Making Web Pages and Applications Accessible Automatically Using Browser Extensions and Apps

Ignacio Peinado and Manuel Ortega-Moral

Fundosa Technosite, Madrid, Spain
{ipeinado,mortega}@technosite.es

Abstract. Web accessibility depends on three factors: the semantics of the web contents, the assistive technologies (ATs) and the capabilities of the web browsers (Fernandes, Lopes, & Carriço, 2011). Moreover, the widespread implementation of Rich Internet Applications (RIAs) poses new challenges for ensuring the equality of access to dynamic web content. This paper presents the development of a solution that will automatically activate the accessibility features and the available ATs in two web browsers that take more than 50% of web browsers market share, and depending on the expressed needs and preferences of the user. The two extensions presented will take advantage of the infrastructures developed in CLOUD4all and APSIS4all in order to inject CSS and JavaScript in any web pages, as well as activating non-out-of-the-box ATs, and hence guaranteeing access to both static HTML pages and Rich Internet Applications.

Keywords: e-Accessibility, Web accessibility, ATs, adaptation of accessibility features, CLOUD4all, APSIS4all.

1 Introduction

People use interactive systems every day. Nowadays, we need technology for performing common activities such as withdrawing money from an ATM, attending online courses or performing administrative duties. These activities can be performed on a plethora of devices and environments: on the same day; we might withdraw money from an ATM, buy a train ticket from a Ticket Vending Machine and then read our mail in our mobile phone on our way to work, where we might then use a computer to perform our daily activities. Nevertheless, many of these technologies do not fulfill the needs and preferences of people with special needs, preventing them from fully participating in relevant activities such as e-Government or e-Learning.

Electronic Accessibility or e-Accessibility has been defined by the World Health Organization as referring to "the ease of use of information and communication technologies (ICTs), such as the Internet, for people with disabilities" (World Health Organization, 2013). The International Organization for Standardization (ISO)

C. Stephanidis and M. Antona (Eds.): UAHCI/HCII 2014, Part IV, LNCS 8516, pp. 58–69, 2014.
© Springer International Publishing Switzerland 2014

broadens the scope defining accessibility as "the usability of a product, service, environment or facility by people with the widest range of capabilities" (ISO, 2008). Problems that people with disabilities encounter when using interactive systems range from minor annoyances to critical flaws that prevent any use of the system at all.

Web accessibility depends on three factors: web page semantics, assistive technology (AT) and Web browser capabilities (Fernandes, Lopes, & Carriço, 2011). The Web Content Accessibility Guidelines (WCAG) 2.0 provides a wide range of recommendations for making web content more accessible (World Wide Web Consortium, 2008). Additionally, User Agent Accessibility Guidelines (UAAG) provides guidelines for designing user agents – i.e. media players, assistive technologies and, especially, web browsers – that lower barriers to Web accessibility (World Wide Web Consortium, 2013). Moreover, the emergence of Rich Internet Applications (RIAs) has posed new accessibility challenges. RIAs use client-side scripting languages, such as JavaScript, or asynchronous communication technologies such as AJAX to enrich the interactivity of web pages. Since ATs need to interact with this dynamic content, new guidelines had to be developed in order to ensure that all content is properly presented to users. The WAI-ARIA, the Accessible Rich Internet Application Suite, defines a way to make Web Content and Web Applications more accessible to people with disabilities. WAI-ARIA provides a framework for adding attributes to identify features for user interaction, how they relate to each other, and their current state (World Wide Web Consortium, 2014).

Popular web browsers like Mozilla Firefox or Google Chrome have implemented several accessibility features such as keyboard shortcuts, page zooms or mechanisms to control web content (Using Google products: How to use accessibility features, 2014) (Accessibility features in Firefox - Make Firefox and web content work for all users, 2014). Plus, browser extensions and apps permit the provision of accessibility features that are not built-in in the browser. For instance, Google has developed extensions such as ChromeVis[1] or ChromeVox[2] that provide users with out-of-the-box accessibility features. Unfortunately, many users are not aware of the accessibility features provided by their browsers or of the existence of extensions and apps that might fulfil their needs.

This paper presents the development of two extensions for Google Chrome and Mozilla Firefox that will take advantage of the infrastructure developed within the CLOUD4all and APSIS4all FP7 projects. These browser extensions with provide users with the automatic activation of the accessibility features available both in the browser and in third-party browser extensions, depending on their expressed needs and preferences.

[1] https://chrome.google.com/webstore/detail/chromevis-by-google/halnfobaneppemjnonmmhngbfifnafgd

[2] https://chrome.google.com/webstore/detail/chromevox/kgejglhpjiefppelpmljglcjbhoiplfn

1.1 Automatic Personalization of Accessibility Features: The APSIS4all and CLOUD4all Approaches

Many researchers and developers have proposed solutions for facilitating the configuration of devices, platforms and applications depending on the preferences of the users. Many of these approaches rely on the construction of user models (INREDIS (INREDIS Consortium, 2010), VUMS cluster (Peissner, et al., 2011)) or on the dynamic generation of optimal UIs (Cameleon Framework (Calvary, et al., 2002), SERENOA (SERENOA project Consortium, 2013)). The approach adopted by APSIS4all and CLOUD4all is slightly different: both projects do not perform user modelling, but capture in a generic form the needs and preferences of the users when interacting with technology. Thus, instead of characterizing a user as 'low vision', both systems will define concrete interaction preferences such as 'high Contrast activated' or 'font size: big'. This permits for much more flexible configuration capabilities, allowing users outside any cluster to define individual parameters for individual situations.

The APSIS4all project is an ICT-PSP project led by Technosite[3] and partially funded by the EC that is developing mechanisms for allowing end-users to overcome the accessibility barriers of public digital terminals (PDTs) such as automated teller machines (ATMs) or ticket vending machines (TVMs). APSIS4all has developed a set of accessible, usable User Interfaces (UIs) that will provide end-uses with an easy way for describing their needs and preferences when interacting with technology. Once users have configured their preferences through a web tool, the UIs of any APSIS4all-compatible PDTs will be able to automatically adapt their UIs to these expressed needs and preferences, therefore facilitating the interaction. APSIS4all has adopted two approaches for enhancing the interaction with PDTs: a 'direct interaction' approach, where users can interact with machines that get adapted; and an 'indirect interaction' approach, where users circumvent the use of non-accessible PDT with an accessible mobile web that represents the machine. Preferences are stored according to specification EN 1332-4, a standard for coding interaction needs. Because the coding of preferences is based on standards, the PDT can retrieve the user's needs and preferences regardless of the service provider, and provide the most suitable UI for that specific set of needs and preferences. The web tool for collecting the needs and preferences of the user is currently online[4]. More than 1000 ATMs from "la Caixa", one of the most important banks in Spain, are APSIS4all-compatible. Although the results from the APSIS4all project have been really promising, it is only limited to the adaptation of the UIs of compatible PDTs, hence lacking generality.

CLOUD4all is another EC-funded project from the 7th Framework Programme led by Technosite that moves one step forward from the work of APSIS4all. CLOUD4all is part of a greater effort - the Global Public Inclusive Infrastructure (GPII) - that aims at developing a complete new paradigm in accessibility by extending the concept of

[3] http://www.technosite.es
[4] http://cajerofacil.apsis4all.eu

automatic configuration of the accessibility features from specific devices to any technology that users encounter during their daily lives. CLOUD4all uses cloud technologies to activate and augment any natural (built-in) accessibility or installed access feature, or recommends the appropriate third-party solutions that better fit the user s' needs. In this sense, CLOUD4all aims at overcoming two important accessibility barriers: 1) users will be recommended with the ATs that better suit their needs and preferences, and 2) users will not need to configure the settings of any CLOUD4all-compatible device they use.

Within CLOUD4all up to 19 proof-of-concept implementations of the auto-configuration capabilities are being developed for different devices, platforms and applications. To this point, it can be installed in desktop computers with Windows 7 and Fedora with Gnome, and in Android smartphones; third-party AT solutions such as Mobile Accessibility for Android (developed by Code Factory)[5], Read&Write Gold (by TextHelp)[6] or Maavis (by OpenDirective)[7] have been modified in order to become CLOUD4all-compatible. On top of that, cloud-based AT solutions such as WebAnywhere[8] or SAToGo[9] can be launched and set up automatically. In order to launch and configure these solutions, some CLOUD4all modules need to be installed in the host device, and solution implementers had to provide access to the settings and the process launchers. However, the experience showed that the access to the device is not always possible or applications do not allow other applications to change the settings on-the-fly. Hence, how can we provide auto-configuration from preferences in those situations?

This paper presents an approach to auto-configuration for web content using CLOUD4all and APSIS4all preferences. This proof-of-concept implementation will permit both Mozilla Firefox and Google Chrome users to change their access features in any platform (Windows, Mac OS X and Linux), and in any desktop computer, regardless the GPII is installed or not.

Section 2 presents an overview of the architectural solutions that support the auto-configuration process in both CLOUD4all and APSIS4all; then, section 3 presents the actual implementation of the auto-configuration capabilities for web browsers using the CLOUD4all and APSIS4all infrastructures. Finally, section 4 presents some conclusions and discuss some future work.

2 Architectural Solutions

As mentioned before, the extensions developed rely on the architectural solutions that support the auto-configuration process in both the CLOUD4all and APSIS4all projects.

[5] http://www.codefactory.es/en/products.asp?id=415
[6] http://www.texthelp.com/North-America/readwrite-family
[7] http://maavis.fullmeasure.co.uk/
[8] http://webanywhere.cs.washington.edu/
[9] http://www.satogo.com/en/

2.1 The CLOUD4all Architecture

The CLOUD4all architecture has been designed to work across platforms and devices, including desktop computers, mobile devices (both feature phones and smartphones) and cloud-based and web applications (Clark, Basman, Zenevich, Mitchell, & Strobbe, 2013). The main components of the CLOUD4all architecture that take part in the auto-configuration process are listed next:

- The **User Listeners** are responsible for starting the auto-configuration process, and provide a set of extensible tools and techniques that will allow users to share their token easily, i.e. via a USB key, a QR code, a NFC card or a text input where users might type their token.
- The **Flow Manager** is the central point of the architecture, and is responsible for orchestrating the personalization workflow.
- The **Preferences Server** is where preferences are stored. Within CLOUD4all, the preferences are formatted according to the ISO-24751 standard, in a platform-agnostic format. No personal information about the users is stored in the preferences.
- The **Device Reporter** provides information about the device (OS, technical features) and about the solutions installed in the platform running CLOUD4all.
- The **Matchmakers** are where the main intelligence of the system resides. The Matchmakers will get information about the preferences of the user, the accessibility features present in the device, the solutions available – taken from the Solutions Registry–, and the context where the interaction is taking place, and will calculate the most appropriate settings for the solution, for that specific user and in the current conditions.
- The **Lifecycle Managers** and the **Setting Handlers** perform the actual launching and setting of applications and access features.

The CLOUD4all architecture is based on web standards, and has been designed to support cloud-level scalability, extensibility and long-term growth. Many of the components of the architecture have been designed to be able to reside locally or in the cloud, or both. Therefore, instances of the flow manager, the matchmakers or the preferences server can be available either on the device, on the cloud or in both places simultaneously. This ubiquity will make it possible to maintain the auto-configuration process even in low-bandwidth connections or no connection at all. On the other hand, modules such as the user listeners, the device reporter and the setting handlers are device-bound, as they provide information about the specific device that will be auto-configured.

2.2 The APSIS4all Architecture

The APSIS4all architecture has been developed with a mix of technologies to support the specific features of each machine. It is distributed both locally (in ATMs and TVMs) and in the cloud, and is based in four main modules:

- **Preference wizard:** a java-based web application that guides users through a questionnaire optimized to maximize the information recovered with a minimum number of questions.
- **Coding system:** Preferences gathered with the wizard are exported from the encrypted database to a XML version of EN 1332-4. They are stored in the cloud and copied to personal contactless cards. Additionally these preferences can be shared with applications with permissions through two protocols: SFTP and XMLHttpRequest.
- **Multimodal interface manager:** a system deployed in ATMs and ticket vending machines able to read cards with preferences coded inside and transform user interfaces appropriately.
- **Monitoring system.** This part of the architecture is in charge of collecting logs and statistical information in order to debug the system and create automatic reports of use.

Whereas the Preference wizard is coded in Java, contactless cards (MiFARE) use private operating systems to store and communicate information with machines. The multimodal interface manager is developed basically in Flash in ATMs while TVMs use a mixture of C++ technologies in the machines and XSLT transformations over theXML UI template to produce the final UIs. Monitoring systems work both locally and in the server-side.

3 Implementing the Auto-personalization from Preferences in Web Browsers

By January 2014, the combined use of Google Chrome and Mozilla Firefox adds a total of the 52.4% of the web browser market share (Awio Web Services LLC, 2014). In Google Chrome, an extension is a zipped bundle of files that adds functionality to the browser[10]. Unlike packaged apps, Google Chrome extensions make it possible to interact with web pages and servers or with browser features such as bookmarks and tabs. In Mozilla Firefox, extensions can add new features to the browser, such as different ways to manage tabs, and they can modify web content to improve the usability of particular websites[11]. Traditional - XUL-based - extensions are more powerful, allowing for instance to change the browser's chrome -, but in most cases they require restarting the browser. On the other hand, add-on SDK (JetPack) extensions provide a set of high-level APIs that permit to develop restartless extensions.

Several extensions providing accessibility features for web pages have been developed and are available via the proprietary extension repositories for both platforms. For instance, Accessibility Evaluator for Firefox[12] allows a page author or user to view a list of navigation landmarks. Google developers have developed Chromevox, a browser extension that implements a screen reader. Most of the extensions currently

[10] http://developer.chrome.com/extensions/overview.html
[11] https://developer.mozilla.org/en-US/Add-ons
[12] http://firefox.cita.illinois.edu

available provide information and cues to web developers for improving the accessibility of their websites (generally checking against WCAG 2.0 guidelines) or focus on specific disabilities (i.e. Chromevox, ChromeVis). The browser extensions presented in this paper will inject CSS and JavaScript into any web pages that are displayed in the browser in order to activate or deactivate the accessibility features of the browser and the available ATs. Moreover, the synchronization capabilities provided by both browsers will permit this personalization to happen in any browser instance with these extensions installed regardless of the platform.

All the personalization process will take advantage of the Cloud infrastructure provided by CLOUD4all, whose main elements were presented in the previous section. The user listener and the lifecycle handler will reside in the background page of the extension. Background pages do not have access to the actual content of the websites, and run in a different process than the content scripts that will inject code in the web pages. The background page will get the token and will scan the defined preferences servers in search for a valid set of needs and preferences. An XMLHttpRequest will be sent to the online flow manager, comprising an object that includes the token of the user and the identifier of the solution. The online flow manager will then make a request to Needs and Preferences Server. If the N&P server stores a set of needs and preferences that corresponds to the token requested, it will respond with an error 500; otherwise, the needs and preferences of the user (in the form of 24751 terms) will be sent to the Matchmakers, that will transform this set into application-specific preferences (ASP). Simultaneously, the XMLHttpRequest to the APSIS4all needs and preferences server will get a set of needs of preferences in EN1334 format or an error 500. I case the request successes, the transformation process will be replicated within the browser. In both cases, the application-specific preferences will be stored using the storage functionalities provided by the browser APIs (storage for Google Chrome, simple-prefs for Mozilla Firefox). The background page will capture any changes in the stored preferences and will communicate them to the content script, that will inject the appropriate CSS or JavaScript to the pages displayed in the browser. The modules involved in the auto-configuration process, as well as the communication amongst them, are presented in figure 1.

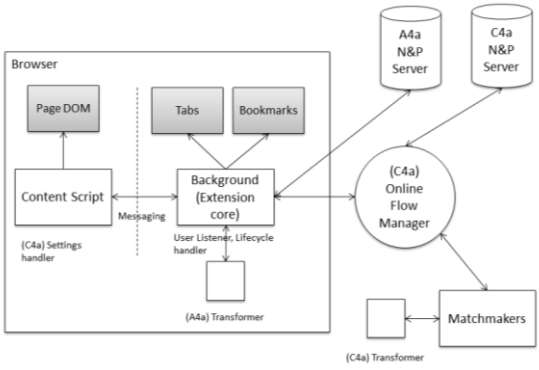

Fig. 1. Arquitectural description of the auto-configuration process

The JavaScript injected in the DOM of the visited web pages will allow us to perform a level of UI remoulding. UI remoulding denotes the configuration of the UI that results from the application of one or several transformations on all, or parts, of the UI (Coutaz & Calvary, 2009), including aspects such as the suppression of UI components that become irrelevant in the given context or the substitution of UI components. In its current state, the extensions support the following customizations:

- Activate / Deactivate the default screen reader (only in Google Chrome).
- Change the font-size of the pages ("medium", "large" and "extra-large").
- Magnification (100%, 200% or 300%).
- Apply high contrast themes ("black on white", "white on black", "yellow on black" or "black on yellow").
- Invert Colors. Applies a CSS filter to invert all colors of the web page.
- Simplifier. The simplifier injects a JavaScript script that removes all unnecessary elements from a webpage and presents them as they will be accessed by a common screen reader.

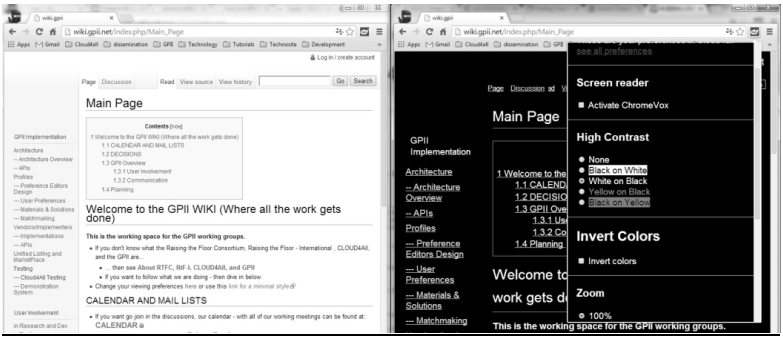

Fig. 2. Google Chrome screen before and after adaptation

The extension core receives the user preferences either in a ready-to-use format (Application specific Preferences) or may need to transform them. The transformer is responsible of taking the terms, expressed in different formats, and transforming them into application-specific terms that can be understood by browser. The extensions are able to interpret and adapt websites based on two user preference formats: EN 1332-4 and ISO 24751 part 2. As mentioned previously, APSIS4all is using the draft revised version of standard EN 1332-4. This European Standard defines the data objects to be stored within an integrated circuit(s) card and exchanged with terminals to identify specific user interface preferences. The preference information may be used by terminals to configure appropriate methods of communicating with the user during a transaction process (IST/17, 2007). CLOUD4all uses a set of common terms based on ISO 24751 part 2 (International Organization for Standardization, 2008). ISO 24751-2 provides a common information model for describing the learner or user needs and preferences when accessing digitally delivered resources or services. The approach is to split needs and preferences into three categories: 1) the display describes the presentation and structure of resources, 2) the control focuses on how resources are to be

controlled and operated; and finally 3) the content describes supplementary or alternative resources to be supplied. The following tables show the rules to translate standards into Application Specific Preferences (ASP). These rules define the way user preferences are used to remould UIs. Table 1 compares tags that are equivalent or similar in meaning whereas Table 2 shows the equivalence took up by the system presented in this paper.

Table 1. Comparison of standard tags and value spaces

ASP		ISO 24751- 2		EN 1332-4		
Tag	Values	Tag	Values	Tag	bits	Values
Screen reader	Boolean	screen reader: usage	required, preferred, optionally use, prohibited	DF61	b1	1/0
Font-size	medium, large, x-large	font size	real(10,4) range (0.0 .. *)excluding (0.0)	DF51	1 byte coded as two BCD digits	the height of characters in millimetres
Magnification	100%, 200%, 300%	magnification	real(10,4) range (1.0 .. *)	DF64	b7b6 b7b6 b7b6 b8b7b6 b8b7b6	01:low 10:medium 11: high 100:very high 101: maximum
High contrast themes	black on white, white on black, yellow on black, black on yellow	foreground colour	RGB plus Alpha	DF52	b4b3b2	000:white,001:red,010:orange,011:yellow, 100:green, 101:blue, 110:purple, 111:black
		background colour	RGB plus Alpha	DF52	b7b6b5	000:white, 001:red, 010:orange, 011:yellow, 100:green, 101:blue, 110:purple, 111:black

Table 1. (*continued*)

Invert Colours	Boolean	invert colour choice	Boolean	-	-	-
Sim-plifier	Boolean	content density	overview, detailed	DF5D	b4b3	01: Simplified text required 10: Very simplified text required

Table 2. Equivalence of standards and value spaces

APfP	ISO 24751-2	EN 1332-4
Screen reader = True	screen reader -usage = "required" OR "preferred"	DF61[b1] = '1'
Font-size = medium	font size < 14pt	'X'< DF51 <'Y'
Font-size = large	14pt < font size < 18pt	'X' < DF51 < 'Y'
Font-size = extra-large	18pt < font size	'X' < DF51 < 'Y'
Magnification=100%	Magnification=1.0	DF64[b7b6]='01'
Magnification=200%	Magnification=2.0	DF64[b7b6]='10'
Magnification=300%	Magnification=3.0	DF64[b8b7b6]='100' OR '101'
High contrast themes=" black on white"	background colour = black AND foreground colour = white	DF52[b7b6b5b4b3b2]='000111'
High contrast themes=" white on black"	foreground colour = white AND background colour = black	DF52[b7b6b5b4b3b2]='111000'
High contrast themes=" yellow on black"	foreground colour = yellow AND background colour = black	DF52[b7b6b5b4b3b2]='111011'
High contrast themes=" black on yellow"	foreground colour = black AND background colour = yellow	DF52[b7b6b5b4b3b2]='011111'
Invert Colours = True	invert colour choice = True	-
Simplifier=True	content density = "overview"	DF5D[b4b3]='10' OR '01'

The notation used in this table is explained with these examples:

- DF5D[b4b3]='10' OR '01' means data object DF5D has the value '10' or '10' it its bits 4 and 3. The APfP system will understand that text simplification is required.
- Screen reader -usage = "required" OR "preferred" means that when attribute usage in container Screen reader takes values "required" or "preferred", then the APfP system activates the screen reader.

Although further work has been done to establish equivalences between EN 1332-4 and ISO 24751-2, this is out of the scope of this paper.

4 Conclusions and Future Work

Browser extensions and apps provide a great opportunity to improve the accessibility of web browsers, as well as to provide personalized access solutions for people with special needs. The auto-personalization capabilities the approach presented in this paper allow us to enhance at least two of the three aspects of web accessibility as defined in (Fernandes, Lopes, & Carriço, 2011): 1) the injection of CSS and JS into web pages provides an enhancement of the accessibility features of the web browsers by allowing the activation of styling features such as magnification or application of UI themes; and 2) the possibility of activating non-out-of-the-box ATs. Regarding the semantics of the web content, the injection of more complex JavaScript algorithms will also permit to enhance the semantics of the pages, both by direct action – i.e. adding WAI-ARIA roles to the appropriate components - or providing recommendations to developers about web practices. Also, more complex JavaScript functionalities will permit UI remoulding by removing elements or adding new ones depending on the preferences of the users. The development of packaged apps in Google Chrome or traditional, XUL-based, extensions in Mozilla Firefox in combination with common applications APIs can be used to enhance the overall user experience when using common applications such as Gmail, YouTube or Picasa, even providing capabilities to adapt the browser's chrome to the needs and preferences of the user.

To sum up, it seems clear that the auto-adaptation of the accessibility features of web browsers has the potentiality to provide access to web content to people with special needs. The capability to read different standards makes the system more compatible and easy to integrate with technologies coding user preferences. Finally, these extensions broaden the scope of APSIS4all project from just public digital terminals to personalize any device where a web browser is running.

Acknowledgements. The research leading to these results has received funding from the European Union Seventh Framework Programme ([FP7/2007-2013]) under grant agreement nº 289016 (CLOUD4all) and by European Union's ICT Policy Support Programme as part of the Competitiveness and Innovation Framework Programme. GA 270977 (APSIS4all).

References

1. Accessibility features in Firefox - Make Firefox and web content work for all users (2014), http://support.mozilla.org/en-US/kb/accessibility-features-firefox-make-firefox-and-we?redirectlocale=en-US&redirectslug=Accessibility (retrieved from Mozilla support)
2. Awio Web Services LLC. Web Browser Market Share (2014), http://www.w3counter.com/globalstats.php (retrieved February 20, 2014)
3. Calvary, G., Coutaz, J., Bouillon, L., Florins, M., Limbourg, Q., Marucci, L., et al.: The CAMELEON reference framework (2002)
4. Clark, C., Basman, A., Zenevich, Y., Mitchell, J., Strobbe, C.: D105.1.1. System Architecture V1. Deliverable, Cloud4all project (2013)
5. Coutaz, J., Calvary, G.: HCI and software engineering: Designing for user interface plasticity. In: Human-Computer Interaction: Development Process, p. 211 (2009)
6. Fernandes, N., Lopes, R., Carriço, L.: On web accessibility evaluation environments. In: Proceedings of the International Cross-Disciplinary Conference on Web Accessibility, p. 4. ACM (2011)
7. Gajos, K.Z., Weld, D.S., Wobbrock, J.O.: Decision-Theoretic User Interface Generation. AAAI 8, 1532–1536 (2008)
8. INREDIS Consortium (2010), INREDIS website http://www.inredis.es/default.aspx (retrieved)
9. International Organization for Standardization. ISO/IEC 24751-2:2008. Information technology – Individualized adaptability and accessibility in e-learning, education and training – Part 2: "Access for all" personal needs and preferences for digital delivery. ISO/IEC, Genève, Switzerland (2008)
10. ISO. ISO 9241-171: 2008. Ergonomics of human-system interaction - Part, 171 (2008)
11. IST/17. BS EN 1332-4: 2007. Identification card systems. Man-machine interfaceCoding of user requirements for people with special needs. BSI (2007)
12. Peissner, M., Dangelmaier, M., Biswas, P., Mohamad, Y., Jung, C., Wolf, P., et al.: D6.4. Interim Report on VUMS cluster standardization. Deliverable, MyUI (2011)
13. SERENOA project Consortium (2013), SERENOA project website: http://www.serenoa-fp7.eu/ (retrieved)
14. Using Google products: How to use accessibility features (2014), Google Accessibility: http://www.google.es/accessibility/products/ (retrieved)
15. World Health Organization. What is e-accessibility? (September 2013), WHO website: http://www.healthinternetwork.com/features/qa/50/en/ (retrieved)
16. World Wide Web Consortium. Web content accessibility guidelines (WCAG) 2.0 (2008), W3C Website: http://www.w3.org/TR/WCAG20/ (retrieved February 2014)
17. World Wide Web Consortium. Web content accessibility guidelines (WCAG) 2.0 (2008)
18. J. Allan, K. Ford, K. Patch, J. Spellman (eds.): World Wide Web Consortium. User Agent Accessibility Guidelines (UAAG) 2.0 (2013), http://www.w3.org/TR/UAAG20/ (retrieved February 19, 2014)
19. Craig, J., Cooper, M. (eds.): World Wide Web Consortium. Accessible Rich Internet Applications (WAI-ARIA) 1.0 (2014), W3C website: http://www.w3.org/TR/wai-aria/ (retrieved February 19, 2014)