D203.2 Prototypes of development and configuration toolkits

Project Acronym: Prosperity4All
Grant Agreement number: FP7-610510
Deliverable number: D203.2
Work package number: WP203
Work package title: Collaborative development tools / environments
Authors: Stefan Parker (KI-I), Lukas Smirek (HdM), Daniel Ziegler (FhG)
Status: Final
Dissemination Level: Public
Delivery Date: 17/08/2015
Number of Pages: 38
Keyword List

WebACS, ACS, AsTeRICS, URC, Socket Builder, Persona Browser, MyUI, User Interface Development Toolkit, AAIM

Version History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Author</th>
<th>Organisation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18/06/2015</td>
<td>Stefan Parker</td>
<td>KI-I</td>
<td>TOC v1</td>
</tr>
<tr>
<td>2</td>
<td>19/06/2015</td>
<td>Lukas Smirek</td>
<td>HdM</td>
<td>URC parts added</td>
</tr>
<tr>
<td>2</td>
<td>22/06/2015</td>
<td>Daniel Ziegler</td>
<td>FhG</td>
<td>TOC for MyUI</td>
</tr>
<tr>
<td>3</td>
<td>10/07/2015</td>
<td>Daniel Ziegler</td>
<td>FhG</td>
<td>Adaptive User Interface Development Toolkit</td>
</tr>
<tr>
<td>4</td>
<td>13/07/2015</td>
<td>Stefan Parker</td>
<td>KI-I</td>
<td>chapters on WebACS added</td>
</tr>
<tr>
<td>5</td>
<td>21/07/2015</td>
<td>Lukas Smirek</td>
<td>HdM</td>
<td>added chapters on URC socket builder and persona browser</td>
</tr>
<tr>
<td>6</td>
<td>22/07/2015</td>
<td>Stefan Parker</td>
<td>KI-I</td>
<td>consolidation of documents, summary, keywords</td>
</tr>
<tr>
<td>7</td>
<td>07/08/2015</td>
<td>Stefan Parker</td>
<td>KI-I</td>
<td>incorporated comments and suggestions from peer review</td>
</tr>
<tr>
<td>8</td>
<td>07/08/2015</td>
<td>Daniel Ziegler</td>
<td>FhG</td>
<td>Incorporated comments and suggestions from peer review</td>
</tr>
<tr>
<td>9</td>
<td>07/08/2015</td>
<td>Gregg Vanderheiden</td>
<td>RfF</td>
<td>Alternative Text added for Figure 1</td>
</tr>
<tr>
<td>10</td>
<td>13/08/2015</td>
<td>Lukas Smirek</td>
<td>HdM</td>
<td>incorporated comments and suggestions from peer review</td>
</tr>
<tr>
<td>11</td>
<td>17/08/2015</td>
<td>Stefan Parker</td>
<td>KI-I</td>
<td>Document finalised</td>
</tr>
</tbody>
</table>

Ecosystem infrastructure for smart and personalised inclusion and PROSPERITY for ALL stakeholders
www.prosperity4all.eu
Table of Contents

Executive Summary ...................................................................................................................... 1

1 Contribution to the global architecture .................................................................................. 2
   AsTeRICS .................................................................................................................................. 2
   URC Socket Builder .................................................................................................................. 2
   The Persona Browser ............................................................................................................... 3
   Adaptive User Interface Toolkit ............................................................................................... 3

2 Initial Prototype of WebACS .................................................................................................. 5
   2.1 Concept.............................................................................................................................. 5
      2.1.1 Initial Situation ............................................................................................................. 5
      2.1.2 Goals and Design Targets .......................................................................................... 6
   2.2 Architecture ........................................................................................................................ 6
   2.3 Implementation ................................................................................................................... 7
   2.4 Future Work ....................................................................................................................... 8

3 URC Socket Builder ................................................................................................................. 9
   3.1 Concept.............................................................................................................................. 9
      3.1.1 Initial Situation ............................................................................................................. 10
      3.1.2 Goals and Design Targets .......................................................................................... 11
   3.2 Current Status .................................................................................................................... 11
   3.3 Future work ....................................................................................................................... 13

4 Persona Browser .................................................................................................................... 14
   4.1 Concept.............................................................................................................................. 14
      4.1.1 Initial Situation ............................................................................................................. 15
      4.1.2 Goals and Design Targets .......................................................................................... 15
      Open Platform .......................................................................................................................... 16
      Search and filter Information ................................................................................................ 16
   4.2 Report on first prototype .................................................................................................... 16
   4.3 Future work ....................................................................................................................... 18

5 Adaptive User Interface Development Toolkit ......................................................................... 19
5.1 Concept ................................................................................................................................. 19
5.1.1 Initial Situation ..................................................................................................................... 19
5.1.2 Goals and Design Targets .................................................................................................. 20
5.2 Architecture and Implementation .......................................................................................... 23
5.2.1 Development Toolkit User Interface .................................................................................. 23
  5.2.1.1 Graphical AAIM Editor .................................................................................................. 23
  5.2.1.2 Textual AAIM Editor .................................................................................................... 26
5.2.2 Relation to the Runtime Environment ............................................................................... 27
  Runtime Environment to Development Toolkit ........................................................................ 27
  Development Toolkit to Runtime Environment ........................................................................ 28
5.3 Future Work ........................................................................................................................... 28
6 References ................................................................................................................................ 30

List of Figures

Figure 1: Overall Picture of Prosperity4all .................................................................................. 4
Figure 2: AsTeRICS Configuration Suite 2.6 .............................................................................. 5
Figure 3: Example of an AsTeRICS model with active Property Editor at the right ................ 6
Figure 4: WebACS with two models open .................................................................................. 7
Figure 5: Screenshot of the Socket Builder Tool (see picture properties for a textual description) ................................................................................................................................................................. 13
Figure 6: Illustration of the content of the PersonaBrowser ...................................................... 15
Figure 7: Screenshot showing the information contained in the PersonaBrowser .................... 17
Figure 8: Detail view: Showing all relevant WCAG 2.0 success criteria for hearing impaired people. ........................................................................................................................................................................... 18
Figure 9: MyUI Development Toolkit with generated PHP Code and AAIM Model Editor [28] ........................................................................................................................................................................ 19
Figure 10: MyUI Preview Browser and profile adjustment sliders [28] ........................................ 20
Figure 11: AAIM Overview with states represented by wireframes .......................................... 24
Figure 12: Details view of one state/screen of the AAIM .............................................................. 24

Ecosystem infrastructure for smart and personalised inclusion and PROSPERITY for ALL stakeholders
www.prosperity4all.eu
Figure 13: Different possibilities to assign interaction situations............................... 25
Figure 14: Typeahead for textual expressions showing only valid possibilities ................. 26

List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full form</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAIM</td>
<td>Abstract Application Interaction Model</td>
</tr>
<tr>
<td>AAL</td>
<td>Ambient Assisted Living</td>
</tr>
<tr>
<td>ACS</td>
<td>AsteRICS Configuration Suite</td>
</tr>
<tr>
<td>AoD</td>
<td>Assistance on Demand</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ARE</td>
<td>AsTeRICS Runtime Environment</td>
</tr>
<tr>
<td>AsTeRICS</td>
<td>Assistive Technology Rapid Integration &amp; Construction Set</td>
</tr>
<tr>
<td>AT</td>
<td>Assistive Technology</td>
</tr>
<tr>
<td>C4A</td>
<td>Cloud4All</td>
</tr>
<tr>
<td>D</td>
<td>Deliverable</td>
</tr>
<tr>
<td>DoW</td>
<td>Description of Work</td>
</tr>
<tr>
<td>DSL</td>
<td>Domain-Specific Language</td>
</tr>
<tr>
<td>DSpace</td>
<td>DeveloperSpace</td>
</tr>
<tr>
<td>FhG</td>
<td>Fraunhofer Gesellschaft</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>GPII</td>
<td>Global Public Inclusive Infrastructure</td>
</tr>
<tr>
<td>HdM</td>
<td>Hochschule der Medien</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>ICT</td>
<td>Information and Communications Technology</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organization for Standardization</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
</tbody>
</table>

Ecosystem infrastructure for smart and personalised inclusion and PROSPERITY for ALL stakeholders
www.prosperity4all.eu
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full form</th>
</tr>
</thead>
<tbody>
<tr>
<td>KI-I</td>
<td>Kompetenznetzwerk Informationstechnologie zur Förderung der Integration von Menschen mit Behinderungen</td>
</tr>
<tr>
<td>KPI</td>
<td>Key Performance Indicator</td>
</tr>
<tr>
<td>MyUI</td>
<td>Mainstreaming Accessibility through Synergistic User Modelling and Adaptability</td>
</tr>
<tr>
<td>P4A</td>
<td>Prosperity4all</td>
</tr>
<tr>
<td>PHP</td>
<td>PHP: Hypertext Preprocessor</td>
</tr>
<tr>
<td>R&amp;D</td>
<td>Research and Development</td>
</tr>
<tr>
<td>RERC</td>
<td>Rehabilitation Engineering Research Center</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>SP</td>
<td>Sub-Project</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>UCH</td>
<td>Universal Control Hub</td>
</tr>
<tr>
<td>UCY</td>
<td>University of Cyprus</td>
</tr>
<tr>
<td>URC</td>
<td>Universal Remote Console</td>
</tr>
<tr>
<td>UX</td>
<td>User Experience</td>
</tr>
<tr>
<td>WAI</td>
<td>Web Accessibility Initiative</td>
</tr>
<tr>
<td>WCAG</td>
<td>Web Content Accessibility Guidelines</td>
</tr>
<tr>
<td>WebACS</td>
<td>Web-based AsTeRICS Configuration Suite</td>
</tr>
<tr>
<td>WP</td>
<td>Work Package</td>
</tr>
<tr>
<td>WPF</td>
<td>Windows Presentation Foundation</td>
</tr>
<tr>
<td>WYSIWOG</td>
<td>What You See Is What Others Get</td>
</tr>
</tbody>
</table>
Executive Summary

This deliverable is about the prototypes of the development and configuration toolkits. In specific this are the following prototypes:

- WebACS
- URC Socket Builder
- Persona Browser
- Adaptive User Interface Development Toolkit

The WebACS is the platform-independent implementation of the AsTeRICS Configuration software. Used technologies are mainly JavaScript and HTML5. There is a prototype with a number of basic functions available at Github, which can be viewed and tested by the interested reader.

The MyUI project has created an adaptive user interface development toolkit based on Eclipse. Key feature of the development toolkit is the graphical editor for the Abstract Application Interaction Model (AAIM). In this context, the following improvements are done within the P4a project:

- Provide more direct feedback on resulting user interfaces
- Improve the overall usability of the model editor
- Upgrade the AAIM to be used as a communication asset in development
- Support main stream software development practises
- Independence of server-side technology

The URC Socket Builder is a tool supporting developers working with the URC framework. The URC Socket Builder and its extensions developed in the context of the P4A project will provide means to connect other user interface technologies to the URC framework. So far the following steps have been taken:

- Testing and local deployment of created sockets
- Improvement of accessibility for screenreaders
- Improvement of keyboard navigation
- Improvement of system stability and bug fixing

The Persona Browser’s underlying principle is to use personas as representation and illustration of accessibility guidelines for the developer. Each success criterion of the guidelines is represented as a set of personas affected by the success criterion. During the first 16 months of the P4a project, a first prototype was created, containing an initial set of personas that can be browsed to illustrate different requirements. The plan is to publish this prototype in September 2015 on http://personabrowser.gpii.net.
1 Contribution to the global architecture

The technologies reported here are all part of the DeveloperSpace “building blocks” and “frameworks. As such, they are intended to make it easier for developer to create new and better solutions faster, and with less cost.

AsTeRICS

The development work moving the AsTeRICS Configuration Suite from Windows and the Microsoft .NET framework with Windows Presentation Foundation (WPF) to HTML5 provides three contributions to the DeveloperSpace. First it makes the AsTeRICS run time configurable from any platform rather than just Windows. It also allows it to be configured without requiring a person to download and install not only ACS but also the .NET framework on their computer. As we try to push development work out to the consumer and service-delivery professionals, the ability to work cross-platform and to not install special software on computers in order to configure solutions becomes more important.

The third reason for moving the AsTeRICS Configuration Suite to web technologies is that it provides a model and open-source code demonstrating how to create model based programming of assistive technologies for individual users. AsTeRICS provides a mechanism for visually combining functional units on screen that allows a person without programming skills to be able to create new assistive technology configurations. By moving this to web based technologies, and providing the source-code, it makes it much easier for other researchers and developers to explore the possibilities in consumer and professional configured solutions for harder to address users. It also opens up possibilities for engaging younger boys and girls in programming and assistive technology by providing tools that are more reachable and useable by them at an early age.

URC Socket Builder

The URC interconnection standard and its affiliated technologies provide a standard means for providing very specialized interfaces that can be used across products and devices. It also provides an interconnection mechanism between different interface functions and services.

The URC Socket Builder greatly decreases the effort needed by developers (mainstream or assistive technology) in creating URCs and URC enabled products. As such it will be an essential tool to support the GPII runtime [5].

The URC Socket Builder will also facilitate connection of the three technologies described here (AsTeRICS, MyUI and URC) by automated code generation. This will simplify the linkage of the three technologies for user interface developers.
**The Persona Browser**

The Persona Browser is being designed to support the RESOURCES section of the DeveloperSpace. Its intent is to make the requirements for accessible web design more understandable.

The Persona Browser uses the concept of personas to explain guidelines and design patterns to web developers in a none-abstract way. By using GPII preference sets users will be able to test the accessibility of web applications and quickly see the effects on different types of users.

**Adaptive User Interface Toolkit**

The goal of the adaptive user interface toolkit is to provide developers with an integrated facility to develop user interfaces that include comprehensive possibilities of adaptation to the individual needs and preferences of different users in different contexts using different devices following the one-size-fits-one approach. It uses the self-adaptive UI modules being created in T202.6 and combines these into an application with an adaptive user interface being able to be run by the runtime environment being built by T203.3. The efforts regarding adaptive user interfaces in both work packages, 202 and 203, together lead to a flexible development approach: from the usage of individual adaptive user interface components, over the traditional API-oriented implementation of adaptive user interfaces for the runtime environment to the model-driven creation of adaptive user interfaces using the development toolkit.
Figure 1: Overall Picture of Prosperity4all
2 Initial Prototype of WebACS

2.1 Concept

2.1.1 Initial Situation

The current AsTeRICS Configuration Suite (ACS) (as developed during the AsTeRICS project, Grant Agreement No.247730, and maintained on Github [1]) is based on Microsoft .NET technology using Windows Presentation Foundation (WPF) for the graphical representation. For the menu system, Microsoft Ribbons are used. Figure 2: AsTeRICS Configuration Suite 2.6 shows the ACS after startup.

Figure 2: AsTeRICS Configuration Suite 2.6

![AsTeRICS Configuration Suite 2.6](image)

The main parts of the ACS are the Deployment Editor (a graphical editor that allows the user to build visual models to realise assistive functionality using a set of about 160 plugins), the GUI Designer (also a graphical editor, allowing the user to arrange components’ UI elements) and the Property Editor, which allows the user to set plugins’ properties (Figure 3).
2.1.2 Goals and Design Targets

The main problem with the current ACS implementation is its dependence on Microsoft Windows and .NET. A more platform independent version would enhance uptake, by users and developers in general – and help us push development out to the user/service-professional level, creating more engagement and participation by users and those working with them (a goal of the DeveloperSpace).

An additional problem is that the current version of the ACS is built on a “grown” architecture, which often creates difficulties during maintenance and further development.

In the scope of the Prosterity4All project a rewrite of the ACS will be done (to the extent of the basic functionalities). The main goal for this rewrite is to make it platform independent by using web technologies. A side benefit is that it will be more serviceable due to a better architecture behind the implementation.

2.2 Architecture

The WebACS’ architecture builds on a Model-View approach, thus making the view easily interchangeable. Also some of the usual design patterns in software development, like Command or Observer, have been followed.

A class diagram of the WebACS’ architecture is kept up-to-date during implementation (using the Dia diagram Editor[2]) and can be downloaded from Github[3].
The WebACS is implemented in JavaScript. In order to avoid global variables and to make encapsulation of data and functions possible, each “class”, or more precisely, each function of the WebACS implements the Module pattern[4].

2.3 Implementation

The HTML5 and JavaScript based implementation of the WebACS is maintained on Github [3], where it can be downloaded and tested. To start the WebACS simply open the file “WebACS.html” in Firefox (35.0.1 or higher).

Here is a list of some of the functions that are already implemented:

- New Model
- Open Model
- Close Model
- Save Model
- Add or delete components
- Add or delete datachannels
- Add or delete eventchannels
- Cut
- Copy
- Paste
- Undo
- Redo

Figure 4: WebACS with two models open
One thing that is new in this version of the ACS is the possibility of opening several AsTeRICS-models in parallel (see Figure 4), thus being able to copy/paste from one model to another. This makes model development significantly easier, for now it is possible to re-use blocks of components without having to set all properties from scratch.

2.4 Future Work

The most important next steps will be

- the connection to the AsTeRICS Runtime Environment (ARE)
- the Property Editor and
- a help-system for the ACS.

The connection to the ARE will be established through the REST API that has been provided by partner UCY.

The help system will be HTML-based and will aim to reuse the current ACS’ help files with as little modification as possible, thus saving effort. Every component in the WebACS shall have its own context sensitive help file that can be accessed by simply pressing F1.

In general the developers will try to include as much functionality as possible in the scope of the Prosperity4All project, focusing primarily (as noted in the DOW) on the most basic functionality of the current ACS.
3 URC Socket Builder

3.1 Concept

The URC Socket Builder is a tool supporting developers working with the URC framework [5]. The latter plays a central role in the GPII runtime architecture developed in T203.3 of the P4A project [6]. Consequently, appropriate tool support for developing with the URC framework in context of the GPII infrastructure is indispensable. The URC Socket Builder and its extensions developed in the context of the P4A project will provide means to connect other user interface technologies to the URC framework. Typical examples would be to connect the AsTeRiCS framework [7] and MyUI [8] that are both core elements of the GPII runtime [9].

Still, it is necessary to get a brief understanding of the URC framework.

The URC framework is standardised in ISO/IEC 24752 [10] and maintained by the openURC Alliance [11]. Its goal is that every user can control any target with a user interface fitting best his or her needs. User interfaces can range from various software interfaces to specialised hardware devices. Targets can be all kind of electronic devices or services. In order to provide the functionality for exchangeable user interfaces – also called "pluggable user interfaces" – every target must provide an abstract description of its operative user interface. In URC terms these abstract descriptions are called User Interface Socket Descriptions – or just Socket Descriptions. Socket Descriptions are XML files and contain information about a target’s variables and how they can be accessed by a remote user interface, which kind of commands can be sent to a target and finally which kind of notifications a target can send to a connected controller.

For every variable, command and notification, additional resources like labels and help texts in different languages or pictures can be defined that can later be rendered on a controller device according to the user’s needs [12].

The URC Socket Builder is a tool simplifying the creation of sockets and additional resources. Instead of directly editing the relevant xml files socket developers can take advantage of a graphical user interface (GUI). Relevant socket elements can be created and edited through a GUI. Then, the URC Socket Builder generates the appropriate XML code and links the relevant files to each other.

ISO 24752 compliant targets provide the required sockets on their own. Non-compliant targets can still be integrated via a middleware solution. The latter is the so-called Universal Control Hub (UCH) that is also developed by the openURC Alliance [13]. Later, together with the AsTeRiCS framework and MyUI the UCH will form the core of the GPII runtime developed in T203.3 [9] of the P4a project.
The third part of the URC framework is the resource server. Sockets, resources and user interface implementations for a certain socket can be stored on this server so that they are globally available.

3.1.1 Initial Situation

At the beginning of the P4A project, an Alpha version of the URC Socket Builder was available. Until then, the software had been developed with funds from the Trace R&D Centre’s IT RERC program. With the beginning of the P4A project, the functionality of the available version was tested by members of the Stuttgart Media University.

At this point in time the main window was available, comprising elements for executing the following functionality:

- Creation of targets and automated generation of corresponding XML files,
- Creation of sockets and automated generation of corresponding XML files,
- Creation of Socket elements like variables, commands and notifications,
- Creation of relating resources and automated generation of corresponding XML files,
- Creation of self-defined XML data types.

Although this functionality was available, practical tests in the course of the P4A project showed several shortcomings. Most problems were related to:

- **Stability:**
  In tests the Socket Builder was not yet running stable, especially for users relying on keyboard navigation.

- **Accessibility problems for keyboard users:**
  Not all areas of the GUI could be reached with pure keyboard navigation.

- **Accessibility problems for screenreaders:**
  Only the main window could be accessed with screenreaders while in the dialogue boxes alternative texts and labels were not accessible for screenreaders.

- **Local deployment and testing:**
  There was no functionality available for deploying the created files to a local UCH installation. Without that, the testing of the created sockets and their usage for further user interface development was very difficult.

- **Uploads and synchronisation with global resource server:**
  The created files had to be uploaded by hand to the resource server. Furthermore, when starting the development of a socket, there was no way to check if a similar socket was already available on the resource server.
3.1.2 Goals and Design Targets

Goals and design targets are influenced by three major thoughts. First of all, it is necessary to overcome the shortcomings discovered during the first tests. Next, not only the final GPII technology should be accessible for all users, but also the development tools themselves. Finally, a way for connecting the URC framework with the other technologies involved in the GPII runtime is required.

These thoughts lead to the following goals and design targets:

(1) General functionality:
   a. Improving overall system stability,
   b. Extending the URC Socket Builder for functions to enable deployment to a local UCH instance,
   c. Configuration of a local UCH instance to test the deployed sockets with appropriate script files,
   d. Providing functionality to upload created Sockets to the resource server and synchronization of the new sockets and resources with existing ones.

(2) Accessibility improvements:
   a. Providing all dialogues and other GUI elements with appropriate labels and alternative texts so that different Assistive Technologies (e.g., screenreaders) can benefit from these meta data and additional information,
   b. Making the URC Socket Builder navigable for users relaying on pure keyboard navigation. This means all GUI elements must be reachable with the keyboard, as well as appropriate visualisation of the focused element.

(3) Integration with other GPII technologies
   a. Implementation of new functionality. Basically this means that the URC Socket Builder provides code snippets that can be used to create GPII-enabled user interfaces that connect to a given socket element.

3.2 Current Status

During the first 16 months of the P4A project the following steps were taken:

- Testing and local deployment of created sockets:
  The implementation of testing and deployment features in conjunction with a local UCH installation was successfully completed. This required updates on two sides – on the side of the URC Socket Builder as well as on the side of a locally installed UCH. The Socket Builder’s functionality, as well as its menu structure was extended so that now all created files can be transferred from the Socket Builder to the resource folder of a locally installed UCH. Additionally, a stump of a simulation script file is created and deployed to the appropriate folder of the UCH. Furthermore, the URC
Socket Builder is now able to edit the appropriate config file on the side of the UCH so that the created script file is used for simulating the different values of the created socket and its elements.

As already mentioned, there were also some changes on the side of the UCH necessary. To be more precise one of its components – the so called Generic Target Adapter - had to be updated.

The configuration mechanism was updated so that all configuration settings can easily be done by the Socket Builder.

- **Providing GUI elements with Meta data:**
  Where necessary, the source code was updated so that all GUI elements are now provided with meta data such as labels and alternative texts. This improves overall accessibility for assistive technologies such as accessibility via screenreaders. Most updates were related to dialogue boxes.

- **Improvement of keyboard navigation:**
  Progress was made on the improvement of the keyboard navigation. All GUI elements can now be reached with pure keyboard navigation. Furthermore, improvements were made by optimising the setting of the focus. This means shorter ways for keyboard users and a more logical user concept for screenreader users.

- **Improvement of system stability and bug fixing:**
  In order to improve the overall system stability, other improvements and bug fixes were made.

These improvements lead to a new Socket Builder version that is now available at [11]. Version 1.1 was stable and functional enough that it could be tested by students in a course at the Stuttgart Media University. The students’ task was to create some simple sockets. This test with the students was successful. Furthermore, the students gave qualitative, valuable feedback about the current Socket Builder version which is now being incorporated in the tool.

The Socket Builder will be used to build some of the URC Master Socket Templates which are part of T202.4 of the P4a project [9].
3.3 Future work

Future versions of the Socket Builder will be able to generate code snippets that can be used to integrate other GPII applications with URC Sockets. This is a totally new functionality that needs to be implemented. The code snippets will include logic to execute the URC-HTTP 2.0 protocol [12] that is used to remotely control a connected UCH.

Furthermore, future versions of the URC socket builder will provide functionality to synchronize and upload new sockets and resources to the resource server. This will improve the global availability of the created items for users as well as for user interface developers.
4  Persona Browser

4.1  Concept

The underlying principle of our approach is to use personas as representation and illustration of accessibility guidelines (e.g., WCAG) for the developer. Each success criterion of the accessibility guideline is represented as a set of personas affected by the success criterion.

Solutions in HTML

The PersonaBrowser will both provide practical information in form of HTML examples on how to meet the individual accessibility guidelines and success criteria and provide a means for them to see the effect of these guidelines for example users. This will provide a web author with hands-on guidance on how to design and implement accessible web applications.

Web authors (e.g., software-developers and web-designers) have a hard time understanding and applying guidelines for accessible web applications [21][22][18].

Existing guidelines are considered as too technical and it’s not commonly known how people with disabilities interact with web applications, what barriers they face and what the implications of assistive technology usage on the people’s interaction paradigm is.

To achieve accessibility, one must not only guarantee guideline conformance, but also the usability and understandability of a web application; hence, knowing the target’s audience is the core concept for providing accessible and usable web applications in the terms of human-centered design.

Therefore, web authors must be aware of particular users’ traits, specific interaction patterns (e.g., due to dedicated usage of assistive technologies) and how to design web applications meeting the users requirements. All this information is scattered in books and web sources, and is not easy to grasp.

The PersonaBrowser is a new innovative platform where all this information is presented and interconnected and will serve as an information platform for web authors. The PersonaBrowser will be available as a web application and will follow the principle of accessible design. A novelty of our platform is that we provide the information wrapped around personas and told by illustrated stories.

Personas are widely known in the school of usability design. One prominent example is the concept of personas [20]. Personas are illustrations of hypothetical users. They are composed by an aggregation of stories, requirements, needs, goals and preferences of the product’s end users. Personas are in the center of the PersonaBrowser and are used to
illustrate accessibility guidelines, barrier descriptions and connected disabilities and consequences for the interaction with applications. Several studies have revealed that personas are an appropriate way to convey accessibility requirements, which results in sustainable accessible applications [19][23][17][24]. The underlying principle of our approach is to use personas as representation and illustration of accessibility guidelines (e.g., WCAG) for the developer. Each success criterion of the guideline is represented as a set of personas affected by the success criterion. Figure 6 illustrates the content of the PersonaBrowser.

Figure 6: Illustration of the content of the PersonaBrowser

4.1.1 Initial Situation

At the beginning of P4a only conceptual work was available but no implementation effort was made.

The foundations of the PersonaBrowser are personas as a representation and illustration of accessibility guidelines for the web authors, e.g., WCAG 2.0 [25]. Each success criterion of the guideline is represented as a set of personas affected (possibly in varying degrees) by the success criterion. A persona also contains information about how they would interact with a web application, together with a description of assistive technologies that the persona would possibly use and typical barriers that this persona is often confronted with.

4.1.2 Goals and Design Targets

The main goal is to transfer the conceptual work into a real browser based implementation.

The final implementation of the PersonaBrowser will provide practical information in form of HTML examples on how to meet the individual accessibility guidelines and success criteria. This will provide a web author with hands-on guidance on how to design and implement Ecosystem infrastructure for smart and personalised inclusion and PROSPERITY for ALL stakeholders www.prosperity4all.eu
accessible web applications.

Major stepping-stones are:

- Creation of a browser based tool,
- providing functionality for creating new personas and
- providing functionality for editing existing personas.

Open Platform

The PersonaBrowser is designed as an open platform where any accessibility expert and web author can share their knowledge. It will be possible for example that web authors can present multiple solutions on how to meet certain guidelines and success criteria depending on specific use cases, platforms (desktop or mobile web) or different web technologies and frameworks. In the same way people will be able to share stories, personas or information about assistive technologies (and also any other information) by contributing to the PersonaBrowser. Therefore, we will have an editorial backend where interested users can register and enter new information or alter existing information.

Search and filter Information

As a web author one will be able to search for information in any direction. This means one can start by looking up the personas and reading their stories, then following the links to their assistive technology and read about typical barriers. Web authors can browse the guidelines and success criteria and read more about the people that are affected by them by following links.

It will also be possible to filter the PersonaBrowser for specific information; hence, it will be possible the look up HTML solutions for people, using a specific type of assistive technology. Or to learn more about the requirements and issues of dedicated a target audience, e.g., by filtering the PersonaBrowser for typical barriers of people with mild cognitive impairments only.

4.2 Report on first prototype

During the first 16 Month of the P4a project a first prototype was created. It will soon be available at http://personabrowser.gpii.net. The prototype ships with an initial set of personas that can be browsed to illustrate different requirements.

The PersonaBrowser is currently undergoing further development. The following screenshots show the PersonaBrowser in its current version.
Note:
The screenshots are taken from an early version and the content is available in German only. The final version of the PersonaBrowser however will have the possibility to switch languages and will also be available in English.

Figure 7: Screenshot showing the information contained in the PersonaBrowser
4.3 Future work

The PersonaBrowser is currently under development and will be available at http://personabrowser.gpii.net. It is planned that a first version will be going live by the end of September 2015.

It is intended that in future versions of the PersonaBrowser there will also be GPII preference sets for the Personas available as information. Like any information and content in the PersonaBrowser, this preference set information can also be altered and changed by contributors of the PersonaBrowser. It is planned to conduct accessibility tests based on this GPII preference set information. Therefore a web author can learn about the Personas and their disabilities, and he can also use the personas to conduct accessibility tests for his current web application project.

Figure 8: Detail view: Showing all relevant WCAG 2.0 success criteria for hearing impaired people.
5 Adaptive User Interface Development Toolkit

5.1 Concept

5.1.1 Initial Situation

Along with the adaptive user interface system itself the MyUI project has also created an adaptive user interface development toolkit based on Eclipse [29]. Key feature of the development toolkit is the graphical editor for the Abstract Application Interaction Model (AAIM) and the generator creating code for the CakePHP-based runtime implementation. Changes in the source code are parsed by the generator in reverse to update the AAIM. This way the model and the code are kept synchronised as shown in Figure 9.

In the AAIM is a model for specifying the interaction possibilities and interaction flow in a notation based on statecharts [27]. Instead of defining concrete user interfaces using a set of user interface components the developer defines which interaction possibilities are available in a state (called an interaction situation). Based on that model the MyUI runtime selects the concrete realisation (called an interaction pattern) to be presented to the user that fits best to the user profile and device capabilities.

Figure 9: MyUI Development Toolkit with generated PHP Code and AAIM Model Editor [28]
The MyUI model editor makes heavy use of context menus to add and manipulate AAIM elements and uses dialogs for data entry. Interaction situations can be assigned to model states by drag & drop from a library but transitions have to be added via the context menu of the source state.

To give the developer an impression of what the resulting user interfaces look like the MyUI development toolkit provides a preview function following the “What You See Is What Others Get” (WYSIWOG) principle. The developer can run the created AAIM and directly manipulate the user profile variables to see the individual resulting user interface.

The preview function requires a full and running installation of the MyUI infrastructure including a webserver and the context manager. The preview can only be run if the AAIM is complete including data acquisition functions supplying data. While data acquisition functions and application functions in concept decouple the user interface from the application’s logic the creation of the user interface is coupled to the availability of these functions or corresponding mocks.

Figure 10: MyUI Preview Browser and profile adjustment sliders [28]

5.1.2 Goals and Design Targets

When developing adaptive user interfaces using the MyUI approach the AAIM is the major artefact the developer gets in touch with. Because of that the way of creating, checking and changing the AAIM and the tools supporting the developer to do so are in the primary focus of the design improvements performed in Prosperity4All regarding the development toolkit for adaptive user interfaces.
• **Provide more direct feedback on resulting user interfaces**

The MyUI development toolkit provides a mechanism of running a preview of an AAIM to give the developer an idea of the resulting user interface. This results in the need of frequent context switches for the developer between modelling and preview. Hence, the feedback regarding the resulting user interface is delayed during the development process. Additionally when using the preview the developer has to navigate from the initial state to the point in the interaction flow currently under work.

To provide immediate feedback while modelling in P4A the modeller will use a “Model by Example” approach. The states of the AAIM are represented by schematic illustrations of the included interaction situations. To give the developer an impression of the variety of resulting user interfaces these illustrations will be based on the concrete interaction pattern based on the combination of selected user and device profiles.

Referring to the requirements for adaptive user interfaces presented by Peissner et al. [29] this improves the *transparency of the adaptation process*. Especially the connection between the AAIM and the variety of generated user interfaces is emphasised.

• **Improve the overall usability of the model editor**

The MyUI AAIM editor has been implemented from scratch using basic Java technologies. In meantime sophisticated modelling frameworks like the Graphical Modeling Framework based on Eclipse (see [32]) have made steady advance to support the implementation of domain-specific model editors providing rich editing possibilities.

For the AAIM model editor that will be implemented in P4A these rich editing possibilities incorporate the usage of inline editors for textual expressions (e.g. the event/guard expressions for transitions) including code completion in the graphical editor. In addition it will be possible to assign interaction situations to states by drag & drop directly from the pattern browser or by typing in their name supported by a typeahead function. The possibility to copy & paste single states of whole parts of an AAIM into another one is another essential functionality. Along with that, basic automatic layouts to apply to an AAIM diagram and layout assistants like the alignment of boxes to each other will help developers to create diagrams with a clean look.

On the one side these improvements will increase the *efficiency in the development process* by speeding up the key task of modelling the AAIM (see [29]). On the other side especially the inline editors with code completion will make it *easier to learn* and understand the concepts and relations of the AAIM.
• Upgrade the AAIM to be used as a communication asset in development

In the MyUI approach the AAIM is the central application specific artefact created by the developer which defines possible interaction flows and assembles these with the application’s data and functions. One of the principles of agile software development is that “business people and developers must work together daily throughout the project” [31]. Obviously the creation of additional documentation for that purpose would contradict the agile principles.

Against this backdrop, the goal in P4A is to make the AAIM an asset intuitively representing the interaction of an application to enable stakeholders like product managers, developers, designers and so forth to gain a common understanding.

Through supporting fast iterations, enabling communication and preventing the creation of redundant documentation this goal supports the requirement *efficiency in the development process* (see [29]). To reach the goal of enabling communication it has to be *easy to learn* (or at least easy to understand) for people of many different disciplines.

• Support main stream software development practises

Currently, the MyUI development toolkit represents the AAIM in the graphical statechart editor and through the generated PHP source code. Since this source code has to conform to the CakePHP conventions it cannot be considered a comprehensible and concise textual representation of the AAIM. Moreover, it is possible to write valid CakePHP code that cannot be translated into an AAIM.

The goal is to provide a textual representation of the AAIM which is human-readable and writeable together with a corresponding editor. This would add a textual domain-specific language (DSL, see [30]) to the existing graphical modelling language. Using this as a persistence format will enable common software engineering practices like version control like computing diffs or automatic merging. Along with that, although not in the focus right now, a textual 1-to-1 representation of the AAIM opens new possibilities to make the development toolkit itself more accessible.

According to the requirement of *efficiency in the development process* (see [29]) the textual DSL provides a concise and clear textual representation as alternative to the graphical model. In addition, this enables the usage of the AAIM in main stream practises like version control and comparison using textual diffs.

• Independence of server-side technology

In P4A the runtime environment for adaptive user interfaces is to be reimplemented using modern client-side web technologies like the web components approach. Accordingly, the development toolkit should also be independent of the used server-
side technology. It should neither need to generate server-side code nor depend on specific server-side technology to provide the preview functionality.

This goal contributes to the requirement of modularity mentioned in [29] by separating the concerns of client- and server-side technology selection. It also leads to the extensibility of not only the user interface solutions provided by the system but also in terms of the supported technical environments for the creation of applications with adaptive user interfaces.

5.2 Architecture and Implementation

5.2.1 Development Toolkit User Interface

5.2.1.1 Graphical AAIM Editor

The graphical editor for the creation of AAIMs comprises two views: First, the AAIM Overview showing the complete AAIM with all states and transitions similar to the existing MyUI AAIM editor shown in Figure 9. Second, a details view of a single state of the AAIM providing a more detailed representation of the resulting screen layout.

The overview shows the contained states and possible transitions in the form of a screenflow or storyboard of the modelled application as shown in Figure 11. Following the “Model by Example” approach the states are represented by wireframes previewing the resulting interface for a certain set of user preferences and device properties. In addition to the modelling canvas the toolkit provides a two-part palette containing the tools for creating states and transitions and the available interaction situations.

The displayed wireframe for each modelled state as well as the icons of interaction situations in the palette change according to the concrete interaction pattern suitable for the set of user preferences and device properties currently selected by the developer. When the developer changes the user preferences by choosing another persona of fine-tuning individual preferences of chooses another device profile the wireframes on the modelling canvas as well as the palette icons are adapted accordingly. This provides a compact overview over the variety of concrete user interfaces generated under different circumstances.

In the overview the developer may select one state/screen to get to the corresponding details view. In the details view only one individual state of the AAIM will be represented by its wireframe. In difference to the very rough representation in the overview the wireframe of the details view for example also contains mock ups of sample content in addition to boxes representing screen elements (see Figure 12).
In the details view in addition to the edit mode a simulation mode is available. When activated the developer is able to interact with the detailed wireframes of the contained interaction situations to activate outgoing transitions. In consequence the details view for the target state of the activated transition will be shown. This way the developer may navigate through the application state-by-state/screen-by-screen and gain an impression how the application’s user interface would look like for a user.
In the graphical editor the developer has multiple possibilities to assign interaction situations to states of the AAIM as shown in Figure 13. First, the developer may drag & drop an interaction situation from to tools palette into the details view of the state to get a more detailed impression of the resulting user interface. Second, the option to drag & drop is also available in the overview to get a fast impression of the relations between states, their interaction situations and the corresponding transitions. Third, instead of using the mouse-oriented drag & drop the developer may just type the name of an interaction situation into an area of the state assisted by the typeahead function suggesting available interaction situations. We expect that developers will use these editing techniques in the presented order as their expertise with the creation of adaptive user interfaces and the available interaction situations rises.

Figure 13: Different possibilities to assign interaction situations
a) per drag & drop in the details view, b) per drag & drop in the overview and c) per typeahead in the overview

There still are several occurrences of textual expressions in the graphical representation of the AAIM. For example the developer has to enter an expression for each transition specifying the condition under which the transition will be performed and its target state will be displayed. Whenever a textual expression needs to be supplied by the developer the model editor will assist with a typeahead feature suggesting only valid options as shown in Figure 14. This supports the developer to learn the formal relations of the AAIM and prevents the creation of invalid models.
Many of the graphical editing features described above are implemented in the Eclipse-based Yakindu Statechart Tools which is as well “open and designed for extensibility, so it can serve as a basis for more specific statechart modeling” [33]. After evaluation of the “Model by Example” concept with developers (see section 5.3) it has to be validated if the feature of representing states by generated wireframes can be build based on Yakindu.

5.2.1.2 Textual AAIM Editor

The textual editor provides developers a second, more technical and programming alike access to AAIMs. While offering a representation of the AAIM based on plain text it is backed by the same AAIM metamodel as the graphical editor. This metamodel serves as a common point for all AAIM editors and makes it possible to keep all representations of an AAIM synchronised on changes.

The concrete syntax of the textual AAIM DSL uses syntactical elements (i.e. curly brackets for command blocks and semicolon for the separation of commands) common to programming languages used in the area of web applications. The syntax of AAIM specific expressions is aligned with the textual expressions contained in the graphical model.

To support the developer while creating an AAIM using the textual DSL the editor provides features that can usually be found in modern programming language editors. This includes customisable syntax highlighting to emphasise the structure of the source code, validation markers to indicate logical or syntactical mistakes and typeahead functionality to suggest possible completions of the typed code.

The prototype of the editor for the textual representation of the AAIM uses the Eclipse Xtext framework [34], which provides an infrastructure to create feature rich editors for textual domain specific languages. The Xtext team is currently working on the integration of Xtext into other development environments like IntelliJ IDEA or even web applications. This way at least the textual creation and manipulation of AAIMs will be possible in other environments than the Eclipse IDE.
5.2.2 Relation to the Runtime Environment

The relation between the adaptive user interface development toolkit and the runtime environment for adaptive user interfaces can be divided into the directions of information flow.

Runtime Environment to Development Toolkit

In one direction the development toolkit requires information of the runtime environment on available interaction situations. Key requirement for this information is the availability of abstract descriptions of the interaction patterns and user interface elements. Additionally abstract descriptions of the device specific patterns defining the structure and areas of the screens are necessary to enable the development environment to render the state
wireframes as well as the palette icons. For the first prototypes this information is implemented into the development toolkit manually having in mind to replace it as the work on the runtime environment proceeds.

**Development Toolkit to Runtime Environment**

In the other direction the development environment has to deploy a representation of the AAIM the runtime environment is able to execute. Generally there are two possibilities: Generate code in the general purpose programming language the runtime environment uses or let the runtime environment interpret the model at runtime.

In the first case the development environment has to contain a code generator that translates the AAIM created by the developer into JavaScript code using the application programming interface (API) provided by the runtime environment. A mechanism has to be established that ensures the separation of generated code and code manually written by developers.

In the second case the textual DSL may be used as the artefact representing the AAIM to be interpreted by the runtime environment. To achieve the goal of being independent of the technology used on the server-side the interpreter needs to completely run in the browser and being implemented in JavaScript.

Since there obviously are strong dependencies to the architecture of the runtime environment the decision on this concern has to be made in conjunction the technical conception of the runtime environment.

**5.3 Future Work**

Before starting the final technical implementation of the development toolkit the concepts described above need to be evaluated with developers. In this evaluation the following key questions will be addressed:

- Does the development toolkit provide an impression of the resulting user interfaces and their potential variety?
- Is the graphical representation of the AAIM suitable to be used in communication between business people and developers?
- Are there software development practices the model-driven approach of the development toolkit may be in conflict with?
- Which way of AAIM editing (textual, graphical overview, graphical detail) is the preferred one?
- Are there necessary or possible optimisations of the AAIM editors regarding their usability?
As a summary the test participants will be asked to provide an overall rating of the presented development toolkit regarding the requirements transparency of the adaptation process, efficiency in the development process, easy to learn, modularity and extensibility presented in [29].
6 References

[1] https://github.com/asterics/AsTeRICS
[3] https://github.com/asterics/WebACS/tree/39659ce49ca8e861560ee79a946245e2d7c4f4db


[20] Cooper A. (1999), „The Inmates are Running the Asylum.“ SAMS/Mac-millan, Indiana, USA, 1999


Ecosystem infrastructure for smart and personalised inclusion and PROSPERITY for ALL stakeholders

www.prosperityforall.eu


[33] Yakindu Statechart Tools, URL: http://statecharts.org/

[34] Eclipse Xtext Framework, URL: https://eclipse.org/Xtext/