Towards Deep Adaptivity – A Framework for the Development of Fully Context-Sensitive User Interfaces

Gottfried Zimmermann[1], Gregg C. Vanderheiden[2], and Christophe Strobbe[1]

[1] Responsive User Interface Experience Research Group, Stuttgart Media University, Stuttgart, Germany
gzimmermann@acm.org, strobbe@hdm-stuttgart.de
[2] Trace R&D Center, University of Wisconsin-Madison, USA
gv@trace.wisc.edu

Abstract. Adaptive systems can change various adaptation aspects at runtime, based on an actual context of use (the user, the platform, and the environment). For adaptable systems, the user controls the adaptation aspects. Both adaptivity and adaptability are pre-requisites for context-sensitive user interfaces that accommodate the needs and preferences of persons with disabilities. In this paper, we provide an overview of the various adaptation aspects and describe a general framework consisting of six steps for the process of user interface adaptation. Based on the framework, we describe our vision of combining the GPII and URC technologies to achieve fully context-sensitive user interfaces.

Keywords: Adaptive user interface, adaptable user interface, user interface adaptation, user interface adaptation aspect, context-sensitive user interface, abstract user interface, Universal Remote Console (URC), Global Public Inclusive Infrastructure (GPII), Cloud4all.

1 Introduction

Systems that adapt their user interfaces in one or more aspects (e.g. font size, navigation structure, closed captions) have been around for a while. However, existing systems tend to focus on selected aspects of adaptations, addressing only the needs of a particular user group. For fully context-sensitive user interfaces it is important to cover the full range of adaptations that is required to address the wide range of needs and preferences of all users, including those that are ageing and those with disabilities.

The remainder of this paper is structured as follows: Section 2 introduces basic concepts of user interface adaptation and a 3-layer model of user interface adaptation aspects. Section 3 looks at selected previous work with varying coverage of user interface adaptation (based on the 3 layers of user interface adaptation aspects). The main contribution of this paper is presented in section 4, where we describe a six-step framework for the development of fully context-sensitive user interfaces. Section 5 describes how a combination of the Global Public Inclusive Infrastructure (GPII) and the Universal Remote Console (URC) framework can produce fully context-sensitive user interfaces with all six steps involved. Finally, section 6 provides a conclusion.

C. Stephanidis and M. Antona (Eds.): UAHCI/HCII 2014, Part I, LNCS 8513, pp. 299–310, 2014.

2 User Interface Adaptation Terms and Concepts

In this section, we define basic terms and concepts that are necessary for a common understanding on the topic of this paper.

2.1 User Interface Adaptation

We define user interface adaptation as a process spanning development time (i.e. when the application is developed and its user interface is designed) and runtime (i.e. when the user interface is shown to the user and the user interacts through it with the application). The activities in this process serve the purpose of preparing and instantiating a user interface in which at least some aspects are tailored (adapted) to the specific context of use at runtime. We will come back to the question what these user interface aspects are in section 2.5.

2.2 Context of Use

The context of use reflects the specific conditions under which the user interacts with the application through the user interface. It consists of the following components:

1. A *user model* describing the needs and preferences of the user at the time of user interaction.[1] A user model may include static information such as the need for a special contrast theme, or the need for sticky keys to be enabled. However, even static information may change over the lifetime of a user, e.g. when the user's abilities change due to ageing. The user model may also include dynamic information that is changed quite often, even during the course of user interaction. Such dynamic information may include the user's current level of attention, or their current mood.
2. A *platform model* comprising information about the user's interaction device (hardware) and the software platform the user interface runs on. This may include the size and resolution of the visual display, the type of a keyboard (e.g. physical, onscreen) or the availability of a runtime engine (e.g. JavaScript) and version.
3. An *environment model* holding information on the specific situation in which the user interaction occurs. This may include the location of the user interaction, levels of ambient light and ambient noise, or the current screen orientation (portrait vs. landscape) that depends on how the user is holding their mobile device.

Due to the heterogeneous nature of the context of use, its information cannot be prepared at full before runtime. Rather it needs to be assembled from various sources of information, sometimes upon demand only. However, some parts of the context of use are rather persistent over time, such as the static part of the user model, and the platform model (only if the user does not change their device). Also, some locations may

[1] In GPII, we explicitly focus on the user's needs and preferences only (see section 5). We do not try to model the users themselves.

always have the same conditions of light and noise. These parts can be set up once, stored safely at a dedicated place, and retrieved any number of times to be used as readily available components of a context of use.

The format in which the context of use is represented varies over different systems. Common formats include nested structures, key-value pairs, and formal ontologies. However, many applications use proprietary formats.

2.3 Adaptable vs. Adaptive System

We call a system *adaptable* if it lets the user adjust its user interface aspects by dedicated user actions, e.g. in a configuration dialog, or by activating a button or menu item. A system is adaptive if it automatically adapts some aspects to the current context of use; which may involve asking the user for confirmation of the suggested change. In brief, a system is adaptable if the adaptation is user-initiated, and it is adaptive if the adaptation is system-initiated.

Oftentimes, systems are both adaptable and adaptive. This will typically result in mixed dialogs in which a user changes some aspect of adaptation, and the system will react upon this by proposing to change another (complementary) aspect to the user.

2.4 User Interface Integration vs. User Interface Parameterization

In general, there are two ways of building adaptability and adaptivity into systems:

1. *User interface parameterization:* The user interface is "tweaked" along predetermined parameters (e.g. font size) at runtime. The capability for tweaking may be part of the user interface code, or be provided by the runtime platform (e.g. browser buttons for changing the font size). In any case, the designer needs to make sure that their user interface design is suitable for presentation with a variety of user interface parameter settings.
2. *User interface integration:* A user interface is assembled from a set of user interface resources (e.g. dialogs, icons, fonts, labels, videos, captions) at runtime, based on the specific context of use or the user's configuration. Usually, user interface resources have to be prepared by humans at development time, although in some cases they can be generated by sophisticated automatic systems (e.g. YouTube can automatically create closed captions for English video clips, but the quality is usually not as good as when done by humans). Unfortunately, at development time the specific context of use is not known. Therefore, usually a set of user interface resource variants are created for a variety of (projected) contexts of use. If, however, user interface resources can be swiftly generated on demand, the creation of the resources and the user interface integration can both happen at runtime, thus resulting in a user interface that can explicitly address the peculiarities of the context of use at hand.

Most adaptable and adaptive systems today employ user interface parameterization as a means for adaptation. If user interface integration is involved, it is usually done

manually at development time, fully automated (with automatically generated resources) at runtime, or provided by the runtime platform (e.g., the browser on an iPhone replaces a pull down menu on web pages by a drum). Although quite flexible and convenient for developers, automatically generated user interfaces sometimes have the drawback of being designed with functional rather than aesthetic goals in mind.

2.5 User Interface Adaptation Aspects

We divide a user interface and its aspects into three layers, stacked on top of each other. This notion is inspired by the layered human-computer interaction model by Herczeg [1]. We describe our model of user interface adaptation aspects with its layers from top to bottom (see **Error! Reference source not found.**).

- *Presentation & input events layer:* Presentational aspects (i.e. related to information output to the user) can easily be changed at runtime, usually by user interface parameterization. Examples include visual presentation aspects such as text size, zoom factor, contrast theme (for text), letter spacing, line spacing, and button size; auditory presentation aspects such as speech volume, pitch and speech rate; and tactile presentation aspects such as Braille mode (6 or 8 dots) and Braille grade (contraction level). User interface aspects for receiving information from the user address input events, including the mapping of user actions to the application's operations. Examples include keyboard shortcuts, keyboard settings (sticky keys, repeat keys, toggle keys), mouse speed and acceleration, and gesture alphabet.

- *Structure & grammar layer:* This layer includes aspects of the information architecture, input and output modalities, and widget substitution. For example, the navigation structure may be modified based on the available screen space (e.g. desktop version vs. mobile version of a Website). The standard grouping of a user interface may change into a "wizard mode" in which the novice user is walked step by step through a set of dialogs. Widgets may get replaced by variants which better suit the particular user or platform (e.g. a set of big push-buttons may be better suited for a user with hand tremor than a drop-down menu). A screen reader may augment the output modality from visual presentation to speech output. The input modality may switch from using the keyboard for text input to speech input.

- *Content & semantics layer:* The "deep" aspects of user interface adaptation involve changes in content and semantics. Information may be presented in alternative languages, including sign languages. Text may be rephrased and shortened to make it easier to read, and images may be changed to simplified versions. Additional content may be provided to assist some users in using the user interface and the functionality of the application. Alternative images may be needed for accommodating a user's or device's limitations on the use of color and color coding. For multimedia, captions, audio descriptions and sign language interpretation may get displayed with the video in a synchronized fashion.

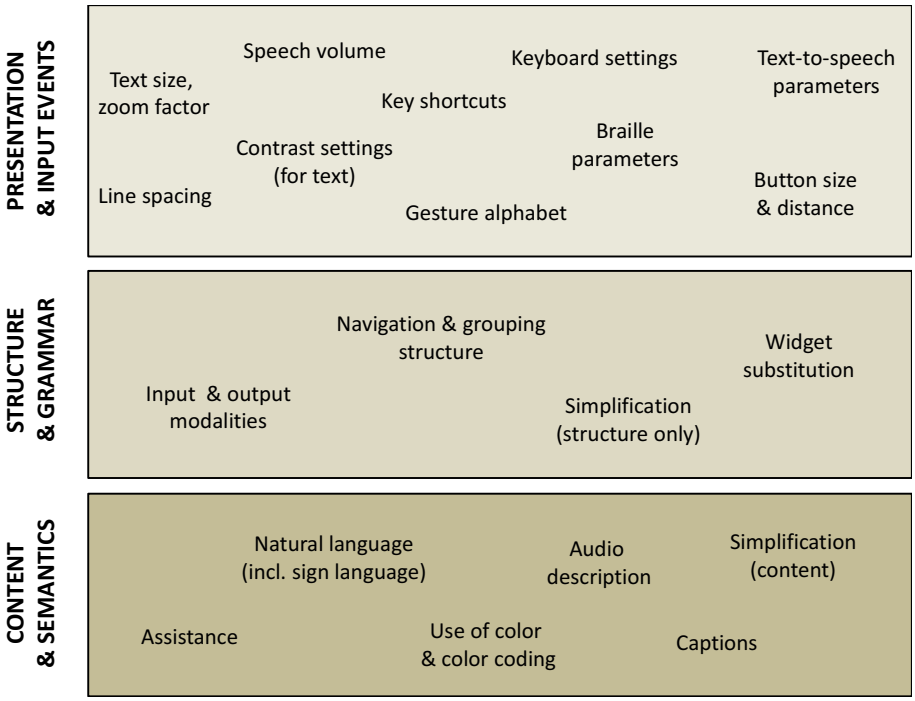

Fig. 1. A 3-layer model of user interface adaptation aspects, populated with sample aspects

Aspects in the upper layer (presentation & input events) are typically handled by user interface parameterization, or alternate presentation modes built into a browser or player/viewer, that take effect at runtime. Aspects in the middle layer (structure & grammar) and lower layer (content & semantics) usually involve user interface resources that have to be prepared at development time, with the resulting user interface integration happening at runtime.

3 Previous Work

Based on the three-layer model of user interface adaptation aspects, in this section we briefly describe some selected systems and tools that address one or more layers of aspects. For the purpose of this paper, we focus mainly on technologies in the area of Global Public Inclusive Infrastructure (GPII) and Universal Remote Console (URC). For a more elaborated review of related work in the field of adaptive user interfaces, refer to [9].

3.1 Fluid User Interface Options

The Fluid user interface options panel [2, 5] is a JavaScript library that can be integrated into any Web page or Web application to allow the user to adapt the Web content in the following aspects:

- text size, text style, line spacing, color & contrast
- show table of contents
- emphasize links, make inputs larger

Web authors who want to make use of the user interface options just need to include a set of external files, and add a small amount of markup to their Web page.

The user interface options panel is part of the open-source development work in GPII. It addresses aspects of the upper layer of the aspects model only (see **Error! Reference source not found.**). It employs user interface parameterization as a mechanism to make a Web application adaptable.

3.2 ATbar

A similar tool has been developed by the University of Southampton, called ATbar [6, 8]. It is a cross-browser plugin that, once installed on the user's browser, adds a toolbar to any Web page. The toolbar allows the user to adjust the following adaptation aspects:

- text size, font selection, line spacing
- selection of page style (foreground and background colors)
- color tint overlay over the entire webpage
- text-to-speech output for highlighted text
- increased readability by removing Web page clutter

In addition to these aspects, the ATbar provides the following functions:

- report an encountered accessibility issue on a Web page to volunteers who will bring them to the website owners ("fix the web" function)
- highlight a word and call its definition from Wiktionary
- word prediction and spell checker for user input into forms

ATbar addresses some aspects of the top and middle layer of the aspects model (see **Error! Reference source not found.**), but no content-related aspects (lower layer). It employs user interface parameterization[2] as a mechanism to make a Web application adaptable.

3.3 URC Pick-An-Interface

The URC framework [3] is a technology for pluggable user interfaces for the control of electronic devices and services (*Targets*). In its most commonly installed architecture, a gateway called Universal Control Hub (UCH) [10] connects any number of controllers with any number of Targets. Supplemental user interfaces and pertaining resources (for projected contexts of use) may be created by any party and deployed to a *Resource Server* that acts as a market place for pluggable user interfaces. If multiple

[2] Except for the text-to-speech function which represents a user interface integration mechanism using an automatically generated resource (the speech output).

user interfaces are available for a Target and the controller platform, the user can select one from the pick-an-interface screen [4].

In the URC framework, user interface resources typically need to be prepared at development time, marked with metadata describing their suitability for specific contexts of use, and are then deployed to the Resource Server. At runtime, a suitable user interface can be assembled by the UCH by pulling together the resources that are most suitable for the current context of use.

In general, URC-based user interfaces can cater for all layers of adaptation aspects. However, due to the specific architecture of URC and its focus on user interface integration, it is most commonly used for adaptations on deeper adaptation aspects, i.e. the lower and middle layer of the three-layer model (see **Error! Reference source not found.**). For example, the set of available user interfaces for a home media system may include a simplified version that displays only the most common functions. The URC framework defines a special resource type for describing a grouping structure which makes it easy to prepare alternative navigation structures for different contexts of use.

Current URC-based implementations are adaptive based on the platform model. However, the user needs to pick their preferred user interface, i.e. they are also adaptable for the user's preferences.

3.4 Summary of Previous Work

Each system described above caters for a part only of the whole set of adaptation aspects. Today's GPII implementations are adaptable and focus on the shallow adaptation aspects. The ATbar is similar in its concept, with some deeper aspects included, and some extra functions. The URC framework addresses the medium deep and deep adaptation aspects, and provides both adaptability and adaptiveness (although in a limited manner for each).

It can be seen that, if we combine the GPII and the URC approach, it is possible to cover the full range of adaptation aspects, across all three layers (see **Error! Reference source not found.**). The combination results in a framework for fully context-sensitive user interfaces that are both adaptable and adaptive based on the full range of the context of use (i.e. the user model, platform model and environment model[3]). Before we further elaborate on the combination of GPII and URC, we introduce a generic framework for user interface adaptation that will be the basis for describing our vision on such a combined system.

4 A Six-Step Framework for User Interface Adaptation

In general, the process of user interface adaptation (which results in fully context-sensitive user interfaces) is composed of six steps (see Fig. 2). Note that the first three steps occur at development time, and the last three steps at runtime.

---

[3] Current GPII and URC implementations don't address adaptations based on the environment model, but this is not due to conceptual constraints. The Cloud4all project is currently working on the accommodation of the environment model.

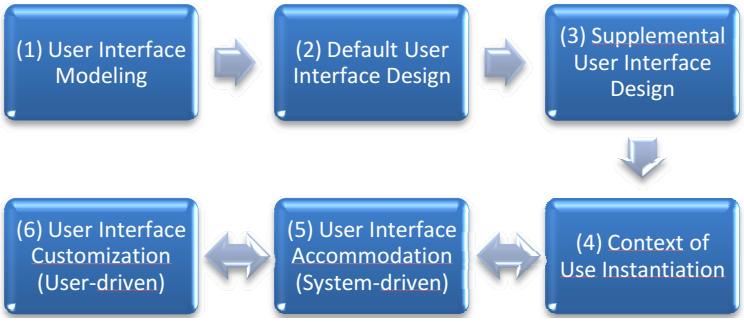

Fig. 2. The six steps of user interface adaptation

We describe the six steps in the following sub-sections. Note that we provide a "canonical" description of the framework; concrete implementations may vary. For example, steps may be executed by automatic systems or humans, and they may be combined or performed in a different order. However, if a step is completely omitted, this will result in a decrease of user interface adaptation capabilities.

4.1 User Interface Modeling

The first step in the user interface adaptation process is for the application developer to describe the user interface in an abstract fashion. This step happens at design time. There are many forms of abstract user interface descriptions, including task models, user interface models, Web service interface descriptions (WSDL), descriptions of an Application Programming Interface (API), or ontological descriptions of a user interface's functionality.

Sometimes, a supplementary user interface model (or parts thereof) is created as an annotation to the default user interface (see section 4.2). In this case, the default user interface is designed first, and then its model is "bolted on" by the designer or another person. For example, the WAI-ARIA [7] markup extends HTML (and other markup languages) by semantic annotations that support user interface adaptation, and would otherwise not be available in the host language.

It is a common mistake to completely omit user interface modeling in the design process. This happens quite often when design is directly aimed (on purpose or unintentionally) at the "typical user" who does not have a conceived need for user interface adaptations. Usually, this results in visual user interfaces with no or only a low level of adaptation.

4.2 Default User Interface Design

The second step is about the design of the default version of a concrete user interface which is usually done at development time by a user interface expert or designer. Typically, aesthetics and corporate identity play an important role in this step.

However, some systems automatically generate the (default) user interface on demand at runtime, tailored to a concrete context of use. In this case, there is no supplemental user interface design, and the third step is omitted (see section 4.3).

The default user interface is based on the abstract description of an application's user interface (see section 4.1). The design of the default user interface includes pertaining resources such as icons, textures, labels, audio clips and videos.

4.3 Supplemental User Interface Design

In the third step, alternative user interfaces or parts thereof are designed. This usually happens at any time between development time and runtime, and is done by the designer of the default user interface, or other designers, possibly by third parties. As the default user interface, supplemental user interfaces are based on the abstract description of an application's user interface (see section 4.1).

Supplemental user interfaces or supplemental user interface resources are created to extend or replace the default user interface and/or its resources (see section 4.2). They are tailored for a particular context of use, and marked as such. They are typically deployed to a resource repository, acting as a "market place" for supplemental user interfaces and pertaining resources that can be downloaded on demand at runtime. However, some systems store supplemental user interfaces locally, and therefore are usually not open for third party contributions. This restricts user interface adaptations to "officially sanctioned" user interfaces, and can lead to inaccessible user interfaces for some non-accommodated user groups.

4.4 Context of Use Instantiation

The fourth step is for the runtime platform to identify and instantiate the context of use (user model, device model and environment model, see section 2.2). This is useful for adaptable systems, and required for adaptive systems. For adaptable systems, the system may have the user's previous settings stored in order to restore them at the next time when the same or a similar context of use is encountered. For adaptive systems, the system-driven accommodation is for the specific user, device and environment which are encountered.

Things can get more complicated for more sophisticated systems that can adapt user interfaces in the course of a user sessions, in response to dynamically changing values of the context of use. For example, the user may change the orientation of a hand-held device (from portrait to landscape mode or vice versa), or the lighting conditions may vary over time.

If this step is omitted (i.e. there is no notion of context of use), the system can only be adaptable by user-driven customization (see section 4.6), and not be adaptive by system-driven accommodation (see section 4.5).

4.5 User Interface Accommodation (System-Driven)

The fifth step is the accommodation of the user interface by the system, adapting the user interface to best match the particular context of use at runtime. This step is required for adaptive systems. It is a good habit, though, to give the user final control over what user interface aspects are adapted in which way, by letting the user confirm or reject system-derived suggestions.

This step can involve both user interface integration and user interface parameterization.

Oftentimes, user interface accommodation is a continuous process, involving multiple iterations of refinements in the course of a user session, e.g. when reacting to dynamic changes in the context of use, or to user-initiated adaptations (see section 4.6).

4.6 User Interface Customization (User-Driven)

The sixth step occurs when the user changes a user interface aspect at runtime, by customizing the user interface by user operations that are either provided by the user interface or by the runtime platform. This step is required for adaptable systems.

As for user interface accommodation, user interface customization can involve both user interface integration and user interface parameterization.

User-initiated customizations may be stored (e.g. in the user model) to make them permanent for the user and the particular context of use. However, this is not sufficient to be called an adaptive system.

Care must be taken for adaptable systems in the design phase (steps 2 & 3) to accommodate for changes on the full range of user interface aspects that the user can control at runtime. For example, if a Web designer ignores the implications of the browser's zooming function, content may get distorted at runtime, or require constant horizontal scrolling, when the user selects an unusual zoom level.

User interface accommodation (see section 4.5) and user interface customization can be used together in an interwoven fashion. For example, the system may react to a user increasing the font size, by suggesting to also modifying the contrast theme to make it easier to read. This can go back and forth, in a mixed dialog negotiation to find the best possible adaptation for a particular context of use.

5 GPII + URC: A Vision for User Interface Adaptation

Taken together, GPII and URC provide the concepts and ingredients for a technology that can fill all gaps in the six-step adaptation framework described above (see section 4). With its abstract user interface and user interface integration concepts, URC provides the means for the preparation and deployment of (manually or automatically) designed user interface resources at development time. At runtime, these resources are selected and/or assembled to provide a "good match" for the particular context of use. Now, GPII with its user interface parameterization approach can further fine-tune the user interface to make it into a "best match" for the particular context of use, ready to also react to changes during runtime, if necessary.

With a combination of the GPII and URC technologies, the six steps of user interface adaptations (see section 4) can be annotated as follows:

1. **User interface modelling:** If suitable (in particular for the control of devices), a user interface model is created as a URC "user interface socket description". If a full-fledged user interface socket is not appropriate (e.g. for information-oriented Web applications), the user interface model can consist of some additional markup (dubbed "URC Light") embedded in the default user interface.
2. **Default user interface design:** The Personal Control Panel (a component for letting the user control the adaptation aspects) is embedded into the default user interface. This is the basis for user interface parameterization, and for adaptability.
3. **Supplemental user interface design:** Alternative user interfaces and supplemental resources are designed by any party (including automatic systems), marked with metadata regarding their suitability, and deployed to the openURC Resource Server. This is the basis for user interface integration.
4. **Context of use instantiation:** A standard format for the description of the context of use is currently being developed in the GPII project Cloud4all. In GPII, the user model is called *personal preference set* since it contains only the user's preferences with regard to user interface aspects rather than a functional description of the user.
5. **User interface accommodation (system-driven):** Based on the specific context of use and the available supplemental user interface resources, the runtime engine assembles a user interface (user interface integration). It then fine-tunes the user interface by user interface parameterization. These steps are reapplied if the context of use changes significantly at runtime.
6. **User interface customization (user-driven):** The user can explicitly change selected adaptation aspects by operating the Personal Control Panel. The user interface is then immediately updated to reflect the change. The system remembers the user's preferred settings together with the context of use at hand, and learns from it. Next time, the preferred settings will be activated as part of the user interface accommodation.

6 Conclusion

User interface adaptation needs to be understood as a process spanning development time and runtime. If done properly, this will result in fully context-sensitive user interfaces, thus accommodating users with varying needs and preferences, hardware and software platforms with varying features and constraints, and environmental constraints that can change dynamically.

In this paper we have described our vision on combining the GPII and URC technologies, and we have shown how this can result in fully context-sensitive user interfaces that are capable of the full range of adaptations, including deep adaptations related to content & semantics. The combination of GPII and URC has already been considered in the development of GenURC [9], and is currently being implemented as *URC Light* technology in the Cloud4all Online Banking Demo application[4].

[4] The Online Banking Demo application is described in a separate paper of this conference titled "A Showcase for Accessible Online Banking".

Acknowledgments. This work has been funded by the US Dept of Education, NIDRR, under Grant H133E080022 (RERC on IT Access), and by the European Commission, under FP7 Grant Agreement 289016 (Cloud4All). The opinions herein are those of the authors and not necessarily those of the funding agencies.

References

1. Herczeg, M.: Interaktionsdesign: Gestaltung interaktiver und multimedialer Systeme. Oldenbourg, München (2006)
2. infusion, Tutorial - User Interface Options - Infusion Documentation - Fluid Project Wiki (2014), http://wiki.fluidproject.org/display/docs/Tutorial+-+User+Interface+Options (retrieved February 4, 2014)
3. ISO/IEC, ISO/IEC 24752-1:2008. Information Technology - User Interfaces - Universal Remote Console - Part 1: Framework. International Organization for Standardization, ISO (2008)
4. openURC Alliance, iUCH client for UCH (2013), http://www.openurc.org/tools/iuch-ios-3.0/ (retrieved February 4, 2014)
5. Treviranus, J.: You say tomato, I say tomato, let's not call the whole thing off: the challenge of user experience design in distributed learning environments. On the Horizon 17(3), 208–217 (2009), doi:10.1108/10748120910993231
6. University of Southampton (n.d.). ATbar, https://www.atbar.org/ (retrieved February 4, 2014)
7. W3C, WAI-ARIA Overview (2014), http://www.w3.org/WAI/intro/aria (retrieved February 10, 2014)
8. Wald, M., Draffan, E.A., Newman, R., Skuse, S., Phethean, C.: Access toolkit for education. In: Miesenberger, K., Karshmer, A., Penaz, P., Zagler, W. (eds.) ICCHP 2012, Part I. LNCS, vol. 7382, pp. 51–58. Springer, Heidelberg (2012)
9. Zimmermann, G., Jordan, B., Thakur, P., Yuvarajsinh, G.: Abstract User Interface, Rich Grouping and Dynamic Adaptations - A Blended Approach for the Generation of Adaptive Remote Control User Interfaces. In: Assistive Technology: From Research to Practice, vol. 33, pp. 1289–1297. IOS Press, Vilamoura (2013), doi:10.3233/978-1-61499-304-9-1289.
10. Zimmermann, G., Vanderheiden, G.: The Universal Control Hub: An Open Platform for Remote User Interfaces in the Digital Home. In: Jacko, J.A. (ed.) HCI 2007. LNCS, vol. 4551, pp. 1040–1049. Springer, Heidelberg (2007), doi:10.1007/978-3-540-73107-8.