing design the fields are already filled with the selected design’s information.

5.1.2 Design Elements

The “Design Elements” module includes the following five basic functions (Figure 18):

- *Add forms*, which redirects the user to the “Form Creator” described in the next section.
- *Edit style*, where the user can choose among a collection of WCAG 1.0 level AAA styles.
- *Edit logo*, where, if logo availability is already selected, the user can upload a logo image.
- *Add/edit free text*, for adding any kind of free text in header or footer area, or edit an already created text.
- *Add/edit hyperlink*, in order to help the user adding a hyperlink over the header or footer, or edit an existing one.

web harmoni@ Accessibility

Enter search term Search site

Home **Design Elements**

- [add forms](#)
- [edit style](#)
- [edit logo](#)
- [add free text](#)
- [add hyperlink](#)

Design: My test Design

Header

| Element name | Placement |
|--------------------------------------|--------------------|
| Logo | First row, left |
| Navigation | First row, middle |
| Site path | Second row, left |
| Secondary navigation | Second row, middle |

Footer

| Element name | Placement |
|--------------------------------------|-------------------|
| Auxiliary navigation | First row, middle |

Style information

| Element name | Selection |
|-------------------------------|-----------|
| Applied style | Default |

Home Op 1 Copyright © some

Figure 18: Design Elements list

5.1.3 Preview Design

The Preview module returns a physical presentation of the created design with the added elements. For the newly created designs – which do not contain any elements – only the available placement areas for the elements are presented (Figure 19).



{Auxiliary Navigation Placeholder}

Figure 19: D Designer Pre-viewer

5.2 “Form Creator”

When the user accesses the first page (Figure 20) of the “Form Creator” tool, a list with the forms assigned to the selected design is displayed, along with a list of forms from other designs, to help user create a new one.

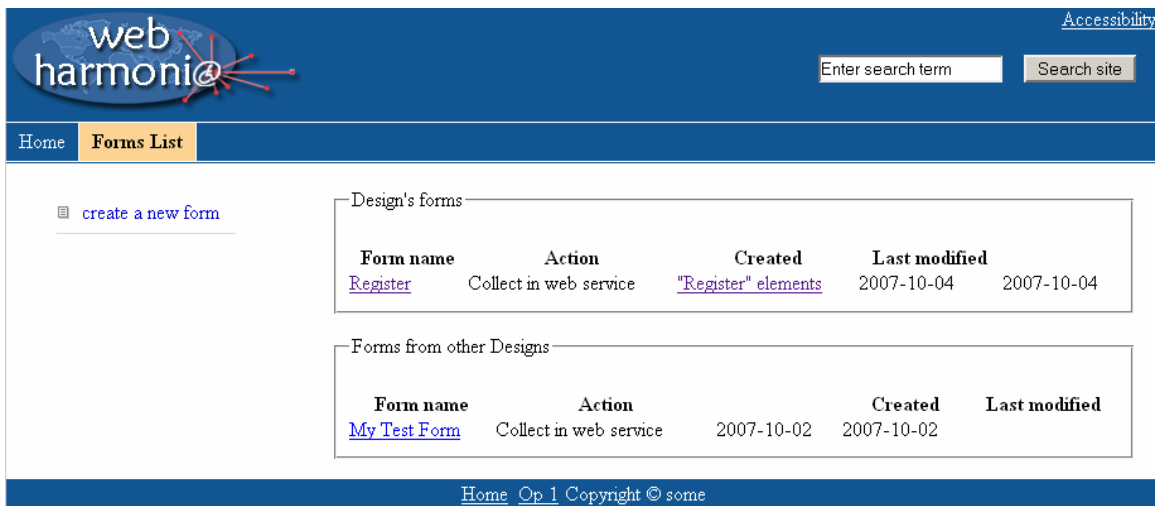


Figure 20: Form Creator - forms list

5.2.1 Create a Form

In order to create a new form, a number of fields have to be filled in (), such as:

- *Form name*, which will be presented also in the secondary navigation of the created page.
- *Form action* and action's link, so that the tool can decide how and where the data inputted in the form will be sent.
- *Number of steps*, in case the designer wishes to split the form elements in steps.
- *Content language*, to provide information about the elements "language" and "character set".

The screenshot shows the 'web harmonia' website interface. At the top, there is a blue header with the logo on the left and an 'Accessibility' link on the right. Below the logo is a search bar with the text 'Enter search term' and a 'Search site' button. A navigation bar below the header contains 'Home' and 'Create Form' buttons. The main content area is titled 'General' and contains a form with the following elements:

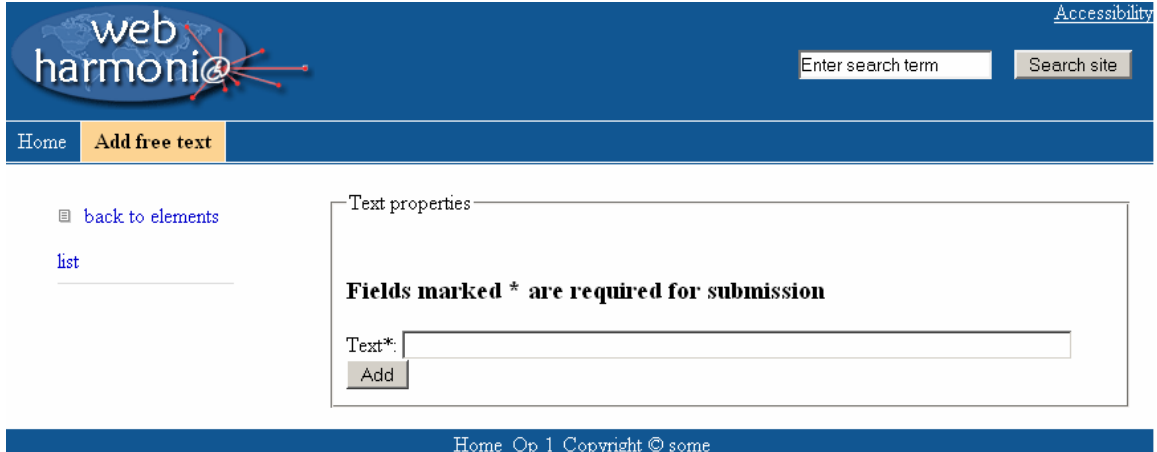
- A link: [back to forms list](#)
- A heading: **Fields marked * are required for submission**
- A text input field: Form name*:
- A section: **Action:**
- Three radio button options:
 - Collect in web service
 - Send data to my email
 - Just post data
- A text input field: Action link*:
- A section: **How many steps:**
- Three radio button options:
 - One
 - Two
 - Three
- A dropdown menu: Page language*:
- A checkbox: My form contains multilingual elements
- A button:

At the bottom of the page, there is a blue footer with the text: [Home](#) [Op 1](#) Copyright © some

Figure 21: Creating a Form

5.2.2 Form Elements

5.2.2.1 Free text



The screenshot shows the 'Add free text' form in the web harmonia system. The page has a blue header with the 'web harmonia' logo on the left and an 'Accessibility' link on the right. Below the header is a search bar with the text 'Enter search term' and a 'Search site' button. A navigation bar below the search bar contains 'Home' and 'Add free text' (highlighted in yellow). The main content area is titled 'Text properties' and contains a warning: 'Fields marked * are required for submission'. Below this warning is a text input field labeled 'Text*:' and an 'Add' button. On the left side of the form, there is a link 'back to elements list'.

Figure 22: Add free text

5.2.2.2 Open-ended questions



The screenshot shows the 'Add open-ended question' form in the web harmonia system. The page has a blue header with the 'web harmonia' logo on the left and an 'Accessibility' link on the right. Below the header is a search bar with the text 'Enter search term' and a 'Search site' button. A navigation bar below the search bar contains 'Home' and 'Add open-ended question' (highlighted in yellow). The main content area is titled 'Open-entity question properties' and contains a warning: 'Fields marked * are required for submission'. Below this warning are three input fields: 'Question label*:', 'Title*:', and 'Pre-fill answer:'. Below these fields is a section for 'Response type*:' with three radio button options: 'Short text' (selected), 'Password (hidden text)', and 'Long text'. Below the radio buttons is a 'Max field size (chars):' dropdown menu set to '50'. An 'Add' button is located at the bottom of the form. On the left side of the form, there is a link 'back to elements list'.

Figure 23: Add open-ended question

5.2.2.3 Multiple choice questions

The screenshot shows a web application interface for creating multiple choice questions. At the top left is the 'web harmonia' logo. At the top right is an 'Accessibility' link and a search bar with the text 'Enter search term' and a 'Search site' button. Below the header is a navigation bar with 'Home' and 'Add multiple choice question' (the latter is highlighted in orange). On the left side, there is a link 'back to elements list'. The main content area is titled 'Multiple choice question properties' and contains the following fields and options:

- Fields marked * are required for submission**
- Question label*:
- Title*:
- Response type*:**
 - Single response (yes or no)
 - Selection from multiple choices, showing all possible answers (using radio-buttons)
 - Selection from multiple choices, showing part of possible answers (using drop-down-list)
 - Multiple selection from multiple choices
- Number of possible answers (not applicable to single response type):
-

At the bottom of the page, there is a footer with the text 'Home Op 1 Copyright © some'.

Figure 24: Multiple choice question creator

web harmonia Accessibility

Enter search term Search site

Home **Add multiple choice question answers**

[back to elements list](#)
[back to element creation page](#)

List of choices

Fields marked * are required for submission

Choice Number 0:

Choice text*:

Choice value*:

Selected?

Choice Number 1:

Choice text*:

Choice value*:

Selected?

[Home](#) [Op 1](#) Copyright © some

Figure 25: Multiple choice answers

5.2.2.4 Grouping

5.2.3 Preview Form

The screenshot shows a web application interface for 'web harmonia'. At the top, there is a blue navigation bar with the logo on the left, an 'Accessibility' link on the right, and a search bar with the text 'Enter search term' and a 'Search site' button. Below the navigation bar, there are two tabs: 'Home' and 'Form elements list', with the latter being the active tab. On the left side, there is a sidebar menu with several items, each preceded by a small square icon: 'back to forms list', 'add free text', 'add open-entity question', 'add multiple choice question', and 'add grouping'. The main content area is titled 'Form elements' and contains a preview of a registration form. The form includes a link 'Here you can register', followed by three input fields labeled 'Username', 'Password', and 'Email'. Below these is a section titled 'Receive notification emails?' with two radio buttons: 'Yes' (which is selected) and 'No'. At the bottom of the form is a 'Country' dropdown menu currently showing 'Greece'. At the very bottom of the page, there is a blue footer bar with the text 'Home Op 1 Copyright © some'.

Figure 26: Previewing Form

5.3 Device Independence and Personalization Support

Server-side adaptation supported by Web Harmonia offers maximum author control over the delivered content, including the ability to radically change the content amount and its styling, navigation, and layout. According to device's capabilities (available in the Web page request's standard HTTP protocol header), generic tags are been "transformed" to appropriate mark-up. Moreover the personalisation mechanism may assist users to select and customize a preferred accessible user interface according to their personal needs.

Table 2 illustrates technical characteristics currently under development, depending on device's capabilities:

Table 2: Device independence support

| Device | Markup | Adaptation | | |
|---------|-----------------------------------|------------|----------------|--------------|
| | | Automated | Semi-automated | User defined |
| Desktop | HTML 4.01, XHTML 1.0 Transitional | ✓ | | |
| | CSS 1, CSS 2 | | ✓ | ✓ |
| PDA | XHTML 1.0, cHTML | ✓ | | |
| | CSS 1 | | ✓ | |
| Mobiles | Basic HTML, cHTML | ✓ | | |
| | | | | |

The web interface of the default profile is a simplistic WAI – AAA (WCAG 1.0) accessible form. Its main characteristic is the absence of layout tables, so that the content becomes readable from the top to the bottom of the page. Additionally, this interface contains a number of navigation shortcuts, such as quick links access keys and tab browsing (Figure 27). When a registered user uses a desktop pc with a browser that supports XHTML, the generated output is a XHTML 1.0 – CSS 2 combination. In case of devices that support different specifications, proper markup is been generated (e.g., the PDA in Figure 28).

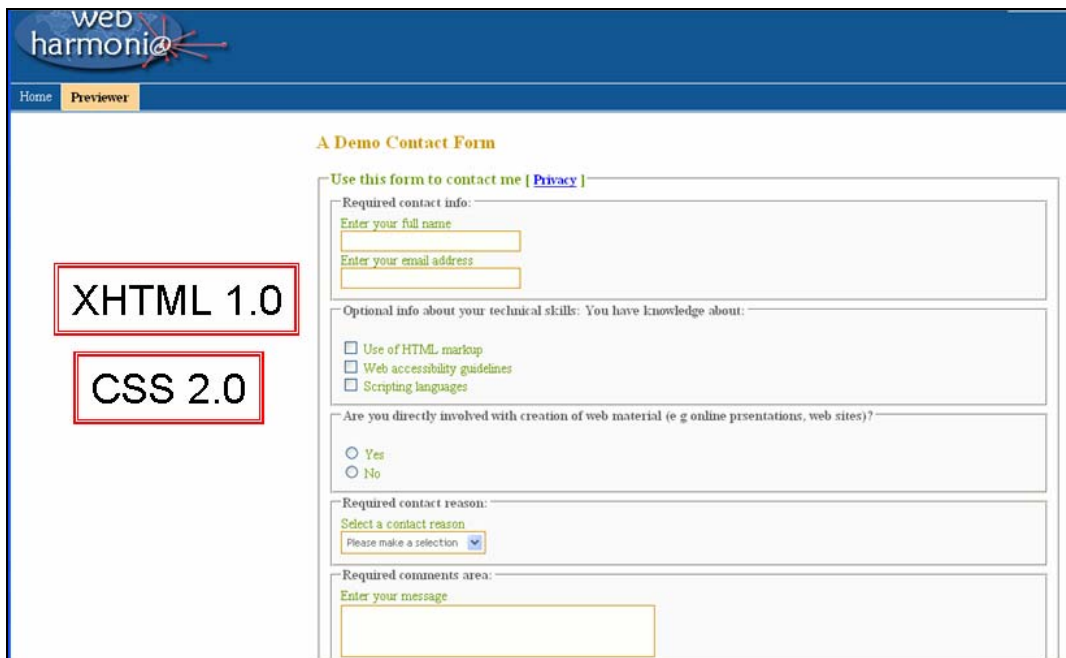


Figure 27: Previewing a demo form (Desktop pc)

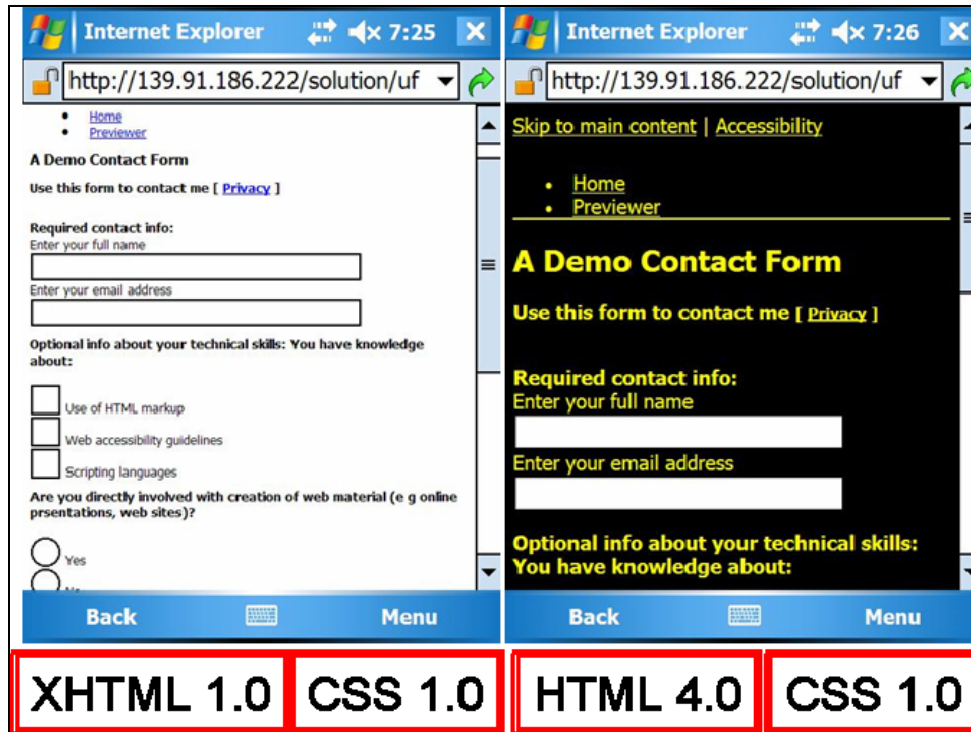


Figure 28: Previewing a demo form (PDA)

5.4 Cost-Effectiveness Analysis

While much web applications are written by professional software engineers, increasingly, important web materials are being created by non-professionals (end-user). In the case of web forms, as they allow a user to enter data that is, typically, sent to a server for processing, and mimic the usage of paper forms, web development knowledge should not be necessary.

The usual model for producing accessible forms is using self-contained collections of form components. Accessibility is established by testing conformance of the resulted form to the best guidelines for accessible authoring (WCAG 1.0). The WCAG themselves do not specify how the goals are to be implemented as they aim to provide normative guidelines that can survive the development of technology. Given the absence of implementation rules, many of the decisions that have to be made at the time of authoring or evaluation are subjective, and it is not possible for an author to know with certainty what is required in any given situation. This means that authors need to be very expert in web accessibility to make authoring decisions. There is ample evidence available to show that such expertise is not common: even among those who are interested and try to make their resources accessible though the Web, there are many perceived difficulties and such people represent only a very small proportion of the people who author digital content for the use of others.

In addition, expertise is often used to design web forms so they perform well on a given set of devices, such as specified software and hardware, when neither of those selected are conformant to the recognised accessibility standards and so what suits the proprietary devices is frequently unsuitable for users with other software or devices.

Currently, an author who attempts to produce an accessible form almost always uses software (e.g., Macromedia Dreamweaver) that enables them to examine and alter the mark-up to increase the accessibility of each element. To make this form universally accessible (peoples dis-abilities, devices technical characteristics), they have to evaluate it over and over again and test in with several devices.

5.4.1 Testing the Cost-effectiveness of the Form Creator

On the basis of the above discussion, the conclusion can be drawn that the production of universally accessible forms using currently available methods and tools has limited practical feasibility, as it is demanding in terms of skills, time and resources.

The proposed tool integrates author expertise for web accessibility, as well as device independence standards, thus making the development of accessible forms much easier and simpler. To confirm such a statement, an informal experiment was conducted in order to assess the effort needed for the production of an accessible web form using a well known authoring tool (Macromedia Dreamweaver) and the Form Creator tool (see Figure 29 and Figure 30). By the comparison, the following results were drawn:

1. Cleaner separation of structure, navigation, and presentation: Form creator allows the user to tackle these aspects separately, and then top merge them together. In a typical setting, earlier prototypes of a form can be created using a default style without interfering with structure and navigation features, and the final look-and-feel is added only at the last rounds of prototyping. By using Dreamweaver, author must apply existing CSS classes in order to cope with appearance.
2. Reduced development effort: this advantage comes from a number of factors: (1) a well-organized development cycle; (2) the stress on prototyping, which causes less revisions or major changes; (3) embedded accessibility standards minimize the effort for testing (4) production of valid mark-up proper for the device in use, minimises the need of several implementations for each situation
3. Improved maintenance and evolution: cost and time reduction shows also during evolution, because changes in the initial form design can be automatically propagated to implementation. Dreamweaver author should provide alternatives for each case.

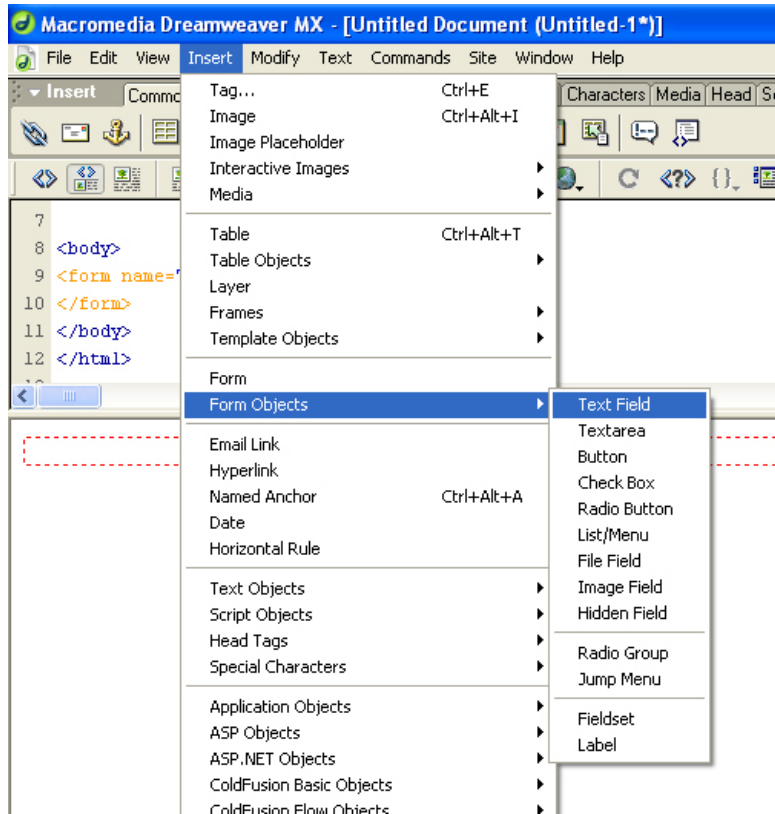


Figure 29: Dreamweaver requires experience in web development, accessibility standards, and produces a WYSIWYG solution

General

Fields marked * are required for submission

Form name*:

Action:

Collect in web service
 Send data to my email
 Just post data

Action link*:

How many steps:

One
 Two
 Three

Page language*:

My form contains multilingual elements

[Create and add elements](#)
[Back to forms list](#)

Figure 30: Form Creator requires minimum experience in web development

Additionally, it can be claimed that designing a web form using a well founded and descriptive formalism improves quality (e.g., the navigation model), with respect to less organized approaches that require technical skills. Improvements come in terms of increased navigation power, increased usability, and consistency across the web pages. Dreamweaver has descriptive a formalism, however the resulted mark-up must be enriched in order to address accessibility requirements.

5.5 Discussion

In times of increasing complexity and reliance on technology, it is important to ensure that web material that is being produced can be utilised by the biggest possible audience, including people with disabilities. Opportunities lost by early introduction of new technologies must be repaired by careful development of newer technologies, but it is important to balance cost-effectiveness with outcomes.

The production of tools that help in the development of a universally accessible web form and web material in general, may contribute in altering the results of recent surveys that demonstrate a very low accessibility conformance for the majority of Web sites. The two tools developed in the context of this thesis help towards this direction with the provision of web resources:

- ◆ comply with all Priorities 1, 2 and 3 of WCAG 1.0 (Figure 31)
- ◆ produce valid mark-up (W3C Specifications)
- ◆ resulting web pages can be accessed through alternative devices (e.g., PDAs)

Results for <http://139.91.186.222/solution/kartakis/designer/list.php?uid=2>

Page last checked on Sun 07/10/2007 at 7:59pm.

| | Automatic Checkpoints | | | Manual Checkpoints | | |
|----------------------------|-----------------------|--------|-----------|--------------------|----------|-----------|
| | Status | Errors | Instances | Status | Warnings | Instances |
| Priority 1 | ✓ | 0 | 0 | ⚠ | 6 | 6 |
| Priority 2 | ✓ | 0 | 0 | ⚠ | 9 | 9 |
| Priority 3 | ✓ | 0 | 0 | ⚠ | 6 | 6 |

Figure 31: The Form Creator tool complies with WCAG 1.0 Level AAA (automatic check)

6 Evaluation

WCAG 1.0 is the de-facto standard for developing accessible web sites. There are some automatic validation tools available to developers, such as Bobby Watchfire. However, testing the level of compliance against this standard only entails a number of drawbacks:

1. The results of such an automatic validation tool present extremely valuable information, but are often overwhelmingly long and detailed, making them difficult to interpret, particularly to non-expert Web developers.
2. An automatic validation tool can only detect a limited number of potential accessibility barriers, and thus additional manual inspection is required on the part of developers in order to supplement results.
3. In some cases, automatic validation tools find a resource inaccessible, even though it does reach an acceptable level of accessibility.

To this effect, although checking against guidelines and using automatic validation tools provides an approximation of the level of accessibility of the resulted web form, no indication of the accessibility of the final outcome is provided.

To overcome this drawback, two different approaches to the evaluation of the two tools, as well as of their outcomes, have been followed. The first was to systematically test compliance against WCAG 1.0 level AAA with the use of the Bobby WatchFire. The second was to perform user-based evaluation, targeted towards assessing the tools accessibility, usability, and usefulness. This testing was performed by one web developer and one blind user.

6.1 Methodology

In order to assess of the Designer and Form Creator tools from an accessibility perspective, the ‘*Preliminary Review*’ process was followed as suggested by W3C-WAI²⁹. This process mainly involves some manual checking of a sample of pages of the tool and of the resulting forms, along with the use of several semi-automatic accessibility checkers. First, the main webpages constituting the tools were examined through the use of various graphical user interface (GUI) browsers (such as Internet Explorer³⁰ 6.0 and 7.0, Netscape Navigator³¹ 7, Mozilla 1.6, Firefox³² 1.0 and 1.5, and Opera³³ 7.54) while adjusting the browsers’ settings. This preliminary review mainly helped to quickly identify some accessibility ‘bugs’ in the source code (such as elusive alt tags provision) and improve the colors and structure of the templates used.

²⁹ Evaluating Web Sites for Accessibility (<http://www.w3.org/WAI/eval/>)

³⁰ Internet Explorer (<http://www.microsoft.com/windows/ie/>)

³¹ Netscape Navigator (<http://home.netscape.com/browsers/>)

³² Mozilla Firefox (<http://www.mozilla.org/>)

³³ Opera (<http://www.opera.com/>)

Two of the most effective automatic validation tools available for assessing accessibility were used to validate the subject site, namely:

- ◆ Bobby - a popular automatic accessibility evaluation tool, the results of which provide important, although incomplete information regarding accessibility levels of a site. As mentioned, Bobby can be a reliable tool to check entire sites for every guideline that can be validated automatically.
- ◆ W3C HTML & CSS Validation Tools - not an accessibility evaluation tool as such, checking however the HTML source code behind a web page for compliance with the HTML standard as specified in the code of the page. A key requirement for assistive technologies to correctly interpret web pages is that the pages must be written in valid HTML, and therefore it is important to include an HTML validation stage in the methodology.

The second evaluation step that took place was a user-based evaluation, targeted towards assessing the tools accessibility, usability, and usefulness. This evaluation phase involved one web developer and one blind user - an experienced web and Jaws (screen reader technology) user - with some technical knowledge concerning the creation of paper-based forms and electronic presentations (PowerPoint).

The method utilised in both cases was the 'think aloud' protocol, where the users were asked to vocalise their thoughts whilst exploring the tools and carrying out a predetermined task. Users were initially given five minutes of free exploration prior to commencing the task.

The users were guided with the use of specific scenarios. Three quantitative measures were recorded:

- ◆ Ease – users were asked to rate how easy the site made it for them to do the task (between 1 = difficult and 7 = easy)
- ◆ Navigation – users were asked to rate how easy it was to navigate their way around the tool (between 1= difficult and 7 = easy)
- ◆ Impairment – the blind user was asked to rate if her impairment had been taken into consideration (between 1 = not at all and 7 = completely)

Two other metrics were recorded:

- ◆ Time taken to complete task
- ◆ Success or failure of the task

The tasks tested were:

- ◆ The creation of a web design (web developer)
- ◆ The creation of a web form (both).

6.2 Evaluation Results

Overall, the web pages produced using the tools comply with WCAG 1.0. However, according to Checkpoint 14.1 *Use the clearest and simplest language appropriate for a site's content*, all dialogs should be re-examined. The tools, besides fully accessible output, generate valid mark-up in three combinations of browser-device examined (Latest version of Internet Explorer – Desktop, Old version of Netscape Navigator – Desktop, Internet Explorer - PDA).

Overall, the users completed the tasks successfully. The blind user, although she had some understanding of the process of creating an online form, experienced more difficulty in using the tools in comparison to the other participant (web developer); this user succeeded in completing the task of creating a form, but she asked many clarifications during the process as she was not familiar with the terms used (e.g., mandatory field, form title and others).

Both users were asked to rate how easy it was to navigate their way around the tool(s) after exploration, and again after they completed the task. The mean rating was 4. The users were also asked to rate the degree to which their impairment was considered; the blind user gave an overall impairment rating of 2. The disabled user asked for help pages (current version does not provide any), not knowing how to use the services provided, and she experienced lack of feedback through assistive technology, although all accessibility features have been incorporated.

The main problems that each user experienced are listed below:

Key problems experienced by the blind user:

-
- Lack of meaningful language: technical terms must be translated into “plain English”
 - Insufficient feedback– information in list boxes not read, alt text not meaningful
 - Navigation was not so helpful: user requested a “back to main page” feature at the end of each screen
-

Key problems experienced by the web developer:

-
- Provision of more advanced features
 - Use of existing Web services to post form’s data (not available in current version)
 - Design was not attractive
 - Lack of help pages
 - No functionality was available to cancel, undo or redo, the user was forced to start

from the beginning each time

- User friendly urls: for demonstrating the resulted form in a PDA, user had to type a quite long url

The above indicates that there are a number of usability and accessibility problems associated with the use of information visualisation, mainly with the users in understanding the data and successfully using the service. The evaluators on this occasion were not subject matter experts for the services they were asked to look at, therefore struggled with interpreting the data. As a result, a number of design improvements have emerged from this process and will be used to further improve the platform and the tools. These include modifications related to:

- ◆ provision of sufficient feedback for each action (e.g., make use of “hidden” text for tipping the user)
- ◆ improved navigation
- ◆ provision of alternative design templates
- ◆ adequate help support
- ◆ carefully selected labels, headings and wording

7 Conclusions and Future Work

The World Wide Web was initially designed to be a medium for sharing information, i.e., people not only read information, but also contribute. In order for this to become feasible for people with disabilities, not only web content must be accessible, but also the tools used to create and modify Web content must be accessible. Ensuring that people with disabilities are able to contribute content is an important crucial argument for making the Web accessible.

At the same time, the proliferation of new devices and the need for information access anywhere and at anytime provide new opportunities to further enhance the Web. Today, users can interact with Web data via a wide range of devices-browsers, but there is still lack of universal interchange data formats. This prevents consumers from exploiting the true value of these rich data. Additionally, there is in many cases lack of compliance with accessibility guidelines and with more generic W3C standards.

Modern solutions, without exceptions, provide platforms and tools for the development of web applications that vary in complexity, from simple static web pages, such as a simple curriculum vitae, to complex processes running CRM (customer relationship management) or enterprise resource planning (ERP) systems. However, such tools can not be utilised by a person using a screen reader who needs to design a simple web page with a simple form. In addition, despite the “rise” of SOAP Web Services, which have been designed to be accessed by other applications since they are based on open standards such as HTTP and XML-based protocols (e.g., SOAP, WSDL), there are only one or two platforms for building interfaces for these applications, and they require software engineering knowledge. Consequently, even though Web services support the idea of seamlessly exchanging data over intranets or the Internet between applications that are written in different programming languages and running on different platforms, in practice it seems impossible to interconnect a simple web interface with the application logic provided by someone else.

When building web applications (i.e., interface and content), web developers are working with a fairly restricted set of widgets, compared to those available for native desktop applications. In that respect, with the provision of fully accessible native widgets, which can be manipulated according to personal needs and device characteristics, one can replicate existing solutions and provide web users with exciting interfaces that bridge the gap between native applications and web applications. However, building interfaces this way usually reduces accessibility. . Web interfaces however, ought to provide the same central functionality to all users and content should be presented in more than one ways.

7.1 *Contribution of this Thesis*

This thesis has discussed the importance of universally accessible web forms in the context of the current evolution of the web, towards a global platform for information and services provision, and has presented two tools, called Form Creator and Designer, which

support the provision of universally accessible web forms in the context of the novel web development platform Web-Harmonia. Web forms were selected as the primary target as these elements are included in almost all web sites since they allow transactions and data gathering from users. Web forms are one of the most difficult aspects of web development, largely because they entail stepping out from simply presenting information to the user of a site.

Creating accessible online forms is a really challenging task, particularly when forms have to be accessible to screen reader users. This is due to the fact that there is a variety of form control types — text, checkboxes, radio buttons, menus, etc. - each with its own distinct accessibility challenges. The two aforementioned tools offer considerable advantages with respect to current practice, as they generate web forms which:

- Fully comply with WCAG 1.0
- Can be accessed by various devices through a web browser
- Can be personalised according to user preferences

These tools are claimed to highly simplify and enhance the production of web forms, as they:

- are operated through a web-based wizard user interface which is itself fully accessible
- do not require prior knowledge or experience in web accessibility and web development
- do not require additional programming
- are extensible by importing interaction resources offered by different interaction platforms
- are orthogonal to other tools, i.e., support the use of business logic produced by external software tools.

The two prototype tools have been evaluated with respect to both accessibility and usability. Accessibility testing has been carried out mainly through the use of automated tools, and the results confirm that both the tools and the produced forms are AAA compliant. Preliminary usability evaluation was conducted through a small think-aloud walkthrough involving an experienced web developer and a blind user. The results have confirmed the usefulness of the tools and have identified some issues for further usability improvement, including the need for a help facility, undo/redo functions, clearer language and terminology, more informative alt text, and enhanced navigation for blind users. However, more extensive usability evaluation is needed.

7.2 Future directions

The need to access and manipulate all types of existing data from everywhere, with every kind of device, becomes increasingly critical in order to achieve successful solutions. Nowadays, technologists are searching for a way to make different kinds of data seamlessly integrate together and transcend multiple data protocols, languages, devices and users. Web tools that support the creation of standardised web content in a cost effective way that can be utilised by all, may provide a solution to such issues. To support

the complete separation of content – application logic for the presentation, a mechanism is needed to enable universal access. This mechanism should provide design solutions and interconnections to business logic developed elsewhere, without the need for data conversions and further adaptation processes, ensuring that:

- ◆ Creation of simple web material can be done by all
- ◆ Presentation elements and content can be adapted for people with disabilities (e.g. use of assistive technologies) with a high level of usability,
- ◆ Accessibility features are integrated into, and a natural part of, technical specifications not requiring a high level of additional effort by content producers nor bloating content
- ◆ Device Independence

The actual development of such suite of tools for the support of web content development will decrease the maintenance costs, but most importantly the development time. The principal benefit consists of making just widgets maintenance and, if a new target language (e.g. HTML 5.0) is needed, a new instance of them. Future work is planned to include:

- Further testing with additional devices (e.g., iTV)
- Enrichment of the available form elements (e.g. Likert-style questions)
- Inclusion of automated validation methods in the created forms
- Development of total solutions for the creation of universally accessible web sites.

8 References

Adipat, B. and Zhang, D. (2005): Adaptive and Personalized Interfaces for Mobile Web. Proceedings of the 15th Annual Workshop on Information Technologies and Systems (WITS'05). Las Vegas, NV, 2005. 21–26.

Basdekis, I., et al. (2007, forthcoming): Development of Web-Harmonia - an interaction platform to facilitate the production of universally accessible web interfaces in a cost-effective and standard-compatible way. FORTH-ICS, Heraklion, Crete, Greece.

Billsus, D., Pazzani, M. and Chen, J. (2000): A learning agent for wireless news access. In Proceedings of the International Conference on Intelligent User Interfaces. (New Orleans, LA, Jan. 9–12, 2000), 33–36.

Bureau of the Census (2000): “Internet access, computer use, and disability status: 1999.” Survey of Income and Program Participation (1999, unpublished tabulation). Washington, DC: Author.

Buyukkokten, O., Garcia-Molina, H., and Paepcke, A. (2001): Seeing the whole in parts: Text summarization for Web browsing on handheld devices. In Proceedings of the 10th International WWW Conference. Hong Kong (May 1–5, 2001).

Casali, S. P., (1995): *A physical skills-based strategy for choosing an appropriate interface method*. In Edwards, A. D. N., (Ed.), Extra-ordinary Human Computer Interaction: Interfaces for people with disabilities, Cambridge University Press, pp. 315-342.

Ceri, S., Fraternali, P., Paraboschi, S.(1999): “Data-Driven One-to-One Web Site Generation for Data-Intensive Applications”. Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999, pp 615-626.

Clark, J.(2003): "Building Accessible Websites" New Riders Publishing, Indianapolis Indiana.

Clark, J., (2005): Web Standards Group interview, Retrieved 10 September 2007 from <http://webstandardsgroup.org/features/joe-clark.cfm>

Clark, J.,(2005): Big Stark and Chunky, CSS articles and tutorials A List Apart, ISSN: 1534-0295. 11 January 2005 – Issue No.

Cloninger, C. (2002): “Usability Experts Are from Mars, Graphic Designers Are from Venus,” A List Apart, No. 74. Retrieved July 28, 2005 form <http://www.alistapart.com/articles/marsvenus/>.

Communications of the ACM, (2000). Special Issue on Personalization. Volume 43, Number 8.

Dalton, P. (1997): Microsoft SQL Server Black Book. The Coriolis Group.

European Parliament (2002). eEurope 2002: Accessibility of Public Web Sites and their Content - European Parliament resolution on the Commission communication. January 7, 2005, from http://europa.eu.int/information_society/topics/citizens/accessibility/web/wai_2002/ep_res_web_wai_2002/index_en.htm.

Glazkov D., (2005): Keeping “pretties” out of content, Retrieved 20 July 2007 from <http://glazkov.com/blog/graphics-and-markup/>

Hanson, Vicki L., Richards, J.,(2004): The User Experience: Designs and Adaptations, ASSETS'04, October 18–20, 2004, Atlanta, Georgia, USA. Copyright 2004 ACM 1-58113-911- X/04/0010

Hudson, R., (2005): Disabilities and Technologies, Retrieved 1 October from <http://www.usability.com.au/resources/statistics.cfm>

Hudson, R., Weakley, R., and Firminger, P., (2005): Developing sites for users with Cognitive disabilities and learning difficulties, Retrieved 10 September 2007 from <http://juicystudio.com/article/cognitive-impairment.php>

Jiang, Z. and Kleinrock, L. Web prefetching in a mobile environment. IEEE Personal Communication (Oct. 1998), 25–34.

Johansson, R., (2004): The alt and title attributes, Retrieved from http://www.456bereastreet.com/archive/200412/the_alt_and_title_attributes/

Johansson, R., (2006): Setting font size in pixels, Retrieved 20 August 2007 from http://www.456bereastreet.com/archive/200602/setting_font_size_in_pixels/

Korpela, J. (1998): IT and Communication. "Improving accessibility with accesskey in HTML forms and links". Available at: <http://www.cs.tut.fi/%7Ejkorpe/forms/accesskey.html>

Laakko, T. and Hiltunen, T. (2005): Adapting Web content to mobile user agents. IEEE Internet Computing (March–April, 2005), 46–53.

Lank, E. and Phan, S. (2004): Focus+context sketching on a pocket PC. In Proceedings of ACM CHI 2004 (Apr. 24–29, 2004, Vienna, Austria), 1275–1278.

Lum, W. and Lau, F. (2003): “User-centric content negotiation for effective adaptation service in mobile computing”. IEEE Transactions on Software Engineering 29, (2003), 1100–1111.

Mao, Z., So, H., Kang, B., and Katz, R. (2001): "Network support for mobile multimedia using a self-adaptive distributed proxy". In Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video. Port Jefferson, NY, (June 25–26, 2001), 107–116.

Marincu, C., & McMullin, B. (2004). A Comparative Assessment of Web Accessibility and Technical Standards Conformance in Four EU States. Retrieved December 23, 2004, from <http://eaccess.rince.ie/white-papers/2004/warp-2004-00/> .

Nylander, S. and Bylund, M. (2002) Device Independent Services, SICS Technical Report T2002-02, Swedish Institute of Computer Science.

Nyman R., March (2006): What is Accessibility?, Retrieved 10 June 2007 from <http://www.robertnyman.com/2006/03/01/what-is-accessibility/>

Platform for Privacy Preferences Project. Available at <http://www.w3.org/P3P>.

Plomp C.J., Mayora-Ibarra O. A., (2002): Generic Widget Vocabulary for the Generation of Graphical and Speech-Driven User Interfaces, International Journal of Speech Technology, Vol 5, 39-47, 2002.

Puerta, A., Eisenstein, (2001): XIML: A Common Representation for Interaction Data, Proceedings ACM IUI'01, pp.214-215.

Richards, J., Hanson, V., and Trewin, S., (2003). "Adapting the Web for Older Users," In Universal Access in HCI (Vol 4): Inclusive Design in the Information Society, C. Stephanidis, Editor, (2003), pp. 892–896.

Schwabe, D., Rossi, G.(1998): "An object-oriented approach to web-based application design". Theory and Practice of Object Systems (TAPOS), Special Issue on the Internet, v. 4#4, pp. 207-225, October, 1998.

Sierkowski, B. (2002). Achieving web accessibility. In Proceedings of the 30th annual ACM SIGUCCS conference on User services (pp. 288-291). Providence, Rhode Island, USA.

Stephanidis, C., & Savidis, A. (2001). Universal Access in the Information Society: Methods, Tools and Interaction Technologies. Universal Access in the Information Society, 1 (1), 40-55 (Managing Editor: Reinhard Oppermann, GMD, Germany).

Stephanidis, C., Akoumianakis, D. (2003). "A design code of practice for universal access: Methods and techniques". In R. Proctor & K. Vu (Eds.). "The Handbook of Human Factors in Web Design", to be published by Lawrence Erlbaum Associates, Inc.

Slatin, J. and Rush, S. (2003): "Maximum Accessibility", Addison-Wesley, Boston.

U.S. Code (1998), The Rehabilitation Act Amendments (Section 508). Retrieved January 7, 2005, from <http://www.access-board.gov/sec508/guide/act.htm>

Yang, C.C. and Wang, F.L. Fractal summarization for mobile devices to access large documents on the Web. In Proceedings of the International WWW Conference (Budapest, Hungary, May 20–24, 2003), 215–224.

Zhang, D. and Adipat, B. Challenges, methodologies, and issues in the usability testing of mobile applications. *International Journal of Human Computer Interaction* 18, 3 (2005), 293–308.

World Wide Web Sources

Australian Bureau of Statistics, (2003): "Disability, Ageing and Carers Survey: Summary of Findings". PDF copy available at: [http://www.ausstats.abs.gov.au/ausstats/subscriber.nsf/Lookup/978A7C78CC11B702CA256F0F007B1311/\\$File/44300_2003.pdf](http://www.ausstats.abs.gov.au/ausstats/subscriber.nsf/Lookup/978A7C78CC11B702CA256F0F007B1311/$File/44300_2003.pdf)

Brown, S. Style sheets for low vision. Available at: <http://people.pwf.cam.ac.uk/ssb22/css/>

Dahm, T. Browser Compatibility Tutorial. Available at: <http://www.netmechanic.com/products/Browser-Tutorial.shtml>

EUROSTAT, (ECHP UDB 06/2003): "Hampered in daily activities by any physical or mental health problem, illness or disability ". Available at: http://epp.eurostat.cec.eu.int/portal/page?_pageid=1073,46870091&_dad=portal&_schema=PORTAL&p_product_code=HAMPERED

Pilgrim, M. "Dive Into Accessibility". Available at: <http://www.diveintoaccessibility.org/>

Shannon, R. "Basic Forms". Available at: <http://www.yourhtmlsource.com/forms/basicforms.html>

Statistics Canada, (2001): "Prevalence of disability in Canada (2001)". Available at: <http://www.statcan.ca/english/freepub/89-577-XIE/canada.htm>

Statistics New Zealand, (2001): "Disability Counts 2001". Available at: <http://www2.stats.govt.nz/domino/external/pasfull/pasfull.nsf/web/Reference+Reports+Disability+Counts+2001?open>

Thatcher, J. "Web Accessibility for Section 508". Available at: <http://www.jimthatcher.com/webcourse1.htm>

UK National Statistics Office (2001), "Health Status: Limiting Long-term Illness or Disabilities". Available at: <http://www.statistics.gov.uk/cci/nugget.asp?id=916>

US Census Bureau, (2003): "Disability Status: 2000". Available at: <http://www.census.gov/prod/2003pubs/c2kbr-17.pdf>

WebAIM (2007). A Review of Free, Online Accessibility Tools. Retrieved September 7, 2007, from <http://www.webaim.org/articles/freetools/>

World Wide Web Consortium. "Essential Components of Web Accessibility". Available at: <http://www.w3.org/WAI/intro/components.php>

World Wide Web Consortium. "Web Content Accessibility Guidelines 1.0. Available at: <http://www.w3.org/TR/WAI-WEBCONTENT/>

W3C-WAI, (2003): Web content accessibility guidelines 2.0 - internal draft. www.w3.org/WAI/GL/WCAG20/, Nov 2003.

W3C-WAI, (2007): Selecting Web Accessibility Evaluation Tools. <http://www.w3.org/WAI/eval/selectingtools.html> Ock 2007.

W3C-WAI: Web characterization terminology & definitions. Available at <http://www.w3.org/1999/05/WCA-terms/>.

XUL, XML User Interface Language. Available at: <http://www.xulplanet.com>