Towards Universally Usable Smart Homes – How Can MyUI, URC and openHAB Contribute to an Adaptive User Interface Platform?

Lukas Smirek and
Gottfried Zimmermann

Stuttgart Media University
Stuttgart, Germany
smirek@hdm-stuttgart.de
gzimmermann@acm.org

Daniel Ziegler

Fraunhofer Institute for Industrial Engineering IAO
Stuttgart, Germany
daniel.ziegler@iao.fraunhofer.de

***Abstract*—The vision of the Smart Home carries high potential to support us in our daily life and makes it more comfortable. The benefits are relevant to all users including those with special needs like elderly people and people with disabilities. To provide the largest possible benefit to their user, Smart Homes have to adjust to the user and not vice versa. Thereby, it must be taken into account that the user group of the future Smart Home is quite heterogeneous, including users of all ages, different levels of familiarity with computers, and various forms and degrees of disabilities. Hence, universally usable user interfaces are a prerequisite for the success and acceptance of Smart Homes. This paper presents criteria for the provision of universal usability in Smart Homes, based on Schneiderman's research agenda on universal usability for web and other services. Three established frameworks – MyUI, Universal Remote Console (URC) and openHAB – are examined as to how they can contribute to an adaptive user interface platform for Smart Homes. Three possible architectures will be presented on how the three frameworks can be integrated to achieve maximum benefit for the user. We finally propose to use the Global Public Inclusive Infrastructure (GPII) approach for accommodating personal user preferences in Smart Homes.**

Keywords*–Adaptive User Interface; Abstract User Interface; Ambient Intelligence; Ambient Assisted Living (AAL); Global Public Inclusive Infrastructure (GPII); Smart Home; Universal Usability; Universal Remote Console (URC).

I. INTRODUCTION

The idea of universal usability is not new and could already be realized, to a great extend, in some traditional public domains like postal codes or telephone numbers [1]. However, the concept of universal usability should also apply in the field of Computer Science and Human-Computer Interaction (HCI). Today, the increasing presence of Information and Communication Technology (ICT) in everyday life situations leads to systems with increasing complexity [2]. Designing appropriate user interfaces for experienced users can be quite challenging, but it becomes even more difficult when considering a wide audience of unskilled users [1]. Hence, the problem of making computers and other ICT based services accessible to everyone extends far beyond the population of disabled and elderly users [2].

An even larger amount of people will be affected when Smart Homes and related technologies become available, yielding a great potential to support us in our daily life. This is true for average users, as well as those with special needs like elderly people, and people with disabilities. Technological developments in the sensor and wireless networks, miniaturization and increased computing power, as well as ongoing research in the field of HCI bring the vision of Ubiquitous Computing [3] and Smart Homes close to reality. This also means that computers, sensors and networked devices will surround us even closer in virtually every situation of our lives. It must be also taken into account that, although the underlying technologies for Smart Homes are already usable, the great breakthrough in terms of broad adoption has not yet taken place. One possible reason – among others – is that a large amount of research effort concerning Smart Homes was spent on technological improvements and pattern recognition so far, aside from some social considerations [4][5]. Only little work can be found concerning Smart Home users and the design of universally usable user interfaces for Smart Homes. Furthermore, we are facing a quite diverse market for smart homes with many proprietary solutions and only a few common standards. Consequently, another major problem is interoperability between different systems [6] and with that a lack of overarching usability concepts. Smart Homes are typically thought of having the following characteristics: context awareness, personalization, adaptive behavior, and anticipatory behaviour. These are without doubt essential elements of Ambient Intelligent systems. However, they frequently refer to the issue of adapting the Smart Home's behavior to the user, instead of focusing on the interaction between the user and their home.

Due to the huge variety of input/output channels in Smart Homes, interaction patterns will range from intended to not intended and from conscious to not conscious ones [7]. Although notable progress in the development of disappearing user interfaces has been made, enabling non-conscious and sometimes non-intended interaction between user and Smart Home (e.g., pattern recognition facilitating automatic control of lights, detection of unexpected situations like emergency situations, and automatic energy management [8]), there will always be situations in which the user wants or has to interact with the system in a conscious, intended and explicit way. Such situations can occur when users have to set up a system or when they need to choose between different options offered to them, or when the system needs confirmation. This kind of user interaction should be personalized, taking the users capabilities and needs into account.

Summing up, a solution for Smart Homes providing universal usability to users must on the one hand overcome the problem of providing appropriate interfaces for all types of users, and on the other hand it must integrate all types of Smart Home technologies, devices and services.

One established approach to overcome usage barriers are *adaptive user interfaces*, which can take the user's capabilities and other contextual data into account [9]. However, in most realized systems so far adaptive user interfaces can provide appropriate access to a single system only, solving the first part of the problem. To overcome the second part of the problem – integrating different and rapidly changing technologies – some middleware approaches [10][11] were proposed to mediate between the underlying, heterogeneous hardware.

This paper aims to address both sides of the problem. It discusses how MyUI, URC and openHAB can be integrated and combine the principles of adaptive user interfaces and their specific middleware approaches. Such a system should comprise at least three layers of abstraction. The bottom layer must provide an abstract view on devices, their functionalities and their internal states [10]. The middle layer must provide an abstraction of the tasks which can be performed in such a system and how devices can collaborate [12]. Finally, the top layer must provide an abstract view on the interaction between users and their Smart Home system [13].

The three technologies, which were chosen for investigation, cover all three layers. The URC technology [10] and openHAB [11] are abstracting from devices and services in a Smart Home system and the MyUI framework abstracts from applications and their tasks, as well as from specific interaction modalities.

MyUI [14] and URC [15] were both chosen due to their appropriate abstraction models. MyUI provides an abstraction model on the level of interaction between a user and an application, thus enabling an adaptation of the user interface. URC abstracts from devices and services connected to a Smart Home system so that they can be controlled with personalized user interfaces. Furthermore, the authors are involved in the development of both technologies and work together in the project *Prosperity4All* [16], which is part of the *Global Public Inclusive Infrastructure* and focuses on adaptations of user interfaces of ICT-based systems [17]. Finally, openHAB [11] was chosen due to the fact that it is an open source project that features a modular, extensible design.

The remainder of this paper is structured as follows. The next Section II will give an overview about current Smart Home related issues in HCI and adaptive user interfaces research. In Section III, requirements for a user interface platform are presented. Section IV introduces MyUI, the Universal Remote Console framework and the openHAB platform, and their possible contributions to adaptive user interfaces in Smart Homes. In Section V, we show how these technologies can be combined to harness their individual strengths and avoid their weaknesses, in three alternative architectures. Section VI then explains how the combined system can benefit from the adoption of personal preferences, as defined by the Global Public Inclusive Infrastructure (GPII). Finally, in Section VII, we summarize the content of this paper and draw some conclusions.

II. Related work

With the increasing influence of the Internet [1], as well as increasing importance of Hypermedia in public and domestic places [18], the question arose on how universal usability in electronic information and services can be made available to as many people as possible [1]. In 2000, Shneiderman proposed his research agenda on universal usability for web and other services [1], focusing on three challenges: technology variety, diversity of users, and gaps in user knowledge. Schneiderman and Richter mention that research on features for people with special needs and their inclusion in ICT can bring benefits to all users [1][2]. Particularly, Richter states that providing universal usability support for users with special needs can not be considered as orthogonal to the application [2].

Today, it is widely agreed that Smart Homes and the concept of Ambient Assisted Living can bring great benefit for a wide spectrum of users. Mavrommati and Darzentas present an overview over HCI issues related to Ambient Intelligence [7], and Saizmar and Kim suggest a three dimensional framework on HCI perspectives on Smart Homes [19]. Human, home and technology are pointed out as the main dimensions, and interaction between human/technology, technology/home and home/human are considered as relevant aspects. Saizmar and Kim also claim that HCI research in Smart Homes is limited and biased to specific situations [19]. This view is also shared by Mavrommati and Darzentas [7]. Abascal et al. criticize that, although many scenarios have been described in the field of Ambient Intelligence, the interface between the user and the system still remains unclear [20]. To improve acceptance of Ambient Intelligence and to make it capable to provide better life quality in a non-obtrusive way, Casas et al. point out the necessity to combine ongoing Ambient Intelligence technological developments with user-centered design techniques [21]. In the same vein, Mavrommati and Darzentas point to the necessity of focusing on a more user centered HCI perspective [7].

Studies like [22] and [23] have shown that also elderly people are willing to use Smart Home technologies for a longer independent life. It is acknowledged that Ambient Assisted Living technologies have the potential of providing safe environments for elderly people [24]. Nevertheless, at the moment technologies do not yet meet the needs of elderly people and current solutions overemphasize the importance of smart devices while either neglecting or lacking real implementations on the side of human interaction and human power [21]. Therefore, several authors have argued for a more user-centered view in the Ambient Assisted Living domain [21][25][26].

Kleinberger et al. [27] and Abascal et al. [20] are concerned with the design of appropriate interfaces in the field of Ambient Assisted Living. They come to the conclusion that natural and adaptive interfaces can bring great benefits to this field.

The PIAPNE Environment [20] is an adaptive Ambient Assisted Living system for elderly people based on three models: A user model (capabilities, permissions), a task model (user activity) and a context (environment) model. The system consists of multiple layers, including a middleware layer to bridge different network technologies and an intelligent service layer to which intelligent applications (interfaces) can be connected.

The DomoEsi Project is carried out at the Escuela Superior de Ingenieros de Sevilla and is mainly concerned with the problem of interoperability. The system is based on Universal Plug and Play (UPnP) as common interface from which software bridges to other Smart Home technologies can be build. Users can access the system via web browser, a Nintendo Wiimote controller or a voice interface. The different input modalities of the Wii controller (infrared camera, buttons, accelerometers) can be used to provide a simple adaptable interface for people with disabilities and with special needs [6].

Regarding the results from this literature research, we can state that so far a large amount of effort was spent on what is technologically possible rather than on user requirements. There is a growing awareness for a more user centered view on Smart Homes and related user interfaces. Nevertheless, existing systems either try to give a user access to a variety of devices and technologies [6][20] or provide adaptive user interfaces for a specific system [27]. An approach spanning over multiple Smart Home technologies, and providing an unconstrained set of personalized and adaptive user interfaces (for the sake of universal access) is still missing.

III. Requirements for an adaptive user interface platform

It must be taken into account that, when considering adaptation, both runtime and design time are essential parts of the process. At runtime, the user interacts with the system through the user interface. Different users come with different requirements regarding their needs and preferences, and the user interface should be universally usable to them. Before a user interface can be rendered at runtime, an application developer must specify some abstract model of the interaction between the user and the application, leaving some leeway of presentation and behavior to the renderer. This also means that components must be developed and designed during development time that can be later applied to the model and adapted to the user's needs at runtime [28]. It is also crucial that, for the successful deployment of adaptive user interfaces, the designer's requirements are taken into account [9].

A. Requirements from the user perspective

Shneiderman [1] presented a research agenda to achieve the goal of universal usability. Although the paper focuses on "web-based services and other services", the ideas contained therein are applicable to the field of Smart Homes and Ambient Intelligence.

The first challenge mentioned by Shneiderman is *technology variety*. Accordingly, problems exist due to a huge variety of hardware, software and network access technologies. The hardware and software aspects mainly relate to devices like PCs and laptops and the main aspect about network access is speed. Another closely related aspect of this problem is the fact that vendors and innovators like to continuously introduce new features and novelties in order to gain a competitive advantage. Unfortunately, this can have negative effects on usability for some users [1].

In the field of Smart Homes, the following similarities can be found. First, a large amount of devices, services and ideas are just on the edge of entering the market and most markets are expected to grow. Hence, these markets are very volatile. New models and new types of devices might appear or disappear from the market. Also, new companies might enter the market, or new protocols and standards might be introduced to control a Smart Home's appliances and services. Just like in all emerging markets, the succes of the Smart Home market will depend on the compatibility of existing with new evolving equipments. Probably, users can benefit the most by combining different systems and technologies in the same installation. This also includes the integration of already existing devices with new ones [6].

A proper approach to handle this challenge is *abstraction (U1)* – a way of providing a seamless and technology-agnostic view of the Smart Home to the user. The user should not have to care which protocol or home automation platform to use to control a certain device. They should be able to integrate any device easily in their Smart Home and use it according to their needs. Therefore, the installation and integration process should be supported by appropriate discovery mechanisms, and the devices should provide an *abstract user interface* [29] for seamless presentation on a variety of devices and environments. These also constitute important issues in the field of Ambient Assisted Living. The easier a new device can be integrated in an existing system, the lower are the costs for specialized personnel responsible for the installation and integration.

The second challenge mentioned by Shneiderman is *user diversity*. This concerns the different types of users using a certain system with a large variety of skills, prior knowledge, age, gender, and possible disabilities [1]. This becomes even more challenging when considering the heterogeneity of Ambient Assisted Living system users. Frequently, user interface designers have to cope with diverse and sometimes contradictory requirements for potential users; therefore, it is almost impossible to follow a design-for-all approach [20].

The solution to this problem are *pluggable user interfaces (U2)* [10][30]. Every user should be able to plug the user interface fitting best their needs to any system they want to control. Some users may want to use their smartphone with a touch screen to control a target system, others still like a regular remote control with buttons and a third group may want to use a speech interface. Finally, people with disabilities may use totally different ways to control their Smart Homes. One could think about Braille devices for blind people or a head mouse for people with motor impairments, for example. Furthermore, it is possible to employ different user interfaces providing different scopes of functionality.

The third and last challenge mentioned by Shneiderman are *gaps in user knowledge*. Concerns are due to the fact that many users do not know how to begin, what to choose in dialogue boxes or how to handle system crashes. Furthermore, the question is raised if users can begin with an interface that contains only basic features (say 5% of the full system) and become experts at this level within a few minutes. A closely related aspect of this issue is the different learning speeds of users when handling a new system [1]. In this paper, we want to address these issues in a larger view by looking at the *longterm development of user knowledge and capabilities*. The gap between the functionality a system offers and the functions which can be or want to be used by a user can dramatically vary over time. Users can gain additional knowledge about a system due to continous usage of it, but they can also become more confused due to unexpected updates and new functions. Also, user skills can increase over time (e.g., recovery from accidents or strokes) or decrease due to aging or continiuous proceeding impairments (e.g., hearing, vision, or Parkinson).

An adequate solution to this challenge are *adaptive user interfaces (U3)*, which take the user, the interaction situation and environmental conditions into account. In [28], user interfaces are modeled as layered systems, consisting of three layers: presentation and input events, structure and grammar, and finally content and semantics. Any property of a user interface can be adapted to the context of use, which includes the user, the platform and the environmental context. The latter also includes issues like sunlight or noise. Further context conditions could also be issues like walking or fast driving [29]. Referring to the presentation layer, it can be affected by changing parameters like font size or colors. Due to appropriate underlying structures, also the way how information is exposed to the user can be adapted – less information, displayed on smaller instead of larger screens, or video clips in sign language instead of text for users with hearing disabilities. Finally, it is also possible to change the amount of displayed information. It might be reasonable to increase the amount of information with the experience of the user or adjust dialogues to interaction situations and with that contribute to a decrease in the gap of user knowledge. Anyway, it is crucial that the adaptations made by a System are transparent to the users and do not confuse them. Even more, an apropriate design of such a system and its adaptation mechanisms can help and guide a user to familarize with it [13].

B. Requirements from the designer perspective

Requirements from the designer's point of view are discussed by Peissner et al. [9], and the following relevant criteria are mentioned.

First, *modularity and clearly defined interfaces (D1)* are mentioned. The subdivision of a system in smaller modules and well defined interfaces allows to exchange different components against each other and to build a variety of solutions. Also, more work can be shared in a project team or several teams can work in parallel.

The second prerequisite is *expandability (D2)*. This means that existing system modules can be scaled within the adaptive system or also added and extended at a later stage. At the beginning of a project, a subset of modules can be provided and continuously expanded, also by external experts.

The final prerequisite is *openness (D3)*. In order to fit best the users needs, experts must be able to contribute to the development of appropriate solutions. It is unlikely that one expert has the knowledge about all user groups, so it is important that different experts can make their contributions.

IV. Technologies

For the purpose of this paper, we focus on three selected technologies. We show how each technology can contribute to the requirements presented in the previous section.

A. MyUI

1) Description of MyUI: The European research project MyUI [14] has developed a pattern-based approach of an infrastructure for automatically generated adaptive user interfaces.

The main components of the MyUI infrastructure [13] are:
The Pattern Repository. In MyUI, all knowledge on various user interface design solutions and adaptations is contained in design patterns. There are several types of design patterns in MyUI. Among them, the interaction pattern is a key concept, providing suitable user interface components for a specific interaction situation. All MyUI design patterns are included in the publicly available pattern repository [31]. This enables the integration of the knowledge on design solutions of a broad range of experts for different domains.
The Abstract Application Interaction Model (AAIM). To enable the automatic generation and adaptation of user interfaces, MyUI provides an abstract format to define the interaction possibilities of a user with the application. This format is called AAIM and is based on UML2 State Machine Diagrams. It concentrates on the common aspects of all possible user interface variants and does not contain specific presentation modalities or user interface elements.
The User and Context Management Infrastructure. MyUI adapts user interfaces based on a user and context profile. To create and maintain this profile, it interprets relevant characteristics of the end user and the environment based on events detected by different sensors [32]. These sensors include physical sensors like ambient light and noise as well as virtual sensors which, for example, detect the user's interaction behavior.
The Adaptation Engine. The generation and adaptation during runtime happens in a three-stage process. First, information on the user and context and the currently used device features are compiled into a user interface profile which defines general characteristics of the user interface. In the second stage, this information is used to select the most suitable design patterns according to the current application's AAIM. In the last step, the selected components render the corresponding concrete user interface. Adaptations due to profile changes additionally include mechanisms to manage the transition from the current user interface instance to the new one. Multiple software components, together referred to as the MyUI Adaptation Engine, implement this complex process.
The Development Toolkit. The role of developers significantly changes when implementing applications using the MyUI infrastructure. Instead of implementing the concrete user interface and its interaction logic, the main task is to create the AAIM and to connect it to the underlying business logic. To support this task, MyUI provides an Eclipse-based toolkit for the creation of adaptive user interfaces.

For demonstration purposes, the MyUI project has built an interactive TV system. Among other things, this demonstrator provides a main menu, an e-Mail client and a weather forecast application. A set of patterns for these applications has been developed and documented in the pattern repository [31]. For example, the developed patterns provide solutions for font-size adjustments as well as for the adjustment of the screen complexity or the way information is displayed.

2) Strengths and weeknesses of MyUI: One of the main strengths of MyUI is its closeness to the user to whom a highly individualized interface is provided. The developed system is able to dynamically adapt the exposed user interface at runtime, taking the user and the environmental context into account (U3). Hence, MyUI clearly contributes to the aspect of adaptive user interfaces. Furthermore, the concept of patterns and their storage in the open repository enables developers to contribute new modules which can be used to support further adaptations for the user. This fits the criteria (D1), (D2) and (D3).

Unfortunately, only a limited number of patterns have been developed as of today. Most of them were developed for the demonstrators and focus on elderly people and patients recovering from strokes. Hence, in practice only a subset of possible users is supported.

Although with the AAIM an abstract model is provided, this does not contribute to an abstraction of the functionality and internal states of controllable devices and services, which can be found in a typical Smart Home. This is due to the fact that AAIM's goal is to build an abstract model of the interaction and the control flow between the user and the application. Still, AAIM might be useful for abstractions on a higher level like building interrelated tasks out of atomic elements.

B. Universal Remote Console (URC)

1) Description of URC: The URC framework is standardized in the international standard ISO/IEC 24752 which was first published in 2008 [15]. Currently, a revision is underway, and a new version is expected to be released by end of 2014. Technical guidelines and implementation guidance with regard to the URC ecosystem have been developed and are maintained by the openURC Alliance [10].

The main idea of the URC technology is to provide pluggable, portable and personalized user interfaces so that any device or service *(Target)* can be controlled by any controller that best fits the user's needs. The standard focuses mainly on electronic devices including the ones which can be found in Smart Homes. Thus, due to the provision of abstract user interfaces, use cases like the following are enabled. A person can change an old household device for a new one (e.g., a TV) while retaining the same familiar user interface, although the new device is produced by another company than the old one. Furthermore, it would be possible to control the same Target with different user interfaces. Also, in a hotel room, every guest could use their smartphone to control the TV and everyone could read all the labels on the screen in their native language [30].

To enable such scenarios, the Targets must provide an abstract user interface and a mechanism through which they can be controlled. In the URC framework, an abstract view on all targets is provided via the concept of *User Interface Sockets (or just Sockets)* and a corresponding description in XML format.

The internal states of a Target are represented in one or several instances of Sockets. These Sockets are the interface through which a target can be accessed by any Universal Remote Control. To familiarize controllers with any target, Targets must provide a corresponding User Interface Socket Description for each of their Sockets. These descriptions can contain:

- **variables:** for exposing dynamic content which can be changed by the Target or by the user;
- **commands:** to give the user access to a certain function of a device which cannot be performed by changing a single variable; and
- **notifications:** to send a message to the user in order to inform them about special situations when their attention is required.

Targets provide an abstract user interface to which any *pluggable user interface* can be connected, to fit the users needs. To build a specific user interface, additional information in form of labels is required. Therefore, the standard specifies an XML language for *Resource Sheets*. A Socket Description's elements can be referenced from Resource Sheets. It is usually possible that for a single Socket Description several Resource Sheets exist, for example one for each language.

The standard does not specify a certain network protocol for the URC technology. Instead, any network protocol can be used providing discovery, control and eventing. Today, we are surrounded by a large number of networked devices. Since it is unlikely that all of them are going to adopt the URC framework, the openURC Alliance follows a middleware approach which is called *Universal Control Hub (UCH)*.

The UCH is a profiling of the URC standard which enables non URC-compliant Targets and controllers to communicate with each other and make the benefit of pluggable user interfaces available to the user. The UCH virtually folds the connected Targets and controllers into a single gateway component. With the UCH approach, the User Interface Sockets no more run directly on the Target. Instead, they run inside the Universal Control Hub . The UCH provides *Target Adapters* and *User Interface Protocol Managers* (UIPMs) to connect Targets and controllers [33].

The internal state of Targets is represented by the corresponding Socket runtime component inside the UCH. A Target and the UCH connect to each other via the Target's preferred protocol (e.g., UPnP, KNX). Inside the UCH, a Target Adapter is responsible for the communication with the Target and for synchronizing the Target's internal state with any connected controller via the UCH. Controllers can communicate with the UCH via any protocol for which the UCH provides a UIPM. The UCH comes with a standard UIPM in form of an HTTP-based protocol (similar to REST) through which a controller can access and manipulate the Sockets running inside the UCH.

Aside from connecting to Targets and URCs, the UCH can provide additional user interface resources (e.g., from third parties) by connecting to a *Resource Server*. The user interface Socket Descriptions, Resource Sheets and other related resources described in the preceding paragraphs are located on the Resource Server.

Thus, the UCH provides a filtered set of pluggable user interfaces to the user, based on a specific use context. For example, this can involve the type of controller that is connected to the UCH: a desktop computer, a smartphone or something totally different like a Braille device or a speech interface. The user can now choose from the already filtered set the user interface that fits best their needs.

2) Strenghts and weaknesses of URC: The main advantages of the URC technology and the UCH infrastructure are: the abstraction from Targets (U1), and the concept of User Interface Sockets which serve as a basis for pluggable, personalized user interfaces – contribution to (U2); the existence of discovery processes and the availability of the Resource Server.

A great advantage is also the availability of the UCH which makes the whole framework accessible to non URC-compliant Targets and controllers. Furthermore, the framework is based on standards maintained by ISO/IEC and the openURC Alliance, thus, providing reliable APIs and XML languages for developers. Everybody is able to develop new adapters for Targets and URCs. The standards specify clear interfaces for these added modules (D1) which can be made available via the Resource Server and loaded at runtime. Hence the criteria (D2) and (D3) are also fulfilled.

One shortcoming is that at the moment only a few Target Adapters, Target Discovery Modules and Socket Descriptions are available for the UCH. However, the openURC Alliance actively promotes the technology and is working on the development of a vital ecosystem.

C. openHAB – Home Automation Bus

1) Description of openHAB: The open-source platform *openHAB* [11] follows a middleware approach addressing the large diversity of devices and network technologies in the field of home automation. There is currently no common language to bridge the different devices and home automation systems, and therefore the dependency on a certain vendor becomes a problem.

openHAB aims at integrating new technologies and devices in an existing home automation system through a community-based approach [11]. Thus, the development of the required software is not dependent on their vendors.

openHAB uses an OSGi based modular system for bridging different technologies and devices. *Bindings* are required to connect to a specific technology and device, which can be developed and deployed as an OSGi bundle.

To provide features like uniform interface or overarching automation rules, openHAB abstracts from specific devices through the concept of *items*. An item is a real or virtual variable of a device or service. Items must be defined and related to specific devices or services (with a specific IP address or unique identifier) in a separate configuration file.

Values of variables are stored inside the runtime system in a *stateful repository*, together with the variable names. The repository is continuously synchronized with connected devices and can be accessed by any component that needs information about the device status (e.g., the integrated rule engine for automation, or a connected user interface).

Beside the stateful repository, openHAB also features an *asynchronous eventbus*. OSGi bundles use the eventbus to inform other bundles about events, and also to be updated by other bundles on external events. Bundles can change items and trigger actions by *commands*, and this change will be disseminated on the eventbus by *status update* messages.

2) Strengths and weaknesses of openHAB: The most important advantage of openHAB is its support for existing and emerging home automation systems. Existing technologies like EnOcean, KNX, Z-Wave and others are supported through special bindings. New technologies can be easily supported through the development of new bindings, which can be deployed at runtime (D2). Furthermore, an already established community is continuously supporting the framework to stay up to date with the latest trends and developments in the field. From its beginning, openHAB focused on a modular approach reflected in the choice of OSGi (D1) and the contributions from a vital community (D3).

Weaknesses of openHAB are especially noticeable in the support of appropriate user interfaces. The framework is following an approach of a unified user interface to the home automation system with all connected devices. There are both native solutions for iOS and Android, as well as Web-based solutions. Some of the user interface components can be manually adapted, but there is very little support for adaptive elements. To define the elements shown in the user interface, a separate configuration file for every user may be created (this could be construed as an *adaptable* user interface). *Adaptive* aspects are only supported in identifying the device on which the user interface is running on, and adapting the user interface properly to the device's capabilities.

D. Contributions of the three platforms to an adaptive user interface platform

Focusing on the requirements adaptivity, personalization and abstraction (see Section III), the following issues are critical for a common platform for adaptive user interfaces in Smart Homes:

1) **MyUI** provides an environment to render and adapt a user interface to the user context during runtime. It also includes mechanisms helping to set up an initial user interface. Hence it serves the goal of exposing an *adaptive user interface*.
2) **The URC technology** with its concept of User Interface Socket Descriptions serves as a basis for pluggable user interfaces and thereby, as a basis on which MyUI can build upon. The User Interface Socket Descriptions provide an *abstract user interface* and with that an abstract view on Targets. The UCH is needed as middleware layer exposing sockets and their descriptions, and it serves as a gateway to the Resource Server which provides User Interface

Socket Descriptions and additional resources required to provide for *pluggable* user interfaces.

3) **openHAB** offers great opportunities to bridge different backend technologies. It is also a middleware hosting component to communicate with different devices using different protocols. Due to its concepts of items, commands and update events it also serves as an additional abstraction layer for heterogeneous home networks.

Although both URC and openHAB are middleware approaches that have some concepts in common, both are equally important and needed. In theory, both frameworks support different home automation system technologies and abstract user interfaces. In practice, openHAB supports more systems than URC does, but URC provides more sophisticated abstraction mechanisms. In addition, URC supports a discovery mechanism that is mandatory for good usability. So both technologies are required, in order to complement each other.

V. Possible architectures for integration

In this section, we present three different architectures, reflecting on how the above described technologies may be integrated into a common platform for adaptive user interfaces.

All approaches have the following aspects in common:

- **Stand-alone controller with optional MyUI runtime support:** The controller runs on a separate device that communicates with the UCH. This controller may execute an instance of the MyUI runtime for dynamic adaptations at runtime. Note that the MyUI controller has a light-weight coupling to the rest of the system via the URC-HTTP protocol (U2). Hence, it can be easily exchanged with any other module, providing a different kind of adaptation engine. It is also possible to run a simple Web interface on the controller, without any dynamic adaptation features.
- **Integrated Resource Server:** The UCH infrastructure and the MyUI framework both use an external server to store additional resources. The UCH connects to the Resource Server to obtain Socket Descriptions and additional resources required for building the pluggable user interfaces which are exposed to the controller. The MyUI framework uses the MyUI repository, which includes the patterns required for parameterizing and building the adaptive user interface. Since both servers supply additional resources for building individualized user interfaces, it makes sense to integrate them in a single one. Such a repository (Resource Server) can contain all URC resources, and all MyUI patterns.

A. Architecture 1: The three-component system

The first approach is a very loosely coupled architecture. All three major components, the controller (optionally with the MyUI runtime), the UCH and the openHAB runtime are deployed on their own, connected only via HTTP and appropriate Web interfaces. The controller connects to the UCH via the URC-HTTP protocol. The openHAB REST API is used to connect the UCH and the openHAB runtime.

Unfortunately, the openHAB framework is lacking appropriate functionality for the discovery of new targets. Hence, we have to devise a rather complex discovery process: After the discovery of a new target, the corresponding Target Discovery Adapter notifies the UCH about the new target. The UCH then connects to the Resource Server and obtains the related information. Now, the new target-related variables (as defined in the User Interface Socket Description) must be introduced in the openHAB runtime. However, this functionality is currently not supported by the openHAB REST API. In order to work around this problem, the architecture could be extended by a *Target Discovery Message Sender (TDMS)* (inside the UCH), and a *Target Discovery Message Receiver (TDMR)* (as OSGi bundle in the openHAB runtime). After the discovery of a new device in the UCH, the TDMS would send a message (e.g., via HTTP) with all relevant information to the TDMR. After the reception of a new message, the TDMR would update the openHAB configuration (i.e. the openHAB configuration file where items and related device addresses are specified). From this point, the UCH and the openHAB runtime would be synchronized via the openHAB REST API.

To avoid redundant status information being stored in the openHAB Item Repository, as well as in the Sockets of the UCH, we propose to implement *light-weight Sockets* in the UCH. That way it would be sufficient to store a Target's internal state only in the openHAB Item Repository. A controller request on a Target variable would then be sent to the UCH where the light-weight Socket would forward it to the openHAB REST API Target Adapter. The openHAB runtime would respond to the request, and the response would be forwarded by the light-weight Socket to the controller.

B. Architecture 2: Integration of openHAB into the UCH

For a deeper integration, we could integrate openHAB into the UCH via a Target Adapter. Target Adapters are protocol handlers enabling the control of connected Targets with the UCH by using a Target-specific protocol (e.g., UPnP, KNX). Developers can build Target Adapters for any protocol they want to support and load them at runtime into the UCH.

Since OSGi frameworks run from any Java application, it is possible to embed the openHAB runtime in a UCH Target Adapter serving as a proxy between the two frameworks. This specialized *openHAB Adapter* could be deployed in the UCH as a regular Target Adapter. Internally, it would have full access over the OSGi framework hosting the openHAB runtime. The OpenHAB Adapter would forward all communication from the UCH to the openHAB runtime and vice versa. The openHAB runtime would be responsible for direct communication to connected devices. Again, the discovery of new Targets must be done by the UCH and then configuration updates must be injected into the openHAB runtime, either through edits on the openHAB configuration file, or through messages sent on the openHAB eventbus.

The advantage of this architecture over the loosely coupled version is that the UCH can access the openHAB runtime in a direct way (including the eventbus) through the hosting OSGi framework, and not only through the openHAB REST API.

C. Architecture 3: UCH as openHAB Item repository

The two previously described architectures have the advantage of keeping to a great extend the independence of the three involved systems, but this also implies some disadvantages. Some problems like the discovery process may only be solved with extensive workarounds and the concept of Items and Sockets could lead to redundancies. Hence, a more integrated solution is desirable.

The openHAB framework and the reference implementation of the UCH are both written in Java. Both use similar abstraction models (Sockets, Item Repository), and keep internal states of connected devices. To send messages to devices, openHAB uses commands over the eventbus and appropriate bindings to translate them to the device-specific protocols. In the UCH, commands are described as part of Socket Descriptions and triggered through specific Target Adapters. The main idea of the following approach is to integrate the UCH into the openHAB framework, mainly due to a replacement of the openHAB Item Repository with the UCH Socket repository. This approach is possible because both systems are implemented in Java, and due to the concept of modularization provided by the OSGi framework. To realize such an approach, the following modifications in the UCH system must be made:

- First, it must be assured that there is an interface through which a controller can access the UCH. There must be a bridge between the servlets (belonging to the UCH) and the OSGi framework in which the sockets are deployed and on which the servlets must operate on. Therefore, the UCH's servlets must be registered at the executing HTTPService of the OSGi framework.
- Second, the UCH must publish an OSGi service to give other bundles access to administrated Sockets. The Socket variables now serve as Item variables. The service's key must be the one of the openHAB Item Repository.
- Third, the UCH discovery process must be modified as follows: After the discovery of a new device or service, a new socket must be instantiated, and at the same time, the device's or service's address/ID must be introduced to the openHAB configuration.

VI. Further Integration with GPII

The Global Public Inclusive Infrastructure (GPII) is an initiative for the development and installation of an infrastructure for adaptations of user interfaces of ICT-based systems. As a main component, it introduces a *personal preference set* carrying a user's individual needs and preferences, to be applied across many platforms. For example, a ticket machine in a public space might automatically display a larger font or a high-contrast mode if the user has set so in their preference set. The GPII personal preference consists of simple key-value pairs. It can be stored either locally (e.g., on a USB stick that the user carries around with them) or centrally on a dedicated secure server, depending on the user's choice.

Among the technologies that we have inspected in this paper, only MyUI has its own notion of user preferences. Although the vocabulary differs from that of the GPII preference set, both use a similar format (key-value pairs). Therefore, it should be possible to transform MyUI's vocabulary into the vocabulary used by GPII, and thus, take advantage of the interaction pattern knowledge base of MyUI. In the URC framework, a UCH may carry any user preference set, including the GPII preferences. openHAB does not define any data on user preferences.

In all three architectures described in Section V, the GPII personal preference set can be easily integrated on the front-end, i.e., on the controller code. This will provide additional benefits, such as:

- The system remembers the user's preferences, e.g., the choice of the preferred pluggable user interface.
- When the person uses a new controller for the first time, appropriate user interface adaptations could be suggested to the user based on their preferences for other controllers that they have already used.
- The system could propose advanced user interface settings that the user may not be aware of. The proposed settings would be derived from a rule base or from the analysis of a pool of users (statistical analysis).

VII. Summary and Conclusion

Future Smart Homes will offer user interfaces enabling explicit interaction between users and the devices and services they host. These user interfaces should be aligned to principles of universal usability and thus, be made accessible for as many users as possible, including older users and persons with special needs.

We have aligned the three challenges for the provision of universal usability for Web and other services, as described by Shneiderman, to the context of Smart Homes. This resulted in the concepts of abstract, personalized and adaptive user interfaces. Applying these concepts properly to Smart Homes, it is possible to bridge different technologies as well as to provide every user with a user interface that best fits their needs and preferences. Furthermore, the user interface can be adapted to environmental conditions and capabilities of the user that may change over the user's lifetime.

MyUI, the Universal Remote Console framework and openHAB were examined regarding their contribution to the three concepts. It was found that MyUI can provide a powerful adaptation engine and can fulfill the requirement of adaptive user interfaces. The URC framework provides, due to its abstract user interface, a platform for pluggable user interfaces on which adaptation frameworks like MyUI can build upon. Furthermore, the UCH can be used together with the openHAB runtime to bridge different technologies and protocols on the backend.

We have presented three approaches as various combinations of these technologies, ranging from a loosely coupled three-component system to a completely OSGi-based system. All three architectures cover the protocol layer up to the user interface layer. Thus, users are able to control their Smart Home, consisting of any combination of devices and services, independent of the required protocol, by using a controller that best fits their needs.

Finally, we have described how the personal preference concept of the Global Public Inclusive Infrastructure (GPII) can be integrated with the three architectures. Thus, personalized preference sets, stored locally or in the cloud, could support user interface adaptation, thereby providing personalized and cross-platform universal usability for everyone and wherever it is needed.

In the future, we plan to implement one of the presented architectures, possibly with further modifications, as part of the European Prosperity4all project [16]. In this joint implementation, the user interface shall adapt to the users needs, thereby taking a GPII preference set into account. The preference set will also include data from the AsTeRICS model reflecting a configuration of input devices for people with motor impairments [34]. The resulting implementation will be made available to the public as open-source release.

Acknowledgment

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2011) under grant agreement no. 610510, Prosperity4All (”Ecosystem infrastructure for smart and personalised inclusion and PROSPERITY for ALL stakeholders”). This publication reflects only the authors’ views and the European Union is not liable for any use that may be made of the information contained herein.

References

[1] B. Shneiderman, “Universal usability,” Communications of the ACM, vol. 43, no. 5, 2000, pp. 84–91.

[2] K. Richter and M. Hellenschmidt, “Interacting with the ambience: Multimodal interaction and ambient intelligence,” Interaction, vol. 19, 2004, p. 20.

[3] M. Weiser, “The Computer for the 21st Century,” Scientific American, vol. 265, no. 3, January 1991, pp. 66–75, URL: http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html [accessed: 2014-08-15].

[4] S. Solaimani, W. Keijzer-Broers, and H. Bouwman, “What we do and dont know about the Smart Home: an analysis of the Smart Home literature,” Indoor and Built Environment, 2013, URL: http://ibe.sagepub.com/content/early/2013/12/27/1420326X13516350.full.pdf+html [accessed: 2014-08-15].

[5] M. Chan, D. Estve, C. Escriba, and E. Campo, “A review of smart homes - Present state and future challenges,” Computer Methods and Programs in Biomedicine, vol. 91, no. 1, 2008, pp. 55–81, URL: http://www.sciencedirect.com/science/article/pii/S0169260708000436 [accessed: 2014-08-15].

[6] J. M. Maestre and E. F. Camacho, “Smart home interoperability: the DomoEsi project approach,” International Journal of Smart Home, vol. 3, no. 3, 2009, pp. 31–44.

[7] I. Mavrommati and J. Darzentas, “An overview of ami from a user centered design perspective,” in Intelligent Environments, 2006. IE 06. 2nd IET International Conference on, vol. 2, July 2006, pp. 81–88.

[8] A. Schmidt, “Interactive context-aware systems interacting with ambient intelligence,” Ambient intelligence, no. Part 3, 2005, pp. 159–178.

[9] M. Peissner, A. Schuller, D. Ziegler, C. Knecht, and G. Zimmermann, “Requirements for the Successful Market Adoption of Adaptive user interfaces for Accessibility,” in Universal Access in Human-Computer Interaction. Design for All and Accessibility Practice, ser. Lecture Notes in Computer Science, C. Stephanidis and M. Antona, Eds. Springer International Publishing, 2014, no. 8516, pp. 431–442, URL: http://link.springer.com/chapter/10.1007/978-3-319-07509-9_41 [accessed: 2014-08-15].

[10] “OpenURC,” 2014, URL: http://www.openurc.org/ [accessed: 2014-08-15].

[11] “openHAB - empowering the smart home,” 2014, URL: http://www.openhab.org/ [accessed: 2014-08-15].

[12] G. Zimmermann, “Open user interface standards-towards coherent, task-oriented and scalable user interfaces in home environments,” in Intelligent Environments, 2007. IE 07. 3rd IET International Conference on. IET, 2007, pp. 36–39.

[13] M. Peissner, D. Häbe, D. Janssen, and T. Sellner, “MyUI: Generating Accessible user interfaces from Multimodal Design Patterns,” in Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems, ser. EICS ’12. New York, NY, USA: ACM, 2012, pp. 81–90, URL: http://doi.acm.org/10.1145/2305484.2305500 [accessed: 2014-08-15].

[14] MyUI Consortium, “MyUI project Official Web Page,” 2014, URL: http://www.myui.eu/ [accessed: 2014-08-15].

[15] “Iso/iec 24752. information technology – user interfaces – universal remote console – 5 parts,” 2008.

[16] Prosperity4All, “Prosperity4All website,” URL: http://www.raisingthefloor.org/prosperity4all/ [accessed: 2014-08-15].

[17] “Home - gpii.net,” 2014, URL: http://www.gpii.net/ [accessed: 2014-08-15].

[18] J. Fink, A. Kobsa, and A. Nill, “Adaptable and adaptive information provision for all users, including disabled and elderly people,” New review of Hypermedia and Multimedia, vol. 4, no. 1, 1998, pp. 163–188.

[19] T. Saizmaa and H. Kim, “A holistic understanding of HCI perspectives on Smart Home,” in Networked Computing and Advanced Information Management, 2008. NCM’08. Fourth International Conference on, vol. 2. IEEE, 2008, pp. 59–65.

[20] J. Abascal, I. F. De Castro, A. Lafuente, and J. Cia, “Adaptive interfaces for supportive ambient intelligence environments,” in Computers Helping People with Special Needs. Springer, 2008, pp. 30–37.

[21] R. Casas, R. B. Marín, A. Robinet, A. R. Delgado, A. R. Yarza, J. Mcginn, R. Picking, and V. Grout, User modelling in ambient intelligence for elderly and disabled people. Springer, 2008.

[22] G. Demiris, M. J. Rantz, M. A. Aud, K. D. Marek, H. W. Tyrer, M. Skubic, and A. A. Hussam, “Older adults’ attitudes towards and perceptions of smart home technologies: a pilot study,” Informatics for Health and Social Care, vol. 29, no. 2, 2004, pp. 87–94, URL: http://dx.doi.org/10.1080/14639230410001684387 [accessed: 2014-08-15].

[23] G. Demiris, D. P. Oliver, G. Dickey, M. Skubic, and M. Rantz, “Findings from a participatory evaluation of a smart home application for older adults,” Technology and Health Care, vol. 16, no. 2, 2008, pp. 111–118.

[24] H. Sun, V. De Florio, N. Gui, and C. Blondia, “Promises and challenges of ambient assisted living systems,” in Information Technology: New Generations, 2009. ITNG’09. Sixth International Conference on. IEEE, 2009, pp. 1201–1207.

[25] O. B. Henkemans, K. Caine, W. Rogers, and A. Fisk, “Medical monitoring for independent living: user-centered design of smart home technologies for older adults,” in Proceedings of the Med-e-Tel conference for eHealth, telemedicine and health information and communication technologies, 2007, pp. 368–373.

[26] C. Röcker, “User-Centered Design of Intelligent Environments: Requirements for Designing Successful Ambient Assisted Living Systems,” in Proceedings of the Central European Conference of Information and Intelligent Systems (CECIIS’13), 2013, pp. 4–11.

[27] T. Kleinberger, M. Becker, E. Ras, A. Holzinger, and P. Müller, “Ambient intelligence in assisted living: enable elderly people to handle future interfaces,” in Universal access in human-computer interaction. Ambient interaction. Springer, 2007, pp. 103–112.

[28] G. Zimmermann, G. C. Vanderheiden, and C. Strobbe, “Towards deep adaptivity a framework for the development of fully context-sensitive user interfaces,” in Universal Access in Human-Computer Interaction. Design and Development Methods

for Universal Access, ser. Lecture Notes in Computer Science. Springer International Publishing, 2014, no. 8513, pp. 299–310, URL: http://link.springer.com/chapter/10.1007/978-3-319-07437-5_29 [accessed: 2014-08-15].

[29] G. Zimmermann, B. Jordan, P. Thakur, and G. Y., “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, ser. Assistive Technology Research Series, vol. 33. Vilamoura, Portugal: IOS Press.

[30] G. Zimmermann and G. Vanderheiden, “A dream... The universal remote console,” ISO Focus+, 2010, pp. 11–13, URL: http://trace.wisc.edu/docs/2010-URC-ISOFocus/ [accessed: 2014-08-15].

[31] M. Consortium, “Myui pattern browser,” 2014, URL: http://myuipatterns.clevercherry.com/index.php?title=Main_Page.

[32] O. Strnad, A. Felic, and A. Schmidt, “Context Management for Self-adaptive user interfaces in the Project MyUI,” in Ambient Assisted Living, ser. Advanced Technologies and Societal Change, R. Wichert and B. Eberhardt, Eds. Springer Berlin Heidelberg, 2012, pp. 263–272, URL: http://dx.doi.org/10.1007/978-3-642-27491-6_19 [accessed: 2014-08-15].

[33] G. Zimmermann and G. Vanderheiden, “The Universal Control Hub: An Open Platform for Remote user interfaces in the Digital Home,” Springer LNCS. Human-Computer Interaction. Interaction Platforms and Techniques., vol. 4551/2007, 2007, pp. 1040–1049, URL: http://www.accesstechnologiesgroup.com/pubs/ZimmermannVanderheiden2007-HCII/ [accessed: 2014-08-15].

[34] C. Veigl, C. Weis, K. Kakousis, D. Ibanez, A. Soria-Frisch, and A. Carbone, “Model-based design of novel human-computer interfaces – the assistive technology rapid integration & construction set (AsTeRICS).” IEEE, pp. 1–7.