Towards Context-Driven User Interfaces in Smart Homes

The Cloud4all Project's Smart House Demo

Gottfried Zimmermann, Alexander Henka, Christophe Strobbe, Simone Mack, Annette Landmesser
Responsive Media Experience Research Group
Stuttgart Media University
Stuttgart, Germany
{gzimmermann, henka, strobbe, sm136, al077}@hdm-stuttgart.de

***Abstract*—One of the challenges in AAL and smart homes is the delivery of user interfaces that accommodate the needs and preferences of a diverse group of users for providing a "personal" user interface. Various technical solutions and frameworks for adaptive user interfaces have been proposed in the past. In this paper, we describe a smart house demo application within the Cloud4all research project. This demo combines the Universal Remote Console (URC) approach for pluggable user interfaces with an adapt-at-runtime approach of the Global Public Inclusive Infrastructure (GPII), thus enabling the provision of personal user interfaces that can be fine-tuned to fit a specific runtime context. Some preliminary example user interfaces for smart house appliances are provided.**

Keywords-Cloud4all; URC; AAL; smart house; adaptive user interfaces; GPII; user preferences; user preference set

I. INTRODUCTION

User interfaces for smart house and Ambient Assisted Living (AAL) environments should adapt to the preferences and needs of their users. We then call them "personal user interfaces". In addition to being personal, they should also take into account the user's device and its characteristics (e.g., screen size), and situational parameters of use (e.g., brightness or noise level in the environment). These parameters that drive adaptation are often summarized as "context" or "context of use".

Context-driven user interfaces are important for applications that are used by diverse types of users. Various technical approaches for the delivery of context-driven user interfaces have been proposed, including employment of abstract user interface models at design-time, and dynamic approaches at runtime (see Section II).

Context-driven user interfaces have not been widely adopted by the industry yet. Among the various reasons, the following are of particular interest for us:

1. Designers tend to imagine visual user interfaces that are designed with "pixel fidelity" rather than thinking in abstract structures. For example, a designer would develop a drop-down menu rather than an abstract "pick one from many" interaction element, and a red "Click me" button with rounded corners rather than an abstract "function trigger". Therefore, if we want to get designers to develop flexible user interfaces, we need to let them design a visual user interface as a basis for a context-driven one.
2. Similarly, developers and designers are familiar with tools that let them design the user interface in the traditional (i.e., visual) way. They will not switch to new tools that require them to learn new design paradigms. Therefore, if we want them to design for flexibility and adaptivity, we will have to allow them to use the development tools that they are familiar with, albeit with some possible extensions.
3. Most companies regard the user interfaces of their products as vehicles for conveying their corporate identity to the user. They want the users to identify themselves with the brand by looking at the product's frontend. Naturally, designers don't want "their" user interface to be changed in a way that could jeopardize the user's identification with the brand. At a minimum, they want to be in control of possible adaptations that could occur at runtime. Therefore, if we want industry to adopt possible adaptation approaches, we need to support them in designing "their" user interface, while allowing for tweaking the user interface at runtime along predictable lines for the purpose of personalization. Also, they should be able to design or let third parties (e.g., user interface experts) design alternative user interfaces that would make their products accessible to users of particular user groups and/or particular circumstances. This could help companies to overcome legislative requirements on accessibility without losing control over which user interface variation would be used under which conditions.

The European Cloud4all project [1] addresses the need for adaptive user interfaces in mainstream products despite these challenges. Cloud4all is part of the international Global Public Inclusive Infrastructure (GPII) effort, whose purpose "is to ensure that everyone who faces accessibility barriers due to disability, literacy, digital literacy, or aging, regardless of economic resources, can access and use the Internet and all its information, communities, and services for education, employment, daily living, civic participation, health, and safety" [2].

As part of its development and dissemination work, Cloud4all is providing a simulation of its adaptive user interface technology on the example of a smart house and its appliances. This simulation has been recently set up and will soon become available for the interested public [3].

The remainder of this paper is organized as follows: Section II provides an overview of related work on user interface adaptation at design time and/or runtime. Section III introduces the technical framework that underlies Cloud4all's smart house demonstration. Section IV gives a glimpse into the early development stage of the smart house demonstration, describing a few appliances and their adaptive user interfaces. Finally, in Section V we draw some conclusions and provide an outlook on further research and development within the Cloud4all smart house demo project.

II. Related Work

User Interface Management Systems (UIMS) (e.g., COUSIN [4], HOMER [5]) are early predecessors of today's adaptive user interface systems. They featured a separate user interface management layer so that the user interface could be adapted to the runtime platform (desktop) and other parameters of use. UIMS and similar approaches have not been widely adopted for various reasons. According to [6], one of them is that designers want to control the look and feel of the interactions at a lower level than the UIMS abstraction allowed.

The User Interface Markup Language (UIML) standard by OASIS [7] has been designed to allow for the generation of platform-specific user interfaces, driven by rules for presentation and behavior that are unique for each runtime platform. However, UIML does not provide a vocabulary for user interface components and the development of such a vocabulary, together with an appropriate set of rules, is cumbersome. Also, UIML does not facilitate fully context-driven user interfaces since adaptation happens at design-time rather than runtime.

Of particular interest to us are approaches that focus on the automatic generation of user interfaces with dynamic runtime adaptation mechanisms. Pebbles [8] allows an author to provide an abstract user interface description at design-time which is used to render a control interface on a mobile device at runtime. Similarly, SUPPLE [9] uses a constraint-based algorithm to solve an optimization problem for a concrete layout at runtime. More recently, MyUI [10] has suggested defining abstract user interface models (based on state transitions) and interaction design patterns at design-time, and combining them at runtime based on a specific use context. Although these approaches can facilitate useful adaptations at runtime that may increase the level of accessibility for users with disabilities, they lack the ability for a designer to control the end result of the rendition process, and are therefore deemed unsuitable for mainstream use.

In the area of Web-based user interfaces, the Composite Capabilities / Preference Profile (CC/PP) standard [11] was specified to allow for fine-grained server-based adaptations to a client's runtime platform characteristics and user preferences. However, it has not been widely adopted by industry since it puts the onus on the Web browser manufacturers to create and maintain CC/PP files for each individual Web browser version and runtime platform. The Fluid project [12] with its "user interface options" component offers a way for authors to make their Web pages adaptable to their users along a small set of presentational aspects such as font size, line spacing and button size. This approach works well for users and devices that need some fine-tuning ("tweaking") of the Web page, but does not cater for radical changes in device characteristics (e.g., screen size) or user needs (e.g., simplified user interface).

Finally, the Universal Remote Console (URC) framework [13] facilitates pluggable user interfaces based on an abstract user interface model (called "user interface socket"). Authors have full control over each pluggable user interface, since they can be specified as fine-grained as needed by the author. A distributed URC ecosystem (including a resource server) facilitates contributions of pluggable user interfaces by third parties (e.g., HCI experts and user groups), thus creating a market for user interfaces that is separate from the market of applications. However, there is still a risk for some users (in particular users with severe or multiple disabilities) to be left out due to the high effort of creating specialized pluggable user interfaces for each user group. This has motivated us to work towards a more flexible approach by combining URC's pluggable user interfaces with the runtime "tweaking" capabilities of Fluid, as employed by the smart house demo of the Cloud4all project described in this paper.

III. Technical Framework

Cloud4all’s smart house demo application is based upon the URC framework [13] which is standardized as ISO/IEC 24752 [14]. Based upon the URC framework, the Universal Control Hub (UCH) architecture [15] facilitates a middleware-centered approach that can accommodate any networked target devices and applications (such as smart house appliances). Also, through the URC-HTTP protocol [16], it allows for any Web-based controller (such as smartphones, tablets, TVs and any other device featuring a Web browser) to be used.

On the client side, we employ up-to-date mainstream Web technologies, such as HTML5, CSS and JavaScript for HTML DOM manipulation. This allows us to provide a basically cross-platform user interface code (HTML5 & CSS) that can be tweaked in its structure, presentation and user interaction at runtime (JavaScript code) to respond to a concrete context of use. In principle, we can thus react to the following parts of a use context:

1. *User needs and preferences,* in particular those pertaining to the user interface. ISO/IEC 24751 (*AccessForAll*) defines a framework and vocabulary for expressing such personal user interface preferences but has not been widely adopted due to complexity and flexibility issues. Cloud4all is contributing to a revised framework and registry-based vocabulary for user preference sets, which is intended to be a basis for the revision of AccessForAll. Thus, personal preferences will impact the automatic activation of a suitable pluggable user interface (which can be provided at design time by any party and which is available from the

resource server), and will also drive the "tweaking" of this user interface at runtime in the browser.

2. *Device capabilities*, in particular user interaction aspects such as screen size, and mouse vs. touch based interaction. We are working on including device aspects in a user's preference set to describe specific conditions under which a preference value should become active. In the smart house demo, controller device capabilities will impact the selection of the pluggable user interface and the "tweaking" of this user interface in the browser at runtime. Controller devices to be demonstrated will include modern smartphones, tablets and desktop browsers.
3. *Situational parameters*, such as the situation in which the user interacts with an application (e.g., driving, sitting, walking), and impediments such as ambient noise and high brightness. Conceptual development within Cloud4all is underway to allow for user interface adaptations based on ambient noise and brightness levels. The integration of these concepts into the smart house demo will facilitate the "tweaking" of a personal user interface at and during runtime in the browser. (By listening to changing conditions in the browser, we will even be able to fine-tune the user interface in the course of a running session.)

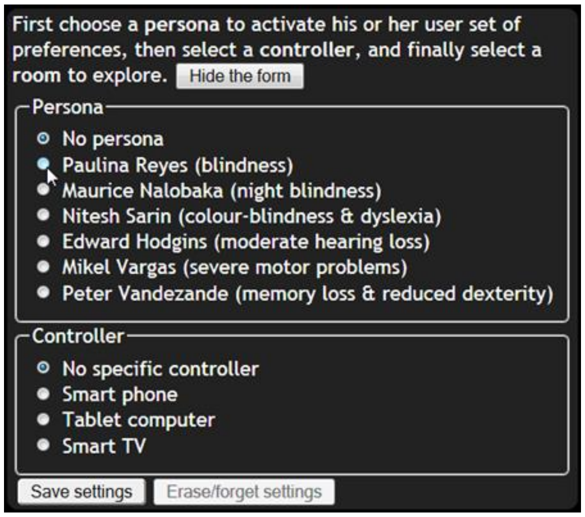
Currently, the Cloud4all smart house implementation is still in an early stage (see Section 4 for a sneak peek). It allows for manual selection of a persona (as "user simulation") and of a controller device (as "controller simulation" in a desktop browser). Once a stable version of the AccessForAll framework exists and a decent set of preference terms are available in the registry, we will implement this in our smart house demo to facilitate an automatic selection of suitable pluggable user interfaces and their "tweaking" at runtime.

As the Cloud4all approach is designed to perform automatic user interface adaptations based on different users and their controller devices, the user will not need to manually trigger such adaptations. However, the user will be able to control such adaptations in an appropriate way. Ultimately, this will need a "matchmaker" in the cloud (i.e., the matchmaker will be available as a service on the Internet), a key component of the Cloud4all architecture [17]. The matchmaker is responsible for identifying suitable runtime adaptation parameters for specific user preference sets and a specific context of use.

IV. The Cloud4all Smart House Demo: A Sneak Peek

The Cloud4all smart house demo employs the latest Web technologies, i.e., HTML5, CSS and JavaScript. Accessibility is an essential requirement for this demo, so, for example, all content — including content manipulated by JavaScript — is keyboard accessible and provides sufficient contrast between text and background. An early prototype using the Adobe Edge Preview tool was abandoned because it did not allow the required accessibility features.

The current demo allows the user to select one of seven personas to whose preferences the simulation should be adapted, and to select a specific controller to be simulated (see Fig. 1). The personas were selected from those created by the European projects AEGIS [18] and ACCESSIBLE [19]. For each of these personas, we created a *preference set* that describes the accessibility features and assistive technologies that they need. These personas are merely a convenient way of grouping and demonstrating various accessibility settings, not a way of classifying users. Tools that allow users to define their preferences are being developed in Cloud4all and other projects that contribute to GPII.


First choose a persona to activate his or her user set of preferences, then select a controller, and finally select a room to explore. Hide the form

Persona
- No persona
- Paulina Reyes (blindness)
- Maurice Nalobaka (night blindness)
- Nitesh Sarin (colour-blindness & dyslexia)
- Edward Hodgins (moderate hearing loss)
- Mikel Vargas (severe motor problems)
- Peter Vandezande (memory loss & reduced dexterity)

Controller
- No specific controller
- Smart phone
- Tablet computer
- Smart TV

Save settings | Erase/forget settings

Figure 1. The persona and controller selection panel.

After having chosen a persona and a controller, the user can select from a set of appliances to be controlled ("target devices"). For the sake of simplicity and demonstrability, the smart house demo includes a simulation of smart house appliances in the Web browser rather than having real appliances connected as a back-end. The target devices are displayed in a virtual smart house depicted as a floor plan (see Fig. 2).

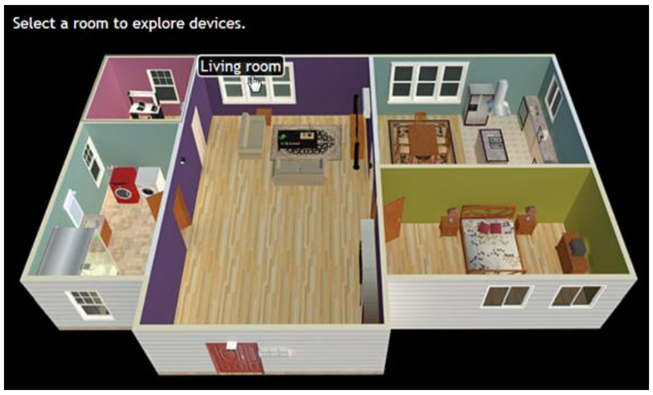


Figure 2. The overview screen for selection of a target device. The mouse cursor is positioned to select the living room. Selection can also be done via the keyboard alone.

After selecting the target device, the system provides a user interface for the target device in the browser; this user interface is adapted to the persona's preference set and the selected controller (see Fig. 3 and 4). In a later version of the simulation, the user will be able to explore the differences between the adaptations by switching to a different persona and/or controller after selecting a device.

As examples of currently implemented pluggable user interfaces, we present the control clients for a stove (Fig. 3) and a coffee machine (Fig. 4). Both user interfaces are implemented as Web clients and are available in two versions each: a "standard" and a self-voicing version for visually impaired users (without needing a native or third-party screen reader). The self-voicing versions announce necessary information and events upon user interaction. This is achieved by including pre-recorded audio files in the web client (via HTML5 <audio> tags). All Web client versions are keyboard accessible.

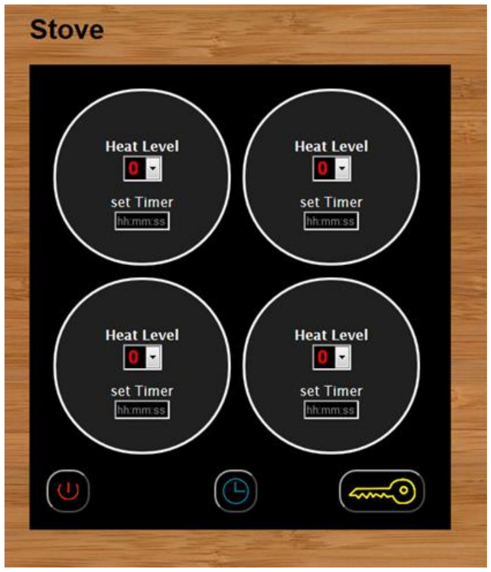


Figure 3. A Web client for a stove. It allows for individual control of four cooking plates (heat level 0-9 for each plate). In addition, it provides extra features, such as a child-lock to prevent a child from turning on the stove, and a timer for automatic switch-off for each plate.

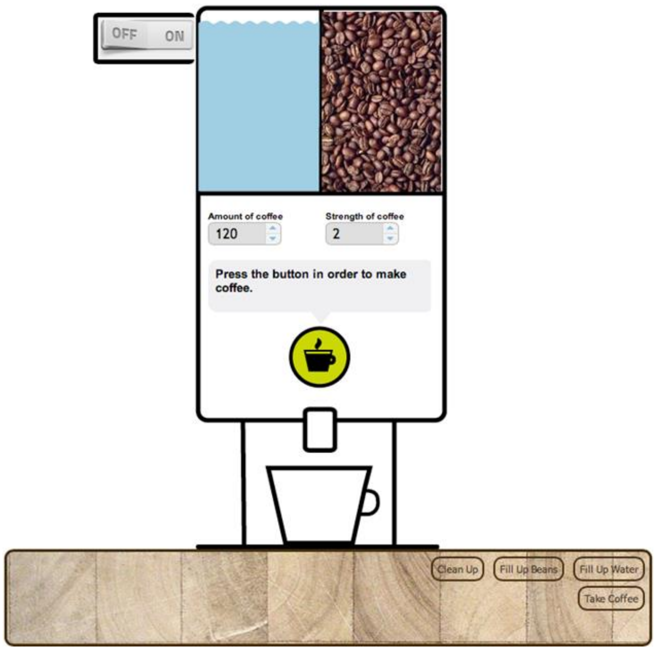


Figure 4. A Web client for a coffee machine. The upper-left button indicates that the coffee machine is switched on. The user can select the strength (1-3, i.e., weak to strong) and the size of the coffee cup (50ml to 150ml). The user receives notifications if either the water or the coffee reservoir is empty or the machine needs cleaning.

V. CONCLUSION & OUTLOOK

In this paper, we have introduced the smart house demo application of the European project Cloud4all. Its technical foundations are the URC framework, the UCH architecture, and the GPII preference set for user preferences and contextual conditions.

The smart house demo and its architecture address the previously mentioned industry concerns on adopting context-driven user interfaces (see Section I) in the following ways:

1. Based on the URC technology, designers can create a visual user interface as a "pluggable user interface" in a first step (see examples in Section IV). The pertinent abstract user interface can then be derived in a second step, resulting in the "user interface socket". The socket is required for enabling alternative (pluggable) user interfaces, possibly using other modalities.
2. For designing a visual user interface, Web designers can use the tools and technologies they are familiar with (in particular HTML5 editors and Web programming environments). However, they will have to add some JavaScript "glue" code manually for retrieving and displaying the values of the pertinent socket elements, and updating them upon user input. (This manual step could be eliminated by special extensions to the HTML editing tools.) Also, they will need to include some framework code (provided to them via a link) that will allow for tweaking the user interface at runtime based on a user preference set and contextual information. (This will also enable user-initiated adaptations at

runtime via a personal control panel, as currently developed by GPII.)

3. Regarding the control over the context-driven user interface at runtime, device manufacturers will have to live with the fact that their user interfaces are tweaked at runtime along previously known dimensions, such as font size, button size, and line height. However, they might construe completely third-party made user interfaces as damaging their corporate identity. Therefore, we need to make sure that the device manufacturer's user interfaces are used as default, and third-party user interfaces for special user groups (with disabilities) and special contexts are first approved by the device manufacturer before they are released to be used at runtime. This can be achieved via a certification process on the resource server which acts as an "app store" for user interfaces in our architecture. At the end, we hope, the resulting context-driven user interfaces will highly increase usability and accessibility of their devices, which will more than outweigh the device manufacturers' discomfort on giving away some control over the user interface at runtime.

The smart house demo application will soon be made available publicly as a showcase of pluggable user interfaces that adapt to a user's needs and preferences, to the controller's capabilities, and to situational constraints. It presents a showcase of emerging user interface adaptation technologies developed within GPII, which will allow for real context-driven user interfaces. The combination of the URC framework and context-driven runtime adaptations is deemed to be a major advancement in adaptive user interfaces for smart houses, Ambient Assisted Living environments, and beyond. URC allows the provision of a variety of pluggable user interfaces at design time, in order to accommodate for a heterogeneous user population with widely differing needs in terms of modalities and user interaction mechanisms. The client-side context-driven adaptation approach, based on user preferences, device capabilities and situational parameters, allows for low-level modifications ("tweaking") of user interface features at runtime, starting from one of the pluggable user interfaces provided at design time. We believe that such a hybrid approach is well suited to accommodate the needs of a broad range of users with disabilities, as well as providing enough incentives for mainstream industry to be adopted in the long run.

Currently, the Cloud4all smart house demo is still at an early stage, populated with a few exemplary appliances and user interfaces only. Further development work is planned to address:

- Full support for the AccessForAll (ISO/IEC 24751) framework, and its registry of preference terms. Based on a user's preference set, the matchmaker will be able to automatically provide a first approximation of runtime user interface, and to adapt it at runtime to better suit the user's preferences and needs.
- In addition to the automatic selection of an appropriate user interface, the user should be able to fine-tune its settings via a personal control panel. For example, a blind user would automatically be provided with a self-voicing user interface, for which they could fine-tune the speech rate and volume at runtime.
- Integration of more appliances and controller devices, together with pertinent user interfaces for all personas.
- Integration of a real identification feature for the user, employing a USB stick or NFC token rather than having to select a persona. (However, we will keep the persona selection panel for demonstration purposes.)
- Automatic detection of browser and runtime platform (including mobile devices) rather than having to select a controller device manually. Based on this information, the matchmaker will be able to automatically select an appropriate pluggable user interface and adapt it at runtime. (However, we will keep the controller selection panel for manual adaptions, for demonstration purposes.)
- Automatic adaptation of user interface features based on situational parameters of use, such as ambient noise and brightness levels. Again, the matchmaker can use this information to adapt a user interface at runtime.
- Usability studies on identifying appropriate ways of letting the user trigger and control automatic user interface adaptations. This is a core issue for overall user acceptance of the GPII framework since users want to be in full control of the user interface, but at the same time do not want to be distracted by frequent prompts on user interface issues.
- Usability evaluations and comparison of alternative user interfaces with regard to their usability and acceptance level for a specific user group (optional).

These activities are planned to be conducted in the course of the Cloud4all project. In addition, other parties are encouraged to contribute to this open-source effort within the GPII realm. Interested parties are welcome to contact the authors in this matter.

ACKNOWLEDGMENT

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 Grant 289016 (Cloud4all). The opinions herein are those of the authors and not necessarily those of the funding agencies.

REFERENCES

[1] "Cloud platforms Lead to Open and Universal access for people with Disabilities and for All" (Cloud4all) website: http://www.cloud4all.info (2013-10-17).

[2] Global Public Inclusive Infrastructure (GPII) website: http://www.gpii.net (2013-10-17).

[3] The Cloud4all public smarthouse demo website will be available at: http://smarthouse.remex.hdm-stuttgart.de.

[4] Hayes, P., Szekely, P., Lerner, R., 1985. Design alternatives for user interface management systems based on experience with COUSIN, Proceedings of the CHI’85 conference on Human factors in computing systems.

[5] Savidis, A., Stephanidis, C., 1995. Developing dual user interfaces for integrating blind and sighted users: the HOMER UIMS, Proceedings of the SIGCHI conference on Human Factors in Computing Systems.

[6] Myers, B., Hudson, S., Pausch, R., 2000. Past, present and future of user interface software tools. ACM Transactions on Computer–Human Interaction 7(1), 3–28.

[7] OASIS User Interface Markup Language (UIML) Technical Committee. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uiml (2013-10-17).

[8] Nichols, J.; Rothrock, B.; Chau, D.H.; & Myers, B.A.(2006). Huddle: Automatically Generating Interfaces for Systems of Multiple Connected Appliances. ACM Symposium on User Interface Software and Technology, UIST'06, October 15-18, 2006, Montreux, Switzerland.

[9] Gajos, K.Z.; Wobbrock, J.O.; & Weld, D.S. (2008). Improving the performance of motor-impaired users with automatically-generated, ability-based interfaces. In: Proceedings of CHI'08, Florence, Italy, 2008.

[10] Peissner, M.; Häbe, D., & Schuller, A. (2012). MyUI deliverable D2.2. Adaptation concept and Multimodal User Interface Patterns Repository. http://myui.eu/deliverables/MyUI_D2-2_final.pdf (2013-10-17).

[11] Klyne, G.; Reynolds, F.; Woodrow, C.; Ohto, H.; Hjelm, J.; Butler, M.H.; & Tran, L. (2004). Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. W3C Recommendation 15 January 2004. http://www.w3.org/TR/CCPP-struct-vocab/ (2013-10-17).

[12] Treviranus, J. (2009). "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, Vol. 17 Iss: 3 pp. 208 – 217.

[13] Vanderheiden, G.; and Zimmermann, G. (2005). Use of User Interface Sockets to Create Naturally Evolving Intelligent Environments. Proceedings of the 11th International Conference on Human-Computer Interaction (HCII 2005), July 22-27, 2005, Las Vegas, Nevada, USA.

[14] Zimmermann, G. (2008). ISO/IEC 24752:2008. Information Technology - User Interfaces - Universal Remote Console (5 parts). International Organization for Standardization (ISO).

[15] Thakur, P. & Zimmermann, G. (2012). Universal Control Hub 1.0. Technical Report. OpenURC Alliance. http://www.openurc.org/TR/uch1.0 (2013-10-17).

[16] Thakur, P. & Rosa, B. (2012). URC-HTTP Protocol 2.0. Technical Report. OpenURC Alliance. http://www.openurc.org/TR/urc-http-protocol2.0 (2013-10-17).

[17] Loitsch, C.; Stiegler, A.; Strobbe, C.; Tzovaras, D.; Votis, K.; Weber, G.; & Zimmermann, G. (2013). Improving Accessibility by Matching User Needs and Preferences. In Assistive Technology: From Research to Practice (pp. 1357–1365). IOS Press. doi:10.3233/978-1-61499-304-9-1357.

[18] AEGIS website: http://www.aegis-project.eu/ (2013-10-17). Project personas: http://www.aegis-project.eu/index.php?option=com_content&view=article&id=63&Itemid=53 (2013-10-17).

[19] ACCESSIBLE website: http://www.accessible-eu.org (2013-10-17).