Accessible Control of Distributed Devices

Supporting Persons with disabilities by Providing Adaptive Interaction

Lukas Smirek, Gottfried Zimmermann
Stuttgart Media University
Stuttgart, Germany
email: smirek@hdm-stuttgart.de
email: gzimmermann@acm.org

Christos Mettouris, Marios Komodromos, Achilleas Achilleos, George A. Papadopoulos
Department of Computer Science, University of Cyprus
Nicosia Cyprus
email: {mettour, mkomod, achilleas, george}@cs.ucy.ac.cy

Daniel Ziegler
Fraunhofer-Institute for Industrial Engineering IAO
Stuttgart, Germany
email: daniel.ziegler@iao.fraunhofer.de

Michael Beigl
Karlsruhe Institute of Technology
Karlsruhe, Germany
email: michael.beigl@kit.edu

Abstract – **It is nowadays common that the growing number of electronic devices and services used in all kinds of environments is coming along with a significant increase in the complexity of offered functionalities, as well as higher dependencies between devices and services. In this context, two big challenges can be identified: firstly, how to offer appropriate interactions to heterogeneous groups of users, and secondly, how to overcome interoperability problems of heterogeneous devices and services. In this paper, we present a prototype that attempts to address the above mentioned challenges by enabling the integration of Adaptive User Interfaces and Middlewares aiming to support real life Assistive Technologies and Internet of Things scenarios. When developing this prototype, a key requirement was to address adaptivity, not only at the graphical level, but rather from a generic interaction perspective in order to also support persons with motor impairments in Assistive Technologies scenarios. Therefore, four technologies/frameworks were integrated: Global Public Inclusive Infrastructure, MyUI, Assistive Technology Rapid Integration & Construction Set, as well as Universal Remote Console.**

Keywords-Adaptive User Interfaces; Assistive Technologies; Platforms Integration; Smart Environments; AsTeRICS; MyUI; URC; GPII.

I. INTRODUCTION

In today's world, we are facing a growing number of electronic devices and services both in public and private places. Thereby, not only the amount of devices is continuously increasing, but also their functionality and their interdependencies are becoming more complex and demanding. These developments can be seen as the precursor of a future Internet of Things (IOT) and the related domains of smart environments and smart homes.

From a user perspective, it has to be taken into account that the increasing complexity coming along with these developments must still be manageable for the end user. As mentioned in [1], providing only more functionality is by far not enough to increase user satisfaction. Even worse, it can lead to frustration. Hence, approaches like user centred design and appropriate user interfaces are of great importance.

This is even more emphasized when we consider that, as the amount of electronic devices and services is increasing both in private and in public places, more and more situations in our daily lives are being affected. Hence, not only skilled computer users will face these devices, but also almost everybody independent from age, computer skill level, social background or disability.

Summing up, in an IOT environment we see a heterogeneous user group facing a tremendous amount of heterogeneous devices and services. This raises new challenges for an appropriate design of user interfaces to interact in such an environment.

The first challenge - appropriate user interfaces for a heterogeneous user group – can be addressed by the well-known approach of adaptive user interfaces [2]. Furthermore, middleware architectures are a well-established approach to overcome the second challenge - interoperability problems of heterogeneous devices and services. However, most systems address only one of these challenges.

In this contribution, we present a prototype developed during the Prosperity4all project [3] that incorporates adaptive user interfaces and middlewares for device overarching use cases. A further development goal was to not restrict adaptivity only to a graphical user interface but also to enable adaptive support for persons with motor impairments in Assistive Technologies (AT) scenarios.

With this goal in mind, the following four technologies/frameworks were combined in order to provide a system for adaptive user interfaces in distributed environments: (I), the Global Public Inclusive Infrastructure (GPII) [4] is used as a means to transfer platform independent user preferences from one application to another and to infer appropriate settings to adapt the target system's user interface according to the user's needs. The adaptive user interface layer is formed by a cooperation of the (II), MyUI framework [5] and (III), Assistive Technology Rapid Integration & Construction Set (AsTeRICS) [6]. MyUI is used to provide an adaptive graphical user interface and to enable device overarching user interfaces, while AsTeRICS is used to accommodate persons with motor impairments in AT applications. Finally, (IV), the Universal Remote Console (URC) runtime [7] is used to mediate between the user interface layer and the devices and services that shall be controlled. It is also used to give third parties, e.g., assistive technology experts, the possibility to make their own contributions.

The remainder of the paper is structured as follows: In section (II), an overview about related work is provided. In section (III), a theoretical model on how to support a heterogeneous user group in an environment with heterogeneous devices is developed. In section (IV), the for technologies that were used to build the prototype are described in more detail. Their integration and their interdependencies are than described in section (V). Section (VI) concludes the paper with a discussion about the developed prototype and possible test cases, as well as open research questions.

II. Related Work

The potential of adaptive user interfaces to support persons with disabilities, as well as the requirements for their market adaptation are identified in [2]. Also, authors like Kleinberger et al. [8] and Abascal et al. [9] point to the potential of natural and adaptive interfaces in the field of Ambient Assisted Living (AAL).

In another dimension, a number of assistive technology systems have been developed, mainly in European Projects. The TOBI project [10] focuses on the design of non-invasive Brain/Neural Computer Interaction prototypes that combine existing Assistive Technologies and rehabilitation protocols. The aim is to improve people's communication by supporting access to devices such as virtual keyboards, internet, email, telephony, fax, SMS and environmental control. The BRAIN [11] project enhances intercommunication and interaction skills of disabled people via the development and integration of Brain-Computer Interfaces into practical assistive tools. The aim of the BRAIN system is to improve interaction of the user with people, home appliances, assistive devices, personal computers, internet technologies, and more. BrainAble's [12] main objective is to assist people with disabilities in overcoming exclusion from home and social activities by providing an ICT-based Human Computer Interface, as well as producing a set of technologies suitable for assisting people with physical disabilities regardless of cause.

OpenHAB [13] provides a scalable and modular architecture that integrates components and technologies in a single solution. OpenHAB is mainly concerned with the integration of devices from the Smart Home domain. The project is open-source with an active community, which enables new features and functionalities to be added, as with AsTeRICS [6]. The restriction of OpenHAB, in comparison to the AsTeRICS framework, is that an expert developer is needed to define in the form of text-based scripts the interactions among the components even for a simple assistive technology scenario. In contrast, the AsTeRICS system enables a non-expert AT designer to use a simple modelling interface to easily model or re-use existing models to provide the necessary assistive technology functionality to the user.

An adaptive Ambient Assisted Living system developed for elderly people is the PIAPNE Environment [9]. It is based on three models: A user model (capabilities, permissions), a task model (user activity) and a context (environment) model. The system has several layers. The middleware layer bridges different network technologies and the intelligent service layer can be used to connect intelligent applications interfaces (only software).

The DomoEsi Project [14] is carried out at the Escuela Superior de Ingenieros de Sevilla and focuses on interoperability problems but uses also some simple, adjustable hardware controller devices. Universal Plug and Play (UPnP) serves as common interface from which software bridges to other Smart Home technologies can be built. The system can be accessed 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 other, special needs.

III. Theoretical Background

In order to provide adaptive and device overarching user interfaces, two major layers of control are required: one being responsible for controlling the devices by the user and a second one for conducting the adaptation of the user interface. Furthermore, to make user preferences globally accessible, an internet based exchange mechanism must be available. Along with that, there must be an external repository for user interface components.

A. Device Control

As also described in [15] three layers of abstraction are required in order to control several devices by any adaptive user interface to integrate different devices and services. First of all, there must be a description available to give an abstract view on the devices' and services' internal states and functionalities *(da)*. Such descriptions can be seen as a kind of contract provided by the device and giving everyone the chance to access and operate it via its API. This is the lowest level of abstraction.

Next, there must be an abstraction from tasks. On this level the execution of functions on different devices and services is coordinated *(ta)*. Finally, a last layer is needed that abstracts from specific modalities and interaction between users and such a system *(ia)*.

B. Adaptive user interface Control

In order to control an adaptive user interface, three components can be usually distinguished. Here, we refer to the terminology of an afferent, an inferent and an efferent component as it is also used in [16]. The afferent component is responsible for collecting available and observable data about the context of use. The context of use comprises data about the user, the environment and the target platform. The data collected by the afferent component serve as input for the inferent component, in order to deduce relevant properties of the user interface that are needed to satisfy the users' needs in the current context. Finally, the efferent component is responsible for the generation of an appropriate user interface based on the conclusions made by the inferent component.

C. *Globally available user preferences and user interface components*

One characteristic of IOT environments is that the devices participating in an interaction situation, as well as the place where users would like to interact, can vary. Consequently, the part of the context of use that is related to the user must be globally available via the internet *(gu)*.

Moreover, users do not always know in advance under which kind of circumstances and with what kind of devices they will interact. Hence, a globally available repository for user interface components and user interfaces is needed that can be used for rendition *(gr)*. A further advantage of such a repository is that user interface experts and assistive technology experts can contribute alternative user interface fragments, even though the related device is already launched.

IV. TECHNOLOGY OVERVIEW

This section gives an overview of the technologies/frameworks used for this work.

A. *Global Public Inclusive Infrastructure (GPII)*

The broader vision of the GPII [4] is to provide adaptive user interfaces for persons with disabilities, in order to make all kinds of electronic devices and services accessible whenever and wherever they are needed. Aiming this, a cloud based infrastructure was built in order to transfer platform independent user preferences from one device to another and to infer appropriate user interface settings.

The general control flow works as follows: In a first step a personal device like a PC or Smart Phone must be configured according to the user's needs. These initial and device specific settings are used to deduce a platform independent user preference set. This preference set can be transferred to any other target device, either by a cloud service or by portable devices such as flash drives.
Due to the global availability of this preference set, any target device connected to the GPII can be customized to any user. Therefore, users can approach and log in to a target device of their interest. (e.g., ticket machine, PC in a public library).

Upon the authentication of the user, the target device contacts a service called matchmaker (local or cloud based) that is inferring the probably best fitting user interface settings for the user. Thereby it takes the user's preference set as well as a target device specific profile into account. The latter should include all available technologies on the target device that might help to assist the user in operating it.

In a final step, the proposed settings are sent from the matchmaker to a setting-handler being responsible for configuring and adjusting the target device to the user.

B. *MyUI framework*

With MyUI, Peissner et al. [17] present a framework to generate a wide range of user interfaces based on multimodal design patterns. It is designed to support runtime adaptations of the user interface aligned to the users' needs and preferences, characteristics and interaction capabilities of the used devices and conditions in the current environment. The process the MyUI runtime implements to create individualized user interfaces is separated into three steps (see Figure 1).

The first step is the user interface parametrization. For that, MyUI employs a mechanism to derive required user interface characteristics based on an existing context of use. The characteristics comprised in the resulting user interface profile include, but are not limited to information on screen complexity, layout structures, audio settings and navigation mechanisms. The user interface parametrization process is triggered each time changes to the underlying context information are detected to derive required runtime adaptations of the user interface. It is performed at least once when a certain user is recognized accessing the system.

The second step is the user interface preparation. MyUI aims at supporting adjustments of presentation formats and modalities as well as more extensive adaptations of interaction mechanisms and navigation paths. To permit the latter, an abstract description of the intended user interface is required. For that purpose, MyUI defines a graphical description language called the Abstract Application Interaction Model (AAIM) as an extension of UML2 behavioural state machines [18]. The states of the state machine represent the states of the interaction between users and the application. For each such interaction state an interaction situation is assigned which represents a certain interaction purpose together with the associated interaction options. Each time the current state in the application's AAIM changes (e.g., due to the user's previous action), MyUI selects the interaction pattern realizing the corresponding interaction situation that matches the current user interface profile.

In the last step, MyUI generates the final user interface based on the selected interaction patterns for the current interaction state. The user interface generation is based on common web technologies. In consequence the generated user interface is presented by a web browser. When the user interface for a certain interaction state is generated repeatedly, the transition between the former version and the new one is managed by adaptation patterns to ensure transparency of and control over the adaptation process for the user.

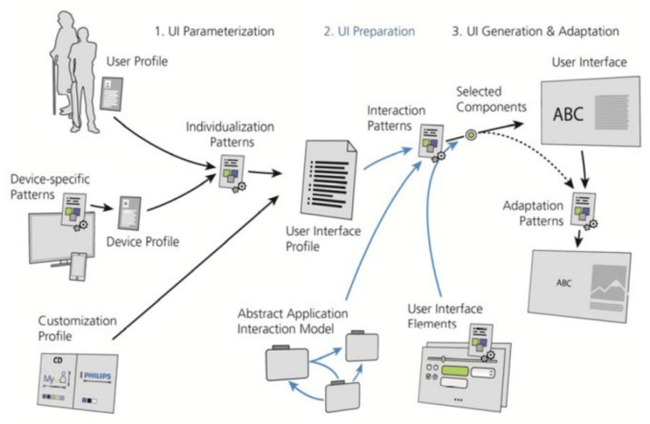

Figure 1. Overview of the MyUI generation and adaptation process [17]

C. *Assistive Technology Rapid Integration & Construction Set (AsTeRICS)*

Many people with disabilities worldwide are supported by assistive technologies [19] [20]. Assistive technology devices however often require individual adaptations, as they have been designed for explicit applications in specific environments. In this respect, routine activities of people with disabilities may be restricted, either because assistive technology devices cannot be adapted based on their needs, or because adaptations are too costly.

The AsTeRICS (Assistive Technology Rapid Integration & Construction Set) project [6] has built a hardware and software framework,that aims to reduce the time, effort and costs of developing assistive technology applications. It offers a flexible and affordable components set that enables building assistive functionalities, which can be highly adapted to the dynamically changing needs of each individual. The system is scalable, extensible and allows easy integration of new functionalities without major changes. It enables people with disabilities to gain access to the standard desktop computer, as well as to embedded devices and mobile services with no specialised user interfaces until present.

AsTeRICS provides the ability to define models, i.e., containers holding information describing the components that produce a specific assistive technology solution and their intercommunication (see Model Components in Figure 2). The components of a model can be classified into three categories: *sensors*, *processors* and *actuators*. Sensors monitor the environment and transmit input information to the rest of the model components. Processors are responsible for receiving, processing and forwarding this information. Finally, actuators receive data and carry out accordingly the desired actions.

AsTeRICS is constituted by two main components. The Web-enabled AsTeRICS Configuration Suite (WebACS), a graphical tool for creating assistive technology AsTeRICS models, and the AsTeRICS Runtime Environment (ARE) responsible for the deployment and runtime execution of the models (see Figure 2). The two components interact via the internet through a Representational State Transfer (REST) API on the AsTeRICS Runtime Environment. TheAsTeRICS Configuration Suite as well as the AsTeRICS Runtime Environment, were initially developed to communicate by using a proprietary protocol named ASAPI (**As**TeRICS **A**pplication **P**rogramming **I**nterface), which was not ideal because of the difficulty and complexity in learning and using. Instead, the REST protocol is easy to use and potential clients can easily adopt it for communication with the AsTeRICS Runtime Environment. The AsTeRICS Configuration Suite is currently under development in order to become web-enabled.

In this paper, we focus on the AsTeRICS Runtime Environment. The AsTeRICS Runtime Environment is a Java OSGi-based [21] middleware. The AsTeRICS framework offers a models@runtime approach, since the AsTeRICS Configuration Suite communicates with the AsTeRICS Runtime Environment to deploy the models and handle them at runtime. The AsTeRICS Runtime Environment enables the communication between OSGi bundles at runtime, which refers to the interactions and exchange of data between the sensor, processor and actuator components. Figure 2 presents the abstract view of AsTeRICS' architecture. The communication between the AsTeRICS Runtime Environment and potential assistive technology applications is conducted via the REST API [22].

The main requirement for making the AsTeRICS Runtime Environment RESTful was to provide remote access to its capabilities regarding integrating the very large set of AT functionalities offered by the implemented components into existing applications. In specific, the REST API offers platform and language independence, meaning that developers of the AsTeRICS Runtime Environment are able to reuse and integrate assistive technology functions into existing software applications without any concerns about the language and/or the platform used to implement and deploy the applications.

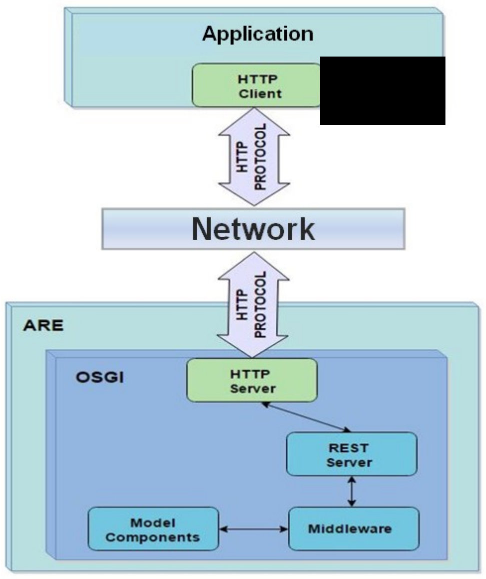

Figure 2. The REST-enabled AsTeRICS architecture.

D. Universal Remote Console (URC)

ISO/IEC 24752 [23] specifies the URC framework. The framework defines a mechanism for providing exchangeable and personalized user interfaces for all kind of electronic devices and services (so called "targets"). Thereby, user interfaces can range from pure software user interfaces to specialized hardware devices.

To enable exchangeable user interfaces, every target must provide an abstract description of its operational interface – the user interface socket description (or short "socket description"). This description contains information about variables that represent the target's internal state, commands that can be sent by controllers to the target and notifications that are sent the other way round.

Furthermore, for every element contained in a socket description additional resources like labels in different languages, help texts etc. can be defined. Moreover, additional resources are not limited in their complexity and diversity. Just to give some examples, a resource can also be a video in sign language, help texts or a whole user interface being related to one or several sockets, targeting a special user group.

Additional resources can either be stored locally on the target or on a dedicated resource server. The resource server provides third parties like user interface or assistive technology experts the possibility to make their own contributions to a target's user interface.

At runtime, any controller can connect to a target, read its socket description and download related resources from the resource server according to the user's needs and preferences to render a personalized user interface.

In order to make the principles of the URC framework applicable to non-standard compliant devices, the Universal Control Hub (UCH) [24] was developed. This middleware solution can connect to various targets via target adapters and download the related socket descriptions from the resource server. Socket descriptions are no more exposed by the targets themselves, but by the UCH, serving as central access point giving controllers a transparent view on them.

Figure 3. Universal Remote Console Infrastructure

Hence, controllers can connect to the UCH and use it as mediator to access all connected targets via a standardized way by using the URC-HTTP protocol [25]. The received messages are forwarded to connected targets by dedicated target adapters. The UCH can overcome interoperability problems by loading/using different target adapters. The UCH and all related components are shown in Figure 3.

E. Summary

Summing up, the four described technologies can contribute to a solution that enables the control of various devices and services via an adaptive user interface. Even more, the adaptive user interface can be provided wherever the user needs it. Doing so, the following advantages are gained: GPII is used to offer a platform independent user preference set to any location/environment wherever an adaptive user interface is needed *(gu)*. The adaptive user interface layer is formed by the AsTeRICS and/or the MyUI runtime by taking the user's needs and preferences into account. MyUI ships with its own real time adaptation engine to render an Abstract Interaction Model into a concrete user interface. Due to the concept of the Abstract Interaction Model the requirements of abstract interaction *(ia)* and abstract task descriptions *(ta)* are satisfied. Also, the usage of different AsTeRICS models contributes to *(ia).*

The two runtimes can then connect to the UCH that serves as central access point for controlling various distant devices and services. The concept of socket descriptions gives a transparent view on the connected targets *(da)* and different target adapters help to overcome interoperability problems. Furthermore, the UCH also connects to a dedicated resource server in order to provide user interface components for the technologies forming the user interface layer *(gr)*.

V. Integration and Prototype implementation

Since all the four frameworks GPII, AsTeRICS, MyUI and URC were originally developed independently from each other, a decision was made to look for a very lightweight and flexible way of integration. This gives system developers the freedom of choice concerning the frameworks they want to use, not limiting them in using one very complex monolithic system. Furthermore, for research purposes and considering also the further independent development of the four frameworks, a lightweight integration bears many advantages. A lightweight integration is also most in line with the GPII and Prosperity4all philosophy of having a set of independent components from which IT developers can choose.

In the current prototype, URC socket descriptions are used to satisfy the requirement of an abstract device layer *(da)*. Socket Descriptions and all connected targets can be accessed via the UCH that serves as central access point for any user interface.

The user interface layer is formed by the MyUI and, potentially by the AsTeRICS frameworks.

Concerning the communication between the user interface layer and the UCH, two cases can be distinguished. Either AsTeRICS or MyUIare used on their own to communicate directly with the UCH via the URC-HTTP protocol or AsTeRICS and MyUI are doing so cooperatively.

If an application needs to benefit both from the adaptive graphical user interface of MyUI, as well as from the various configuration possibilities provided by the AT-enabled input/output devices of AsTeRICS, both frameworks can be used in conjunction. In this case, MyUI communicates with the UCH via the URC-HTTP Protocol, while AsTeRICS and MyUI communicate via the operating system (OS) layer (see Figure 4). In specific, MyUI is used to provide users with an adaptive graphical user interface while AsTeRICS is used to control this graphical user interface by user specific input/output AT-enabled modalities (e.g., controlling the mouse via head movements, an AT functionality necessary for people with limited or no hand movement).

If AsTeRICS is used in cooperation with MyUI, the choice of which AsTeRICS model to use may depend only on the user preference set. This is due to the fact that the AsTeRICS task in this scenario is limited in controlling the MyUI graphical user interface, as well as perform other assistive technology enabled functionality (depending on the active model), while the coordination of interacting with URC targets is being conducted by MyUI (scenario 2, Figure 5).

More precisely, the coordination of tasks is specified in an Abstract Application Interaction Model. The choice of which model to select depends on the targets being currently connected to the UCH. The different Abstract Application Interaction Models available at a certain moment are provided by the UCH that downloads them from the URC resource server. The selected model is rendered by the MyUI adaptation engine into a real graphical user interface. The conducted adaptations are based on the information contained in the user preference set provided by the GPII. If special Input/Output modalities are required, the user preference set is also the resource that is used to specify certain needed AsTeRICS model characteristics that determine which AT models to be loaded on the AsTeRICS Runtime Environment from the resource server.

In case AsTeRICS is used on its own (no direct communication with MyUI), the selection of which AsTeRICS model to load depends not only on the user's preferences, but also on the targets available via the UCH, since AsTeRICS needs also to communicate with and control URC sensors/actuators (scenario 1, Figure 5).

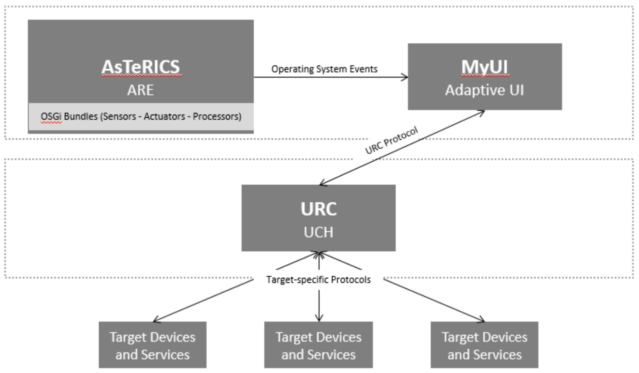
AsTeRICS
ARE
OSGi Bundles (Sensors - Actuators - Processors)
Operating System Events
MyUI
Adaptive UI
URC Protocol
URC
UCH
Target-specific Protocols
Target Devices and Services
Target Devices and Services
Target Devices and Services

Figure 4: An application uses the MyUI adaptive graphical user interface and the AT-enabled devices of AsTeRICS: MyUI communicates with UCH via URC-HTTP Protocol; AsTeRICS and MyUI communicate via OS layer

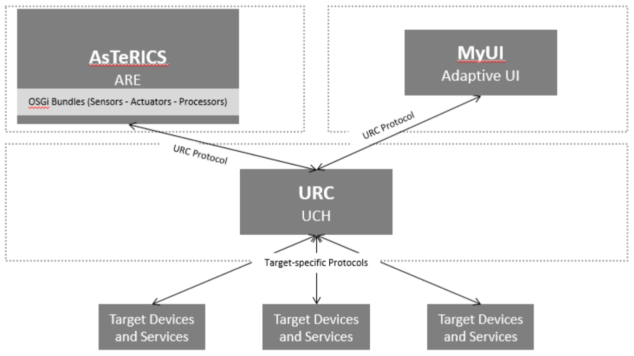
AsTeRICS
ARE
OSGi Bundles (Sensors - Actuators - Processors)
MyUI
Adaptive UI
URC Protocol
URC Protocol
URC
UCH
Target-specific Protocols
Target Devices and Services
Target Devices and Services
Target Devices and Services

Figure 5: AsTeRICS without direct communication with MyUI: needs also to communicate with and control URC sensors/actuators

An important issue to consider is that the requirements of an abstract device overarching task description *(ta)* and the one of an abstract interaction description *(ia)* are only fulfilled by making use of the MyUI engine and theAbstract Application Interaction Model. The latter can be used to specify a sequence of function calls to different targets and with that the execution of tasks.

VI. Discussion and open Research questions

Although the current prototype provides an adaptive user interface layer, there is space for further improvements, mainly to the continuity of the adaptation process. At this stage, the system provides only in some parts a continuous adaptive user interface, while in others only initial adaptivity is supported. Due to the continuous monitoring of the user and other connected sensors, the MyUI interface provides mechanisms for continuous adaptations. This is different for the user interface parts that are related to the AsTeRICS runtime. At this stage, AT model loading is specified by the user's preference set and there is no feedback mechanism that could be used by the MyUI engine to refine the parameters being set within the loaded AsTeRICS model, i.e., on model component level. MyUI, as with any other application, can interact with the AsTeRICS Runtime Environment via the REST API only on model management level, i.e. determine which model to start, stop, pause, etc., but not on model component level, e.g., which sensors to use within a model.

Furthermore, based on user needs and preferences, only one predefined AsTeRICS model can be loaded. This requires that the related hardware components are always available on the target system. If this is not the case, there is no alternative model to be loaded. Hence, a prioritized list of alternative AsTeRICS AT models for a particular set of user needs and preferences could be specified as a future improvement. Even more interesting would be to investigate whether it would be possible to use a GPII matchmaker instance to infer parameters or even a whole AsTeRICS model by taking a user preference set into account.

More importantly, there is the need for an appropriate security mechanism that protects the system from malware injection. Since any third party can contribute its solutions and make them available via the Resource Server, there is always the risk of introducing malware. To cope with such issues, one could think about a review process for resources, as it is known from some app stores for mobile applications. The current implementation is lacking such techniques and also GPII itself is still requiring an appropriate security framework.

Next, there is the need for appropriate user interface development tools. At this moment there are tools available to create URC sockets as well as tools to create MyUIAbstract Application Interaction Models. However, this can be done only separately. All in all, there is a need to define a development process for adaptive user interfaces that coordinate different target devices. It is not yet clear how such a process will look like, but it is for sure that future development tools need to provide a tighter integration between the abstract target descriptions and the abstract task descriptions.

Finally, appropriate testing needs to be conducted. We plan to test the proposed integrated prototype by using real users in real environments. The aim will be to address their real needs in terms of accessibility, assistive technologies and smart interfaces.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union Seventh Framework Program (FP7/2007-2011) under grant agreement no. 610510, Prosperity4All (”Ecosystem infrastructure for smart and personalized 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] C. Rich, “Building Task-Based User Interfaces With ANSI/CEA-2018,” 2009, Computer , 42 (8), 20-27.

[2] ‘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, . [online]. Available from: http://link.springer.com/chapter/10.1007/978-3-319-075099 41 2016.03.30

[3] Prosperity4All | One-size-fits-one digital inclusion . [online]. Available from: http://www.prosperity4all.eu/ 2016.03.30

[4] Home | gpii.net. [online]. Available from: http://www.gpii.net 2016.03.30 2016.03.30

[5] MyUI Consortium: MyUI project Official Web Page. [online]. Available from: http://www.myui.eu/ 2016.03.30

[6] AsTeRICS Homepage. [online]. Available from: http://www.asterics.eu/index.php 2016.03.30

[7] openURC Alliance: OpenURC. [online]. Available from: http://www.openurc.org/ 2016.03.30

[8] ‘T. Kleinberger, M. Becker, E. Ras, A. Holzinger, and P. M ¨uller’, “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.

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

[10] “TOBI: Tools for Brain Computer Interaction. [Online] Available from: http://www.tobi-project.org 2016.03.30

[11] BRAIN: Brain-computer interfaces with Rapid Automated Interfaces for Nonexperts. [online]. Available from: https://www.brain-project.org/ 2016.03.30

[12] BrainAble: Autonomy and social inclusion through mixed reality Brain-Computer Interfaces: Connecting the disabled to their physical and social world. [Online] Available from: http://www.gtec.at/Research/Projects/BrainAble, 2016.03.30

[13] “openHAB” . [online]. Available from:http://www.openhab.org 2016.03.30

[14] '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.

[15] ‘L. Smirek, D. Ziegler, and G. Zimmermann’, “Towards Universally Usable Smart Homes – How Can MyUI, URC and openHAB Contribute to an Adaptive User Interface Platform?,” CENTRIC 2014 : The Seventh International Conference on Advances in Human-oriented and Personalized Mechanisms,Technologies, and Services, 2014

[16] M. Peisner, “Entwurfsmusterbasierter Ansatz für adaptive Benutzungsschnittstellen zur Überwindung von Nutzungsbarrieren, FRAUNHOFER VERLAG, SCHRIFTENREIHE ZU ARBEITSWISSENSCHAFT UND TECHNOLOGIEMANAGEMENT, Vol 12,

[17] ‘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, . [online]. Available from: http://doi.acm.org/10.1145/2305484.2305500 2016.03.24

[18] Object Management Group (OMG): Unified Modeling Language Superstructure, V2.4.1. [online]. Available from: http://www.omg.org/spec/UML/2.4.1/Superstructure 2016.03.30

[19] Eurostat: Population and Social Conditions: Percentual Distribution of Types of Disability by Sex and Age Group. [online]. Available from: http://ec.europa.eu/eurostat/statistics-explained/index.php/Disability_statistics_-_prevalence_and_demographics 2016.03.30

[20] Assistive Technologies: Principles and Practice (2nd Edition) (15 December 2001) by Albert M. Cook, Susan Hussey

[21] OSGi Alliance Homepage. [Online] Available from: http://www.osgi.org/Main/HomePage

[22] ‘L. Richardson, M. Amundsen, and S. Ruby’, “RESTful Web APIs”. O'Reilly Media, September 2013.

[23] “Iso/iec 24752. information technology – user interfaces – universal remote console – 5 parts,” 2014.

[24] 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

[25] “openURC Alliance: URC-HTTP Protocol 2.0 (ATR)” . [online]. Available from: http://www.openurc.org/TR/urc-http-protocol2.0-20131217/ 2016.03.30