Chapter 29
User Interface Adaptation for Accessibility

Sergio Firmenich, Alejandra Garrido, Fabio Paternò and Gustavo Rossi

Abstract In this chapter, we discuss methods and tools for adapting user interfaces to make them more accessible. We introduce the problem of user interface adaptation and characterize different techniques to be adapted to the user interface. We show that there is a broad range of methods and tools to transform existing interfaces to make them accessible. We describe such approaches by grouping them in two types of solutions: those that provide built-in adaptation mechanisms for the application and those which are external to the application.

29.1 Introduction

Adapting a user interface (UI), for example, to make it accessible, implies changing, or adjusting its structure, contents, and/or available actions according to the users' current goals and abilities (including the context of use). This adaptation may be initiated and controlled by the user, or built-in in the application itself or performed by a third party (not the user, not the original application).

The need for UI adaptation has been recognized by Edmonds since the early 80s (Edmonds 1982). The traditional idea that one system fits all is antagonistic toward the special needs or preferences of different users. Even the same user may change her ability regarding the task she performs with the system, and the interface should evolve (adapt) accordingly. Edmonds also introduced the concept of dynamic

S. Firmenich · A. Garrido · G. Rossi (✉)
LIFIA, Facultad de Informática, Universidad Nacional de La Plata and CONICET,
50 y 120 s/n, La Plata, Argentina
e-mail: gustavo@lifia.info.unlp.edu.ar

S. Firmenich
e-mail: sfirmenich@lifia.info.unlp.edu.ar

A. Garrido
e-mail: garrido@lifia.info.unlp.edu.ar

F. Paternò
CNR-ISTI, HIIS Laboratory, Via G. Moruzzi 1, 56124 Pisa, Italy
e-mail: fabio.paterno@isti.cnr.it

Y. Yesilada and S. Harper (eds.), *Web Accessibility*, Human–Computer Interaction Series, https://doi.org/10.1007/978-1-4471-7440-0_29

adaptation or self-adaptive interfaces, i.e., those which do not need the intervention of the developer or the user to perform the adaptation.

We are accustomed to different degrees of adaptation in the interfaces we regularly use. A simple example is the Windows start menu, which changes its contents dynamically according to the most (recently) used applications. Amazon adapts the contents presented to each user in relation to their browsing and shopping story, adjusting the recommended products in their home page and in every sub-store. It also adapts forms (e.g., to perform the check-out process) according to the information it has about the user (e.g., frequently used address, check-out preferences, etc). Email applications (Google, Yahoo, etc) let end-users change the structure, look and feel, and available operations of their site.

When dealing specifically with accessibility, different factors might impact on the need to adapt the UI. In the past, research work has focused on user-related factors such as perceptual skills, motor or sensing abilities, preferences, emotional state, cultural and education issues, in addition to the ability of the application to support users in their task, and afford to adapt regarding the user acquired experience. However, the emergence of mobile computing and the possibility of using application software in different contexts brought other factors into consideration such as those related with technology (screen resolution, connectivity, battery life, etc.) or the environment (location, noise, etc) (Paternò 2013). In any case, just considering the myriad of different requirements for accessibility related to specific motor or sense abilities let us conclude that adaptation is a must.

There are many considerations to take when building adaptation in interfaces for accessibility, and many dimensions to classify them. We next summarize some of the most important topics related to the general problem of adaptation, and the rest of the chapter will discuss some of the peculiarities of each approach.

- Who configures the adaptation: There may be coarse-grained interface variants, for example, for a particular disability, which is configured during design time. Alternatively, the interface may be self-adaptive, i.e., it learns about the user's needs dynamically, or the user may configure the adaptation herself.
- What is adapted: According to Brusilovsky (2001), a Web interface may be adapted regarding its structure, contents, and/or links. We may refine this coarse-grained classification considering, for example, what is adapted regarding the contents' presentation: it may be its media transforming text into audio (as in screen readers) or other properties such as size and colour (of text or images), volume (audio), etc.
- How we represent the user model: A critical issue is the representation of the systems knowledge about the user and her context, including preferences, abilities, device, environmental context, social context, etc. This representation must be expressive enough to capture all the information needed to perform the adaptation, and it must be dynamic in terms of both the information and its structure. Additionally, the user model may be deduced from the users actions or built by the user by configuring some options.
- When adaptation occurs: Assuming that the interface changes automatically in response to its experience with users, we must decide the rhythm of change. This

decision is not minor since, for example, changing too often might affect stability of the interface and therefore comprehension and usability.
- Where adaptation occurs: The adaptation may occur inside the system or may be external and performed by a third party or application built explicitly to fulfill this purpose.

Each one of these issues requires more than a book chapter, but for the sake of clarity and conciseness, we will address only some of them and provide pointers to others. The next section introduces a classification of User Interface Adaptation types, which includes a brief revision of existing literature.

29.2 Classifying Adaptive Interfaces

There are many different classifications in the literature for UI adaptation. One of them distinguishes between adaptable versus adaptive systems (Stephanidis and Savidis 2001). In the case of adaptable systems, end users have the capability to adapt the UI to their needs, i.e., users are in control of the adaptation, whereas adaptive systems have internal mechanisms to directly perform the adaptation, with little or no control from users. Other classifications exist that categorize the involvement of the user versus system at different stages of the adaptation, like Dieterich's taxonomy (Dieterich et al. 1993) and the recent PDA-LPA taxonomy (Bouzit et al. 2017), which provides a fine-grained characterization of end-user involvement versus system self management with respect to Perception, Decision, Action, Learning, Prediction, and Adaptation.

While the above are relevant classifications, they tend to leave out coarse-grained architectural differences that have appeared with recent technological innovations. A similar argument can be made about McKinley's taxonomy (McKinley et al. 2004), which considers three dimensions: How to adapt, Where to adapt, and When to adapt, but does not provide insight into the design and implementation of different adaptation techniques (Bouzit et al. 2017).

Another classification divides adaptive systems from the point of view of the development approach, in window managers and widget toolkits on the one hand, and model-driven engineering on the other hand (Akiki et al. 2014). Thus, this classification misses adaptive frameworks. Furthermore, although several approaches exist to create adaptive Web applications for accessibility, other approaches have emerged to allow users to adapt their preferred Web applications even beyond what these applications support.

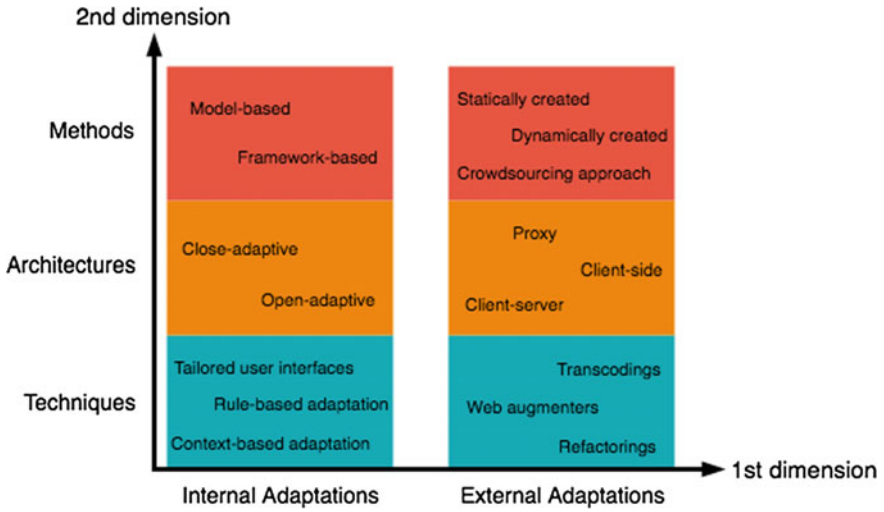

Fig. 29.1 Classification of user interface adaptation for accessibility

Based on the above discussion, we propose a classification structured into two dimensions (see Fig. 29.1). First, a coarse-grained partition between two broad types of adaptations:

- Those which are built in the original system, together with those that may be added because the original system provides some infrastructure to allow for new adaptations. In general, we may say that these are adaptation-aware systems, because the system was constructed to be able to perform some adaptations. Instances of this category are model-driven adaptive systems. We call them **Internal Adaptations**.
- Those which are external to the system, i.e., the original developers did not create a system with adaptation capabilities but the system is adapted from the outside by third-party software artifacts, or with techniques that intervene at a later stage, from which the Web application is unaware. Instances of these techniques are transcoding (Asakawa and Takagi 2008), augmentation (Bigham 2007), and refactoring (Garrido et al. 2013). We call them **External Adaptations**.

The second dimension in our classification aims at characterizing the different approaches in each partition with a finer grained definition with regard to:

- the **technique** by which the adaptation mechanism is activated;
- the **architecture** that establishes adaptation mechanism constraints;
- the **method** for developing the adaptation.

The next two sections analyze, correspondingly, internal and external UI adaptation approaches for improving or supporting Web applications accessibility.

29.3 Internal Approaches for UI Adaptation

There has been a number of interesting contributions in the area of methods and tools for accessible UIs adapted to people with various disabilities. This is an important area since there are many people with disabilities that can only access their applications through assistive technology, and they need adapted versions of their applications in order to accomplish their tasks with them.

29.3.1 Techniques

There are two main types of techniques to activate internal adaptations: tailored application versions and rule-based adaptive solutions. The former is mainly used at design time or at the beginning of a user session so that users (in some cases with the support of developers) can directly select one of the versions available or configure the desired version. The latter is more often used to obtain adaptive solutions where the applications modify some parts depending on dynamic contextual events, whose occurrence trigger specific rules that can change UI aspects. We can add a further technique: context-aware (Run-time adaptivity, system-initiated), which is related to run-time adaptivity, i.e., the context and all user activities are captured as the user interacts with the system, and the system acts accordingly.

29.3.1.1 Tailored User Interfaces

One of the first contributions in this area has been the result of the AVANTI project (Stephanidis et al. 1998), which aims to adapt Web applications in terms of content, navigation, and presentation for people with disabilities. It classifies users to different stereotypes, and, accordingly, it presents optional content and chooses appropriate information from alternatives.

In model-based approaches (Paternò 2005), the basic idea is to start with logical descriptions and then derive implementations for the target platforms and users. In the human–computer interaction area, the CAMELEON reference framework (Calvary et al. 2002) was introduced to distinguish the various possible abstraction levels in describing UIs: task and domain models, abstract UIs (the interaction is described independently from the possible modality used), concrete UI (the interaction is described dependent of some specific modalities but independent of specific implementation languages). The Supple system (Gajos et al. 2006) is an example of model-based system. It has focused on automatically generating UIs at design time for people with disabilities from logical descriptions taking into account device, tasks, preferences, and abilities. The UI generation is organized as a discrete optimization problem solved by using a branch-and-bound algorithm. The Supple authors focused on how to exploit Supple in order to support disabled users, for example, by auto-

matically generating UIs for a user with impaired dexterity based on a model of her actual motor abilities. The authors have carried out laboratory experiments that indicate positive results in terms of speed, accuracy, and satisfaction of users with motor impairments.

A different approach has been adopted in The Global Public Inclusive Infrastructure (GPII), proposed by Vanderheiden and Treviranus (2014), which is an infrastructure aimed at automatically providing disabled users with solutions able to enhance their interaction with different public services. For instance, someone needing to access an inaccessible service, in a specific moment and in a concrete place, can obtain an accessible interface from GPII. To accomplish this process, users store their preferences in a local device or in the cloud. Subsequently, they carry out the login process wherever they are and GPII provides them with a tailored UI and the required assistive technologies. This seems useful but limited support since it can just provide access to a few predefined versions or configurations of the application, while the wide variety of possible user characteristics and preferences as well as contexts of use call for more flexible adaptations, which should be able to provide changes in the UI at various granularities.

29.3.1.2 Rule-Based Adaptation

In rule-based approaches, the rules generally indicate some events or conditions that should trigger the consequent adaptation. Miñón et al. (2016) have investigated how to exploit such rules in the model-based generation of accessible UIs. The adaptations can be applied in any of the CAMELEON abstraction levels at design time. For instance, adaptation rules related to task sequencing should be considered at the task and domain level, whereas adaptation rules related to some specific UI modalities have to be considered at the concrete UI level. At run-time, the solution proposed involves obtaining the necessary level of abstraction by means of an abstraction process in order to apply adaptation rules when a change in the context occurs, and then generate again the final UI. Ghiani et al. (2014) have put forward a proposal for obtaining run-time adaptation able to support dynamic reverse engineering of Web pages in order to obtain their model-based description, and then generate an implementation using different modalities more suitable for the new context of use. Yang and Shao (2007) have introduced the use of an expert system, JESS (Java Expert System Shell), in managing the adaptation rules. It uses a special algorithm called Rete to match the rules to the facts, which should be faster than a simple set of cascading "if. . .then" statements in a loop. A JESS rule is defined in such a way to trigger actions after matching knowledge base patterns. The adaptation knowledge base consists of a fact base, i.e., context profiles, and a rule base, i.e., adaptation rules. W3Touch (Nebeling et al. 2013) has not used model-based languages for supporting Web pages adaptation. For this purpose, it considers user interaction, in particular, the occurrence of missed links or frequent zooming as indicators of layout issues, however the adaptation rules supported do not consider the use of multimodality. In this case, the adaptation rules can be defined based on the logged events and

may only be restricted to specific devices and viewing conditions. The possibility of using rules to specify the desired adaptations has raised interest in environments able to support the specification of such rules even by people who are not professional developers. An example of tool addressing this topic is TARE (Ghiani et al. 2017) that aims to provide an easy to understand way to specify contextual events and conditions as well as the corresponding actions, which can modify the application or even the state of surrounding appliances (e.g., lights, radio). This tool has been used in projects aiming to support elderly, thus giving them or their caregivers the possibility to personalize their applications according to specific situations.

29.3.1.3 Context-Based Adaptation

The increasing availability of various types of sensors, both personal and environmental sensors, has made it possible to detect useful information concerning the context of use in which users are interacting with the application, and then adapt the UI accordingly. The possible contextual aspects can involve the user (the tasks to perform, the emotional state, the current disabilities, etc.), the environment (e.g., light, noise, position), and the technologies (the available devices, appliances, and objects). Such technological evolution has also stimulated the development of solutions exploiting multimodal UIs, in which the adaptation can involve different communication channels between the user and the system. An example of accessible solution in this area is provided by Ghiani et al. (2009), who support the blind by exploiting the haptic channel as a complement to the audio/vocal one. It includes vibrotactile feedback enhancement for orientation and obstacle avoidance obtained through the use of unobtrusive actuators applied to two of the user's fingers combined with an electronic compass and obstacle detector sensors connected wirelessly to the mobile device. The user localization is obtained with the support of RFID tags associated with objects of interest. Later, Ahmetovic et al. (2016) proposed a smartphone-based system that provides turn-by-turn navigation assistance based on accurate real-time localization over large spaces. In addition to basic navigation capabilities, the NavCog system also informs the user about nearby points-of-interest (POI) and accessibility issues (e.g., stairs ahead). For this purpose, the system makes use of a network of Bluetooth low energy (BLE) beacons to localize the user with an approach based on the K-nearest neighbor (KNN) algorithm.

In this perspective, Hussain et al. (2018) support rule-based adaptivity by collecting real-time data from multimodal data sources, e.g., smartwatch, mobile phone, camera, Kinect. It aims to generate the UI at runtime, without redeploying the system, and with the help of authoring tools, new rules are added without affecting the running system. Additionally, the adaptation on UI is made when the context is changed, which is observed by implicit and explicit (user feedback) ways. It also aims to receive user feedback: the implicit feedback is acquired from the user behavioral responses, which are collected automatically when users interact with the system, while the explicit feedback is acquired through questionnaires. However, currently the rule authoring is able to manage only basic-level adaptation rules.

29.3.2 Architectures

The support for adaptation can be obtained through different architectures. According to Sottet et al. (2007), a system is close-adaptive when adaptation is self-contained. It supports the innate adjustments planned at the design stage as well as new adjustments produced by its own internal learning mechanisms. The system is open-adaptive if new adaptation plans can be introduced during run-time. Thus, in close-adaptive systems rules are prefixed. Adaptation rules are designed at development time. Whenever a new rule is to be added, the system needs to be redeployed. This is the case for MyUI (Peissner et al. 2012) in which relevant interaction patterns are identified for the target users and devices, but if the targets change then the UI parametrization and preparation needs to be performed again before deployment. In order to obtain more open solutions, Miñón et al. (2016) propose an architecture in which there is an autonomous Adaptation Integration System, which applies adaptation rules to model-based descriptions of the interactive application. The rules consider user disability (cognitive impairment, motor impairment, deafness, etc.) and various granularity levels (single element, group element, presentation, application). Lastly, the adapted UI is generated. The resulting adaptation process tends to be slow and not very flexible. An efficient and flexible architecture for open-adaptive solution is presented by Ghiani et al. (2017). It is based on rules repositories and a middleware able to detect dynamic events in the context of use. Adaptation rules associated with a given application can be added and executed at any time. They are provided to an architectural component called adaptation engine in a trigger-action format. The adaptation engine subscribes to the underlying middleware (Context Manager) in order to be notified when relevant events occur. In this case, the corresponding actions are transmitted to the application for actually performing the desired adaptation to its UI or the state of some connected appliances. This type of architecture has then also been adopted in Hussain et al. (2018), which also considers the use of models for context, user, and device. In this approach, in the offline phase of adaptive UI design, all the relevant models are built and the adaptation rules are generated using a rule authoring tool. The created rules subscribe to the relevant events in a context evaluator. Then, the monitoring module is responsible for data collection while user is interacting with the system through different sensors and trackers (e.g., facial, vocal, eye, and analytics). The evaluator component evaluates the acquired information and decides whether adaptation is required on the UI or not.

29.3.3 Methods

In this section, we discuss the methods proposed for development of adaptable UIs for accessibility. A first distinction can be made between approaches using some model-based descriptions of the UI and framework working directly on the Web implementation.

29.3.3.1 Model-Based Methods

In model-based methods, we can distinguish two types of approaches: static and dynamic. In static model-based approaches, there is the possibility to provide some relevant model-based description and then generate the implementation version for the target users. In the dynamic approaches, the use of models can be updated according to some contextual dynamic change in order to obtain updated implementation without having to deploy again the application. Examples of the static approaches are MARIAE (Paternò et al. 2011) and Supple (Gajos et al. 2006). Supple automatically generates UIs, taking as inputs device-specific constraints, such as screen size and a list of available interactors, a typical usage trace, a functional specification of the interface, which describes the types of information that need to be communicated between the application and the user, and a cost function. The goal is to automatically generate, ability-based UIs that should significantly improve speed, accuracy, and satisfaction of users with motor impairments compared to manufacturers' default interfaces. Miñón et al. (2016) propose to make the model-based development more open to dynamic environments. The basic idea is that when some dynamic contextual change occurs, a model-based description is provided to an adaptation integration system, which is able to access a repository of adaptation rules, apply them to the model-based description, which is then passed again to the tool for model-based implementation generation. A solution aiming to obtain adaptation in terms of interaction modalities with the support of model-based descriptions is proposed by Ghiani et al. (2014). The goal is to overcome some limitations of responsive design (Marcotte 2011), which is able to consider only changes in device resolution and orientation and supports only graphical UIs. An approach aiming to obtain dynamic adaptation with the support of models is in Hussain et al. (2018). It considers context, user, and device models. It needs an offline phase during which all the relevant models are built and the adaptation rules are generated using a rule authoring tool. The rules are then applied during actual use of the application.

29.3.3.2 Framework-Based Methods

Current frameworks (for example, Bootstrap) mainly provide support for adaptation according to the responsive design approach, which support adapting to various device features through fluid layout and stylesheets. They also provide the possibility of associating various visual attributes with groups of devices identified by some features detected through media queries. However, such adaptations are too limited for supporting accessibility because they do not consider the many types of contextual events that can influence user interaction and the various possible user disabilities. The context toolkit (Salber et al. 1999) was among the earliest supports for developing context-enabled applications by providing a library to facilitate integration with sensors. It initially considered a limited set of events and led to meld the context awareness code with the application. Later, the Context Toolkit has been augmented with support to facilitate development and debugging of context-dependent

applications (Dey and Newberger 2009). However, such approaches mainly refer to providing changes in the appliances states as a consequence of the detected events, and pay little attention to UI adaptations. W3Touch is an interface instrumentation toolkit for web designers to collect user performance data for different device characteristics in order to help them identify potential design problems for touch interaction. Web designers can visualize the data aggregated by W3Touch and use simple metrics to automate the adaptation process for many different viewing and interaction contexts.Thus, it provides a more flexible support but still without considering accessibility issues with particular attention. To facilitate the development of frameworks able to address such issues the W3C has developed the WAI-ARIA standard (WAI-ARIA (W3C) 2019), which helps with dynamic content and advanced UI controls developed with Ajax, HTML, JavaScript, and related technologies. Further aspects about dynamic content concerning accessibility may be found in chapter "Dynamic Web Content" in this book.

29.4 External Approaches for UI Adaptation

This category, as explained earlier, belongs to approaches for adapting a system from the outside, i.e., the target system is unaware of the adaptations. The benefits of these approaches are mainly that they may be applied to any Web system, i.e., the system does not need to be constructed in any particular way or depending on any infrastructure (which simplifies development) and often provide the final users with the possibility to control the adaptation and personalize it. In the context of accessibility, however, users controllability may not always be an advantage, since depending on the disability, it may require the assistance of third persons, like family members or caregivers. Adapting third- party Web contents is an old idea that has been applied in very distinct ways, and may involve end-users alone or contemplate a volunteer role.

29.4.1 Techniques

In this section, we will talk about two mainstream techniques for manipulating existing and third-party Web content: traditional transcoding and client-side adaptation.

Both client-side adaptation and transcodings are very powerful and have been applied with very similar goals in some approaches, while very different in others. Even the terms are used indistinctly sometimes and also combined, such as client-side transcoding, or browser-side transcoding (Díaz and Arellano 2015). Originally, transcodings systems were defined as those that transcode a resource before it is delivered to their target client (Asakawa and Takagi 2008). Once the resource (mainly an HTML page together with CSS and JavaScript) is delivered to the client, it behaves as usual, meaning that even if the Web content was adapted (transcoded) by an intermediary server, once it is loaded and rendered on the Web Browser, it is still a normal Web site, and the adaptation mechanisms are restricted to this situa-

tion. Client-side adaptation approaches, on the other hand, manipulate the content after it is loaded and rendered in the client, because these techniques manipulate the actual DOM that Web browsers create for the loaded HTML pages. Client-side approaches bring new opportunities in comparison with intermediary servers. This architectural difference (architectural aspects for external adaptation approaches are discussed on Sect. 29.4.2), directly impacts on how the adaptation mechanisms are defined and triggered, and subsequently on how easy it is to apply some adaptation technique/method. With this in mind, we separate this subsection into transcoding systems and pure client-side adaptation. For each of these techniques for manipulating existing Web content, we discuss their main uses and scope.

29.4.1.1 Transcodings

The problem of improving accessibility by adapting third-party Web content was first tackled by approaches inspired in transcoding systems. Transcoding was defined as a system that transforms content or a program on the fly at an intermediary server, resulting in other formats; this served, for instance, to change the content encoding on the fly using a proxy (Asakawa and Takagi 2008). The same idea of using an intermediate server to manipulate existing content was applied to improve accessibility of third-party websites. According to Asakawa and Takagi (2008), transcodings methods (text magnification, content reorder, page simplification, etc.) are applicable on an intermediary server (proxy) and also directly at client-side by using client-side adaptation scripts. In the same article, Asakawa established two main techniques for transcodings, which are the use of annotations and simplification based on differential analysis.

Basically, these accessibility transcodings act like transformation functions that, once a target UI element is specified, apply a transformation method. Content annotation was and still is a widely used technique for deciding which transformation method to apply over which UI elements for a given Web page. Approaches around this idea were very well described in a previous chapter focused specifically on transcodings (Asakawa and Takagi 2008). Since that time, there have been new works on transcoding tackling different problems. For instance, some works have explored other ways to do annotations via CSS. Other works have taken advantage of the collaborative nature of annotation-based systems, which allows a whole community of end-users to create and share annotations (Takagi et al. 2008, 2009). In some cases, these kinds of approaches may involve a volunteer role, coined to create the annotations when these require some technical skill.

Web applications became more complex at client-side, for instance, by incorporating asynchronous content load and later RIA concerns. In the context of Accessible-Rich Internet Applications (ARIA), some approaches proposed to incorporate RIA functionalities as a new application for transcoding (Lunn et al. 2009; Brown and Harper 2013).

Several aspects related to this technique are discussed in the chapter “Document Engineering” from this book.

29.4.1.2 Client-Side Adaptation

With the evolution of client-side Web technologies, another approach that has emerged to adapt existing third-party Web contents (perhaps the most popular in terms of actual users) is the one based on Web browser extensibility, that enables third-party client-side adaptation based on Web content manipulation once it is loaded in the Web browser, without previous intervention of any proxy server. This idea of extending the Web browser for adapting Web pages was stated as Web augmentation several years ago (Bouvin 1999), and newer literature reinforce this idea and define accessibility as a dimension to improve existing Web sites through the use of Web augmentation software (Díaz and Arellano 2015).

Despite that transcodings can be performed at client-side, other contributions for external adaptation for improving accessibility, in particular those based on client-side, are not easily classifiable into the categories Asakawa defined in the context of traditional transcodings for accessibility (Asakawa and Takagi 2008). New and often used techniques such as user interaction analysis, eye tracking, etc., are tied to client-side adaptation, because these mechanisms need to work when the Web site is already in use (i.e., once it is loaded, parsed, and rendered on the Web browser), and not before it is delivered to the client, as it occurs in proxy servers.

For instance, Hanson and Crayne (2005) discussed how older end-users may personalize their web browsing activities by applying client-side adaptations. Another similar approach, called Farfalla (Mangiatordi and Sareen 2011), similarly proposes to augment Web pages with a toolbar that let end-users customize some aspects of content presentation such as magnify text, change font size, etc. Also this kind of adaptations could be applied automatically if the client-side component may read a user profile from which it takes the information to make adaptation decisions (Peinado and Ortega-Moral 2014). Renarration UI (Prasad et al. 2017) is another similar client-side approach, that also offers a fixed set of transformations the user may perform over the Web sites s/he is visiting.

Other approaches propose an architecture that serves to install artifacts created by the community, instead of offering a fixed set of available transformations. For instance, Accessmonkey (Bigham 2007) proposes a weaving engine together with an authoring tool that, correspondingly, let end-users install and create scripts that are run in the Web browser.

The case of Accessmonkey is a specialization of a general-purpose engine called Greasemonkey, which executes JavaScript scripts when a specific (or a set of) URL is loaded (Pilgrim and Mark 2005). In both cases, however, the reason for the creation of a new adaptation artifact is a non-satisfied users need or preference.

Another approach, called Client-Side Web Refactoring (CSWR) (Garrido et al. 2013), is similar in terms of how the adaptation is performed (through DOM manipulation), but it is different in terms of its motive. In the case of CSWR, the transformations of UI elements are driven with the philosophy of well-known code refactoring (Fowler and Beck 1999). This means that these transformations are motivated by accessibility "bad smells" and the result must guarantee that the original application functionality is still available.

Finally, client-side approaches that work inside Web browsers bring new possibilities. For instance, Puzis et al. (2013) propose an automation assistant for people with vision impairments. This work is interesting because the automation assistant is an agent that adapts the interaction with Web applications, which is very important today, given the complex business processes behind Web applications. The approach proposes a model that uses the navigational history and the current Web site to predict browsing actions (such as filling a text input, or click a button).

29.4.2 Architectures

The techniques described above are mainly deployed using at least one of the following architectures:

- Intermediate Proxy Server: In this architecture, the transformation machinery is hosted in a proxy server that transforms the content delivered by the application server before this content reaches the client, i.e., the user's Web browser. Necessarily, Web browsers must be configured to work with the desired proxy. Extensibility in this approach is achieved, mainly, by annotation. Examples of approaches using this architecture, which are mainly transcoding systems, are SADIe (Lunn et al. 2008), Social4All (Crespo et al. 2016; Takagi et al. 2009; Asakawa and Takagi 2000).
- Client-Side: The client is any software able to communicate with a Web server and rendering the HTML responses. According to this definition, specialized Web browsers are considered a client-side mechanism, even if they apply static transformations, such as the emblematic IBM Home Page Reader (Lunn et al. 2008). This is a good example of a transcodings system without using an intermediate server. However, most of the latest works on client-side external adaptation rely on well-known standards Web browsers, because they have high user adoption, and allow very powerful extensibility mechanisms through which Web content transformation is very easy to achieve. A Web browser extension is aware of every event happening during the navigation session. In this way, when a Web page is loaded, the extension recognizes this event, and it is able to manipulate the loaded DOM to change it. By altering the DOM at client-side, users perceive the Web page adaptation. Examples of approaches using this architecture are Farfalla (Mangiatordi and Sareen 2011), Accessmonkey (Bigham 2007), CSWR (Garrido et al. 2013), Social Accessibility (Takagi et al. 2008).
- Client–Server: there are several approaches, such as Social Accessibility, that propose collaboration among users and volunteers. Also, it is common to require a user profile to make it available from every user's devices (Hanson and Richards 2005). In this way, although stand-alone client-side components are enough to perform the adaptation, it is not enough to contemplate every concern behind the problem of making the Web more accessible, such as collaboration, crowdsourcing, profiling, etc., which are important concerns to be considered. This is the reason

for the existence of client–server architectures, in which the client part performs the adaptation but consumes services provided by the server counterpart to achieve its goals.

A priori, it seems that the power of HTML transformation (or its run-time version: DOM manipulation) of intermediary proxies and client-side adaptation is equivalent, and, in some way, it is true. Thus, although technically almost any UI transformation could be made in any of these architectures, what is not equal is when and under what stimulus or events the alteration is made. This is crucial nowadays, because Web 2.0 and RIA applications make it difficult to transform the whole UI without contemplating user's behavior, just because the content delivered to the client-side not necessarily contains all the UI, but contain a basic layout that will be populated at client-side asynchronously. This problem was reported before (Hanson and Richards 2005), where the authors describe that obtaining a trustable version of the UI (which is the input for the transcoding process) in a proxy server is very complicated given the dynamism with which the UI is composed. Besides this aspect, in other cases the authors say it is directly impossible because of the use of SSL connections.

While dealing with dynamic Web sites (those fully interactive Web sites using CSS, HTML, and JavaScript) is difficult through an intermediate server, this is straightforward in the case of client-side architectures, because these approaches are mostly based on DOM manipulation. When the adaptation is performed at client-side, any aspect of the user interaction may be easily used as part of the adaptation system. This aspect is mandatory for some approaches like refactoring, in which the "bad smells" may be detected automatically by analyzing user interaction at the client-side (Grigera et al. 2017). Another interesting aspect is composition. In client-side approaches, several end-user tools may be integrated, for instance CSWRs may be used for structural and behavioral adaptations but combined with Farfalla (Mangiatordi and Sareen 2011) to adapt other aspects of content presentation, such as color schemes. In the case of intermediate servers, the configuration in cascade of several servers is hard to achieve.

Finally, these architectures may be analyzed also from the point of view of openness. Often they are a natural environment for installing (by plug and play) new kind of adaptation artifacts and for authoring processes. The use of visual tools for content annotation or UI transformation is based on the interaction between users or volunteers with Web content, in some cases, applying changes on the fly (Garrido et al. 2013); then running the end-user created artifacts at client-side is very convenient.

29.4.3 Methods

In this dimension, we discuss the methods in which the adaptations are created or built into the adaptation system. That is, on the one side there are adaptations statically created into the system and later provided by way of a fixed menu of options, and on the other side of the spectrum there are no adaptations provided statically but all

of them are created dynamically by volunteers or end-users. In the middle, we may find a range of hybrid methods which provide some adaptations but leave the door open to receive new ones. Thus, we could also say that this dimension is about the openness of the adaptation system.

Among augmentation systems which are closed to new adaptations we may cite the work of Chung et al. (2013) for deaf people, the work of Hanson and Crayne for older adults (Hanson and Crayne 2005), and the more recent Farfalla project (Mangiatordi and Sareen 2011). In the first case, Chung et al. propose an algorithm to simplify the grammatical structure of complex sentences in news articles to make them easier to understand by deaf people, in addition to showing a graphical representation of the relationships among sentences (Chung et al. 2013). In the case of the Farfalla system, which is an active project similar to the older work of Handon and Crayne, there are a fixed number of adaptations provided in a sidebar menu for the user to control: font size, contrast and color combination, mouse pointer size, capitalized text for easier reading and on-screen keyboard (Mangiatordi and Sareen 2011). Nevertheless, Farfalla is an open-source project that invites for participation, so volunteers could actually add more adaptations by coding them in the Farfalla source code.

There are many examples of open augmentation systems, for instance, Accessmonkey (Bigham 2007). With respect to the transcoding technique, it is specially suited for an open adaptation method, that is, a mechanism to add semantic annotations into the transcoding system. The reason is that semantic annotations are very tight to the particular web application being adapted, so the cost of creating an scalable transcoding system is not affordable by a single group of people. Although there are some proposals to add annotations automatically from CSS classes (Lunn et al. 2008), or automatic transcoding of images into text (Bigham et al. 2006), they did not prosper since automatic methods have accuracy limitations (Takagi et al. 2008). Instead, from their early works, the research group of Chieko Asakawa created authoring tools for volunteers to add annotations to their transcoding system (Asakawa and Takagi 2000). The annotations thus created are added into an annotation database organized by target URL. Other transcoding systems that rely on external annotations are Dante Yesilada et al. (2004) and WebAdapt2Me (2019).

It is worth to note that when an adaptation system has to rely on users to grow, it must necessarily provide a simple and possibly visual interactive tool to make the task very easy and promote adoption among volunteers. Takagi et al. discuss the advantages of open transcoding systems, and present the "Social Accessibility Approach" (Takagi et al. 2008). These authors take a step further by adopting a crowdsourcing approach, i.e., calling the entire community of users to create annotations by providing them with a collaborative authoring platform. Another tool that proposes the use of a crowdsourcing platform if Social4All (Crespo et al. 2016). Using the Social4All platform, volunteers create adaptation profiles for any website, each profile containing a set of adaptations to solve WCAG issues. Last but not least, crowdsourcing has also been proposed in the context of refactoring systems (Garrido et al. 2017). In this case, a crowdsourcing platform is proposed not only for creating new adaptations (applying CSWRs), but also for users to report bad smells manually or collect them automatically, and for the crowd to evaluate the effectiveness of solutions.

29.5 Discussion

Though not stated explicitly in the previous sections, external and internal approaches to UI adaptation also have a difference in the role of end-users in the process of building the adaptable/adaptive interface. In internal approaches, the burden of designing and implementing the adaptation machinery (be it in the form of rules or other different approach) often lies on developers, even if recently some work to enable non professional developers to specify their personalization rules has been put forward. Meanwhile, in some external approaches for adaptation, end-users (not directly involved in the design of the target application) might help in the process through crowdsourcing. In internal approaches, as explained in Sect. 29.3, designers are profiting from long software engineering and user modeling experience in the construction of flexible systems, which can be either seamlessly modified at design time, or can adapt dynamically when the context changes. User interfaces are certainly one part of the system and flexibility in UIs, e.g., for improving accessibility, is a good example of the impact of modularity in system design. In order to limit the effort in designing adaptation in internal approaches, there have been recent proposals aiming to allow even people without programming experience to provide the desired adaptation rules.

External approaches, meanwhile, are relatively new. Specially, the growth of client-side adaptations could not be predicted 10 years ago when the future seemed to bring the growth of proxy-based solutions (Asakawa and Takagi 2008). While transcoding-based approaches have some years now, the increasing and overwhelming growth of social networks have made end-users much more aware of their own (for example, accessibility) problems; these problems are not only shared between them but they are also involving themselves in finding solutions, e.g., via crowdsourcing. This involvement, which is also pushed by the popularity of end-user approaches, puts also some pressure on the improvement of internal approaches and on designers themselves, since it shows that those features not originally provided by designers can be eventually added by the end-users.

Something that the literature is still missing, to the best of our knowledge, is a thorough experimentation on very important aspects like user adoption, real coverage of user needs, and also a comparison of the effectiveness of the different approaches discussed in this article. We consider this a crucial task in the near future.

29.6 Future Directions

Even though UI adaptation is a consolidated topic in the literature, there are still areas in which research is needed. Some of them are mentioned here.

- Regarding model-based approaches for internal adaptation, one problem that has hindered part of their popularity is the relative low penetration of model-based and model-driven development in industry. This topic has been extensively discussed

elsewhere (Whittle et al. 2014). Better and more stable tool support might help these approaches to gain penetration. Better training and education is needed (as in other fields related more directly with accessibility).

- In Framework-based approaches for adaptation, there is also a growing interest to include accessibility issues. For example, the accessibility plugin for the Bootstrap framework (BootstrapAccessibilityPlugin 2019). Yet, covering the broad possibilities of adaptation for accessibility is a missing issue in Web development frameworks (not only considering adaptation as a target issue).
- Regarding external approaches, most of them share a complex weakness, which is the evolution of the source Websites. All external approaches maintain some kind of reference to the targets UI elements that will be adapted. When the Website changes, these references may become old, and the adaptation mechanism may not work. Since authoring tools are becoming a common place for scripts or annotation creation, it is important also to support the maintenance of artifacts, and not just their building. We believe that automatic or semi-automatic testing and end-user driven maintenance must be faced both at methodological and at technical level.
- Though not explicitly discussed in this chapter, new interaction techniques (like those based on gestures or eyes gaze) are beginning to gain momentum for improving accessibility (Kumar et al. 2017). However, little work has been done on adapting the interaction technique to the needs of the end-user (see, for example, Yoda 2018). Moreover, there is a bunch of work in gesture recognition within the field of robotics and rehabilitation (see, for example, Lin et al. 2017). The combination of adaptive interaction techniques with other technologies such as the Internet of Things (see Chapter "Internet of Things" in Part 6 of this book) will leverage existing techniques.
- Finally, the extensive application of Artificial Intelligence (AI) techniques (such as machine learning) will have an impact in UI adaptation. In fact, rule-based approaches like those discussed in Sect. 29.3.1.2 have their roots in the work on expert systems in the early 90s. Abou-Zahra et al. (2018) discuss different aspects in which AI will improve Web accessibility and particularly, interface adaptation. For example, natural language processing may be used to allow text adaptation (e.g., simplifying text) for people with cognitive disabilities. Besides, AI might help to better learn the preferences and needs of people with changing conditions and therefore help in content adaptation. Related with this last trend, Galindo et al. (2017) present an approach to provide UI adaptation driven by emotions. They also use a rule-based adaptation engine which interacts with an inference engine to detect the actual user emotion.

29.7 The Author's Opinion of the Field

A disability is an impairment that may be cognitive, developmental, intellectual, mental, physical, sensory, or some combination of these. Such impairments may impact the way how people can interact with Web applications in different ways. Thus, it

becomes crucial to provide user interfaces that can change presentation, navigation, and content according to the user abilities and preferences. Over time, developers and designers have started to become aware of such important issues, and we can find several applications that provide some level of adaptation. Unfortunately, often they are not sufficient to meet users needs, and more flexible and usable solutions are necessary.

The technological fast evolution makes this issue more challenging because people are more and more used to access their applications through a variety of devices ranging from wearables to large screens, also exploiting different interaction modalities, and there are various emerging JavaScript frameworks that are changing the way how people develop their applications. Flexible solutions should allow developers and designers to control the adaptation of their user interfaces at various granularity levels (single elements, groups, pages, etc.) and for various types of attributes. The adaptation should consider the various contextual aspects in order to be more effective, also considering emotional and environmental parameters. This area can benefit from the use of intelligent techniques that, based on the analysis of previous interactions, can predict the most suitable adaptations. However, more importantly, the users should be in control of the adaptation; they should know when the adaptation is triggered, where, how, and why it is applied. In this way, new tools may be developed to empower even nonprofessional developers to directly personalize their applications according to their actual and dynamic needs.

29.8 Conclusions

In this chapter, we have discussed several issues related to UI adaptation for accessibility. Adapting the UI to the special needs of different kinds of users is a must and the problem has been discussed in the literature for more than 30 years. While each particular user or user profile might pose a different challenge, there are already techniques that allow fine-grained personalization of the interface to improve its accessibility and usability.

We have presented a discussion of the proposed solutions in which we separate those approaches in which the interface adaptation is somewhat “built-in” in the system design and those in which adaptation occurs “outside” of the original application.

In both types of approaches, there are a wide range of different techniques that so far have shown to be powerful enough to face existing challenges to achieve adaptation. While in “internal” approaches much of the burden for foreseeing adaptation is often dealt with by designers, even if some end-user development approach is emerging, in “external” ones, there is a tendency to involve end-users either by building their own adaptations or solving others’ problems via crowdsourcing.

However, there is yet much work to do as mentioned in Sects. 29.5 and 29.6. New implementation frameworks (such as Angular, Node.js) pose new technical issues in applying the adaptation solutions. More generally, further longitudinal studies are needed with final users representing the various target communities in order to validate the various technical solutions and their actual effectiveness.

Acknowledgements The authors acknowledge the support from the Argentinian National Agency for Scientific and Technical Promotion (ANPCyT), grant numbers PICT-2015-3000 and PICT-2015-2050.

References

Abou-Zahra S, Brewer J, Cooper M (2018) Artificial intelligence (AI) for web accessibility. In: Proceedings of the internet of accessible things on - W4A 2018. ACM Press, New York, pp 1–4. https://doi.org/10.1145/3192714.3192834. ISBN 9781450356510

Ahmetovic D, Gleason C, Ruan C, Kitani K, Takagi H, Asakawa C (2016) NavCog. In: Proceedings of the 18th international conference on human-computer interaction with mobile devices and services - MobileHCI 2016. ACM Press, New York, pp 90–99. https://doi.org/10.1145/2935334.2935361. ISBN 9781450344081

Akiki PA, Bandara AK, Yu Y (2014) Adaptive model-driven user interface development systems. ACM Comput Surv 47:1–33. https://doi.org/10.1145/2597999. ISSN 03600300

Asakawa C, Takagi H (2000) Annotation-based transcoding for nonvisual web access. In: Proceedings of the fourth international ACM conference on Assistive technologies - Assets 2000, pp 172–179. https://doi.org/10.1145/354324.354588

Asakawa C, Takagi H (2008) Transcoding. In: Web Accessibility. Springer, London, pp 231–260. https://doi.org/10.1007/978-1-84800-050-6_14

Bigham JP, Kaminsky RS, Ladner RE, Danielsson OM, Hempton GL (2006) WebInSight: In: Proceedings of the 8th international ACM SIGACCESS conference on computers and accessibility - Assets 2006. ACM Press, New York, p 181. https://doi.org/10.1145/1168987.1169018. ISBN 1595932909

Bigham JP (2007) AccessMonkey: enabling and sharing end user accessibility improvements. ACM SIGACCESS Access Comput 89:3–6. https://doi.org/10.1145/1328567.1328568

BootstrapAccessibilityPlugin, https://www.paypal-engineering.com/2014/01/28/bootstrap-accessibility-plugin-making-the-popular-web-development-framework-better/

Bouvin NO (1999, February) Unifying strategies for Web augmentation. In: Proceedings of the tenth ACM conference on hypertext and hypermedia: returning to our diverse roots. ACM, pp 91–100. https://doi.org/10.1145/294469.294493

Bouzit S, Calvary G, Coutaz J, Chene D, Petit E, Vanderdonckt J (2017) The PDA-LPA design space for user interface adaptation. In: 2017 11th international conference on research challenges in information science (RCIS). IEEE, pp 353–364. https://doi.org/10.1109/RCIS.2017.7956559. ISBN 978-1-5090-5476-3

Brown A, Harper S (2013) Dynamic injection of WAI-ARIA into web content. In: Proceedings of the 10th international cross-disciplinary conference on Web Accessibility - W4A 2013. ACM Press, New York, p 1. https://doi.org/10.1145/2461121.2461141. ISBN 9781450318440

Brusilovsky P (2001) User modeling and user-adapted interaction 11:87. https://doi.org/10.1023/A:1011143116306

Calvary G, Coutaz J, Bouillon L, Florins M, Limbourg Q, Marucci L, Paternò F, Santoro C, Souchon N, Thevenin D, Vanderdonckt J (2002) Cameleon reference framework in cameleon reference framework

Chung J-W, Min H-J, Kim J, Park JC (2013)Enhancing readability of web documents by text augmentation for deaf people. In: Proceedings of the 3rd international conference on web intelligence, mining and semantics - WIMS 2013. ACM Press, New York, p 1. https://doi.org/10.1145/2479787.2479808. ISBN 9781450318501

Crespo RG, Espada JP, Burgos D (2016) Social4all: definition of specific adaptations in web applications to improve accessibility. Comput Stand Interfaces 48:1–9. https://doi.org/10.1016/J.CSI.2016.04.001. ISSN 0920-5489

Dey AK, Newberger A (2009)Support for context-aware intelligibility and control. In: Proceedings of the 27th international conference on Human factors in computing systems - CHI 09. ACM Press, New York, p 859. https://doi.org/10.1145/1518701.1518832. ISBN 9781605582467

Díaz O, Arellano C (2015) The augmented web. ACM Trans Web 9:1–30. https://doi.org/10.1145/2735633. ISSN 15591131

Dieterich H, Malinowski U, Kuhme T, Schneider-Hufschmidt M (1993) State of the art in adaptive user interfaces. Adapt User Interfaces: Princ Pract 10:13

Edmonds E (1982) The mancomputer interface: a note on concepts and design. Int J Man-Mach Stud 16:231–236. https://doi.org/10.1016/S0020-7373(82)80060-6 ISSN 00207373

Fowler M, Beck K (1999) Refactoring: improving the design of existing code. Addison-Wesley, Boston. ISBN 0201485672

Gajos KZ, Long JJ, Weld DS (2006) Automatically generating custom user interfaces for users with physical disabilities. In: Proceedings of the 8th international ACM SIGACCESS conference on computers and accessibility - Assets 2006. ACM Press, New York, p 243. https://doi.org/10.1145/1168987.1169036. ISBN 1595932909

Galindo JA, Dupuy-Chessa S, Céret E (2017, August) Toward a generic architecture for UI adaptation to emotions. In: Proceedings of the 29th conference on l'Interaction Homme-Machine. ACM, pp 263–272. https://doi.org/10.1145/3132129.3132156

Garrido A, Firmenich S, Grigera J, Rossi G (2017) Data-driven usability refactoring: tools and challenges. In: 2017 6th International workshop on software mining (SoftwareMining). IEEE, pp 52–55. https://doi.org/10.1109/SOFTWAREMINING.2017.8100854. ISBN 978-1-5386-1389-4

Garrido A, Firmenich S, Rossi G, Grigera J, Medina-Medina N, Harari I (2013) Personalized web accessibility using client-side refactoring. IEEE Internet Comput 17:58–66. https://doi.org/10.1109/MIC.2012.143. ISSN 1089-7801

Ghiani G, Leporini B, Paternò F (2009) Vibrotactile feedback to aid blind users of mobile guides. J Vis Lang Comput 20:305–317. https://doi.org/10.1016/j.jvlc.2009.07.004. ISSN 1045926X

Ghiani G, Manca M, Paternò F, Porta C (2014) Beyond responsive design: context-dependent multimodal augmentation of web applications. Springer, Cham, pp 71–85. https://doi.org/10.1007/978-3-319-10359-4_6

Ghiani G, Manca M, Paternò F, Santoro C (2017) Personalization of context-dependent applications through trigger-action rules. ACM Trans Comput-Hum Interact 24:1–33. https://doi.org/10.1145/3057861. ISSN 10730516

Grigera J, Garrido A, Rivero JM, Rossi G (2017) Automatic detection of usability smells in web applications. Int J Hum-Comput Stud 97:129–148. https://doi.org/10.1016/j.ijhcs.2016.09.009

Hanson VL, Crayne S (2005) Personalization of web browsing: adaptations to meet the needs of older adults. Univers Access Inf Soc 4:46–58. https://doi.org/10.1007/s10209-005-0110-9. ISSN 1615-5289

Hanson V, Richards J (2005) Achieving a more usable World Wide Web. Behav Inf Technol 24:231–246. https://doi.org/10.1080/01449290412331327465. ISSN 0144-929X

Hussain J, Ul Hassan A, Muhammad Bilal HS, Ali R, Afzal M, Hussain S, Bang J, Banos O, Lee S (2018) Model-based adaptive user interface based on context and user experience evaluation. J Multimodal User Interface 12:1–16. https://doi.org/10.1007/s12193-018-0258-2. ISSN 1783-7677

Kumar C, Menges R, Müller D, Staab S (2017) Chromium based framework to include gaze interaction in web browser. In: Proceedings of the 26th international conference on World Wide Web

companion - WWW 2017 companion. ACM Press, New York, pp 219–223. https://doi.org/10.1145/3041021.3054730. ISBN 9781450349147
Lin Y, Breugelmans J, Iversen M, Schmidt D (2017) An adaptive interface design (AID) for enhanced computer accessibility and rehabilitation. Int J Hum-Comput Stud 98:14–23. https://doi.org/10.1016/J.IJHCS.2016.09.012. ISSN 1071-5819
Lunn D, Bechhofer S, Harper S (2008) The SADIe transcoding platform. In: Proceedings of the 2008 international cross-disciplinary workshop on Web accessibility (W4A) - W4A 2008. ACM Press, New York, p 128. https://doi.org/10.1145/1368044.1368073. ISBN 9781605581538
Lunn D, Harper S, Bechhofer S (2009) Combining SADIe and AxsJAX to improve the accessibility of web content. In: Proceedings of the 2009 international cross-disciplinary conference on web accessibililty (W4A) - W4A 2009. ACM Press, New York, p 75. https://doi.org/10.1145/1535654.1535672. ISBN 9781605585611
Mangiatordi A, Sareen HS (2011) Farfalla project: browser-based accessibility solutions. In: Proceedings of the international cross-disciplinary conference on web accessibility - W4A 2011. ACM Press, New York, p 1. https://doi.org/10.1145/1969289.1969317. ISBN 9781450304764
Marcotte E, Impr. EMD (2011) Responsive web design, Eyrolles, ISBN 2212133316
McKinley PK, Sadjadi SM, Kasten EP, Cheng BH (2004) A taxonomy of compositional adaptation. Rapport Technique numéro MSU-CSE-04-17
Miñón R, Paternò F, Arrue M, Abascal J (2016) Integrating adaptation rules for people with special needs in model-based UI development process. Univers Access Inf Soc 15:153–168. https://doi.org/10.1007/s10209-015-0406-3. ISSN 1615-5289
Nebeling M, Speicher M, Norrie M (2013) W3touch. In: Proceedings of the SIGCHI conference on human factors in computing systems - CHI 2013. ACM Press, New York, p. 2311. https://doi.org/10.1145/2470654.2481319. ISBN 9781450318990
Paternò F (2005) Model-based tools for pervasive usability. Interact Comput. https://doi.org/10.1016/j.intcom.2004.06.017. ISSN 09535438
Paternò F (2013) User interface design adaptation in the encyclopedia of human-computer interaction. In: The encyclopedia of human-computer interaction, 2nd edn
Paternò F, Santoro C, Spano LD (2011) Engineering the authoring of usable service front ends. J Syst Softw. https://doi.org/10.1016/j.jss.2011.05.025. ISSN 01641212
Peinado I, Ortega-Moral M (2014) Making web pages and applications accessible automatically using browser extensions and apps. Springer, Cham, pp 58–69. https://doi.org/10.1007/978-3-319-07509-9_6
Peissner M, Häbe D, Janssen D, Sellner T (2012) MyUI. In: Proceedings of the 4th ACM SIGCHI symposium on engineering interactive computing systems - EICS 2012. ACM Press, New York, p 81. https://doi.org/10.1145/2305484.2305500. ISBN 9781450311687
Pilgrim M, Mark (2005) Greasemonkey hacks. O'Reilly, Sebastopol. ISBN 0596101651
Prasad GVRJS, Soumya MS, Choppella V (2017) Renarrating web pages for improving information accessibility. In: 2017 12th international conference on intelligent systems and knowledge engineering (ISKE), IEEE, pp 1–8. https://doi.org/10.1109/ISKE.2017.8258772. ISBN 978-1-5386-1829-5
Puzis Y, Borodin Y, Puzis R, Ramakrishnan I (2013) Predictive web automation assistant for people with vision impairments. In: Proceedings of the 22nd international conference on World Wide Web - WWW 2013. ACM Press, New York, pp 1031–1040. https://doi.org/10.1145/2488388.2488478. ISBN 9781450320351
Salber D, Dey AK, Abowd GD (1999) The context toolkit. In: Proceedings of the SIGCHI conference on human factors in computing systems the CHI is the limit - CHI 1999. ACM Press, New York, pp 434–441. https://doi.org/10.1145/302979.303126. ISBN 0201485591
Sottet J-S, Ganneau V, Calvary G, Coutaz J, Demeure A, Favre J-M, Demumieux R (2007) Model-driven adaptation for plastic user interfaces. Springer, Heidelberg, pp 397–410. https://doi.org/10.1007/978-3-540-74796-3_38

Stephanidis C, Paramythis A, Sfyrakis M, Stergiou A, Maou N, Leventis A, Paparoulis G, Karagiannidis C (1998) Adaptable and adaptive user interfaces for disabled users in the AVANTI project. Springer, Heidelberg, pp 153–166. https://doi.org/10.1007/BFb0056962

Stephanidis C, Savidis A (2001) Universal access in the information society: methods, tools, and interaction technologies. Univers Access Inf Soc 1(1):40–55. https://doi.org/10.1007/s102090100008. ISSN 1615-5289

Takagi H, Kawanaka S, Kobayashi M, Itoh T, Asakawa C (2008) Social accessibility. In: Proceedings of the 10th international ACM SIGACCESS conference on computers and accessibility - Assets 2008. ACM Press, New York, p 193. https://doi.org/10.1145/1414471.1414507. ISBN 9781595939760

Takagi H, Kawanaka S, Kobayashi M, Sato D, Asakawa C (2009) Collaborative web accessibility improvement. In: Proceeding of the eleventh international ACM SIGACCESS conference on computers and accessibility - Assets 2009. ACM Press, New York, p 195. https://doi.org/10.1145/1639642.1639677. ISBN 9781605585581

Vanderheiden GC, Treviranus J, Ortega-Moral M, Peissner M, de Lera E (2014) Creating a global public inclusive infrastructure (GPII). Springer, Cham, pp 506–515. https://doi.org/10.1007/978-3-319-07509-9_48

WAI-ARIA (W3C), Web accessibility initiative (WAI) – W3C. https://www.w3.org/WAI/standards-guidelines/aria/

WebAdapt2Me. https://www-03.ibm.com/press/us/en/pressrelease/19515.wss

Whittle J, Hutchinson J, Rouncefield M (2014) The state of practice in model-driven engineering. IEEE Softw 31:79–85. https://doi.org/10.1109/MS.2013.65. ISSN 0740-7459

Yang SJ, Shao NW (2007) Enhancing pervasive web accessibility with rule-based adaptation strategy. Expert Syst Appl 32(4):1154–1167. https://doi.org/10.1016/j.eswa.2006.02.008

Yesilada Y, Harper S, Goble C, Stevens R (2004) DANTE. In: Proceedings of the 13th international World Wide Web conference on alternate track papers and posters - WWW Alt. 2004. ACM Press, New York, p 490. https://doi.org/10.1145/1013367.1013540. ISBN 1581139128

Yoda I (2018) A study of the adaptive gesture interface for the severely physically handicapped. Impact 2018:41–43