A Runtime Middleware for Enabling Application Integration and Rapid Re-Engineering

Marios Komodromos[1], Christos Mettouris[1], Achilleas P. Achilleos[1], George A. Papadopoulos[1], Martin Deinhofer[2], Christoph Veigl[2], Alfred Doppler[3] and Stefan Schürz[3]

[1] Department of Computer Science, University of Cyprus, Nicosia, Cyprus
{mkomod05, mettour, achilleas, george}@cs.ucy.ac.cy

[2]Institute of Embedded Systems, University of Applied Sciences (UAS) Technikum Wien Vienna, Austria
{veigl, martin.deinhofer}@technikum-wien.at

[3]LIFEtool gemeinnützige GmbH, Linz, Austria
{alfred.doppler, stefan.schuerz}@lifetool.at

Abstract. Assistive Technologies devices and systems aim to improve the quality of life of people with disabilities by providing a wide range of assistive services. The AsTeRICS framework provides a Runtime Environment and a toolset that can be highly adapted to the changing needs of each individual and targets to reduce time, effort and costs of developing assistive applications. A key limitation of the platform, as well as similar platforms, is that the integration of accessibility features into existing software applications is difficult due to heterogeneous implementation technologies. Moreover, the capabilities offered by sensors, actuators and other mobile devices deployed on different machines cannot be exploited for the development of AT applications. This paper presents substantial improvements and changes to the Runtime Environment that address these issues and offer the capability to integrate applications built with different technologies and thus to improve the accessibility of an application. In fact, the middleware environment enables to rapidly re-engineer existing software applications. Two examples are re-developed in this work, the "FlashWords" and "EURO" applications, which are extended in a way so that they can also be used by children with motor disabilities.

1 Introduction

A significant number of people with disabilities worldwide are supported by Assistive Technologies (AT) [1, 2]. Available AT devices and systems provide a wide range of assistive functionality, improving thus the quality of life of people with disabilities. However, AT devices often require adaptation, since they have been designed for explicit applications, and thus cannot be used in slightly different environments without serious customizations. In this respect, routine activities of people with disabilities may be restricted, either because AT devices cannot be adapted based on their needs, or because the device itself or its adaptations introduce unaffordable costs.

The AsTeRICS (Assistive Technology Rapid Integration & Construction Set) project [3] has built a hardware and software framework, which targets 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 and 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 and mobile services that did not offer highly specialised user interfaces until present. By using its scalable and extensible architecture, it provides easy means for designing and running specialized AT applications, which can simplify and assist the daily routine of people with disabilities.

A key limitation of the platform, as well as comparable platforms, is that they do not permit reuse and quick integration of the functions offered by its assistive components to an existing application developed using another technology. In specific, the work presented in this paper simplifies and expedites integration of assistive functionalities in existing software applications built using different technologies, as well as improving the accessibility of the application. Another new feature offered by the refined runtime middleware environment is the capability to exploit the capabilities offered by sensors, actuators and other mobile devices deployed on different machines for the development of assistive applications. The runtime environment is re-designed and developed as a Java-OSGi middleware, which offers assistive functionalities via the pool of existing OSGi components (sensors, actuators, etc.), which are exposed now as REST services.

This paper presents the runtime environment and validates it through two use cases for the rapid re-development of two software applications, namely FlashWords and EURO. This is performed via the rapid integration of additional assistive functionalities, transparently via the developed REST-enabled runtime environment, into these software applications. FlashWords is an application that targets persons who can hardly speak or have insufficient speech abilities. According to their condition, children with Down Syndrome have special problems with their speech development (hearing, speech motor function, short auditory memory, comprehension). FlashWords is based on a concept described in the series Small Steps, called Early Reading that allows helping children with auditory problems. The environment enables to rapidly re-implement the FlashWords application into an application that can be used also by children that have motor disabilities. EURO on the other hand is an application that assists children and adults with learning disabilities to get to know money. Also in this scenario, the runtime environment enables to rapidly re-implement the EURO application, so that it can be used also by people who are not able to control their hands or feet with precision in order to use ordinary physical buttons.

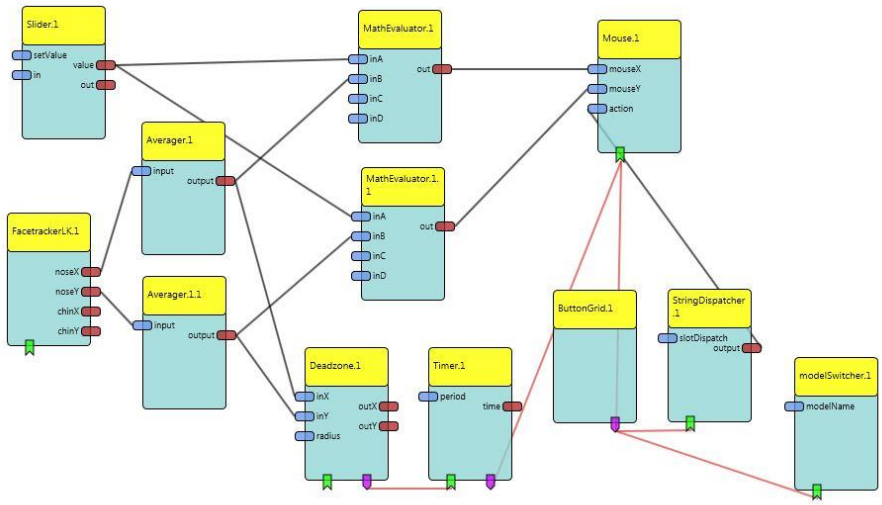

Fig. 1. AsTeRICS AT model designed via the ACS graphical tool.

The rest of the paper is structured as follows. Section 2 presents related work, while Section 3 provides a technical description of the architecture and implementation of the runtime middleware environment. In Sections 4 and 5 the FlashWords and EURO application scenarios are presented and the demonstrators of the re-developed applications are presented. Finally, the paper presents conclusions and future work in Section 6.

2 Related Work

A number of AT systems have been developed, mainly in European Projects. The TOBI project [4] focuses on the design of non-invasive BNCI 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 [5] 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 at improving interaction of the user with people, home appliances, assistive devices, personal computers, internet technologies, and more. BrainAble's [6] main objective is to assist people with disabilities on overcoming exclusion from home and social activities by providing an ICT-based Human Computer Interface (HCI), as well as producing a set of technologies suitable for assisting people with physical disabilities regardless of cause.

A project with many similarities with AsTeRICS both in terms of the concept, the implementation and the system architecture is OpenHAB. It provides a scalable and

modular architecture that integrates components and technologies in a single solution. OpenHAB is open-source with an active community, which enables new features and functionalities to be added, as with AsTeRICS. It is also based on JAVA OSGi [7] and provides APIs for integration with other systems. In addition, it provides remote communication with a REST API and intra communication. The "new thing" that OpenHAB introduces is that it gives the ability to the user to define the interaction of things and devices. 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 amongst the components even for a simple AT scenario in the Smart Home. In contrast, the AsTeRICS system enables a non-expert AT designer to use a simple modelling interface to easily model or reuse existing models to provide the necessary functionality to the user.

The contribution of this work is related to the substantial improvements and changes identified and performed to the Runtime Environment of the AsTeRICS framework. These improvements were performed so as to offer the capability to integrate applications built with different technologies and thus to improve the accessibility of applications. In specific, via the REST-enabled runtime environment, existing applications can be transformed into assistive applications enable people with motor disabilities to use them.

3 Implementation

3.1 The AsTeRICS Architecture

In AsTeRICS, a model is considered as the container that holds the information to describe the orchestration of the components that will produce a specific solution. The components of each model are 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 then 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 AsTeRICS Configuration Suite (ACS), a graphical tool for creating AT models (see Figure 1), and the AsTeRICS Runtime Environment (ARE), which is responsible for the deployment and runtime execution of the models. In the early stages, the two of them co-existed necessarily on the same machine and the communication was accomplished by exploiting the AsTeRICS Application Programming Interface [8] (ASAPI) protocol. In principle, ASAPI is a service that is provided by the ARE and can be consumed by different clients, such as the ACS, allowing them to control the runtime environment according to their needs.

The ARE is a Java OSGi-based [7] middleware. The OSGi technology provides component-based modularity and parallel execution for an application. Bounded by the OSGi principles, everything that lives in the application has to be defined as an OSGi bundle (i.e., self-contained component). Hence, each model component (i.e.,

sensor, processor, and actuator) must have an OSGi bundle instance inside the ARE. The AsTeRICS framework offers a models@runtime approach, since the ACS communicates with the ARE to deploy the models and handle the models via the ARE at runtime. ARE 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.

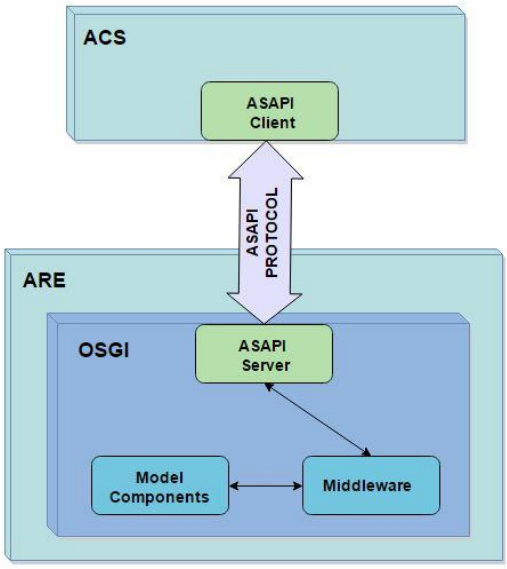

Fig. 2. The original AsTeRICS architecture.

3.2 Enabling Multi-Platform Accessibility

A serious architectural limitation was that the AsTeRICS Runtime Environment integration method (ASAPI) was primary and mainly designed for communication with ACS. It is a not widely known technology and furthermore it is a proprietary protocol, almost prohibitive for a third party client to implement. It naturally prevents any exploitation of the AsTeRICS Runtime Environment's technical qualities. Taking into account the above, it was essential to substitute the ASAPI protocol with an API that allows integration with minimal effort, as well as communication over the network.

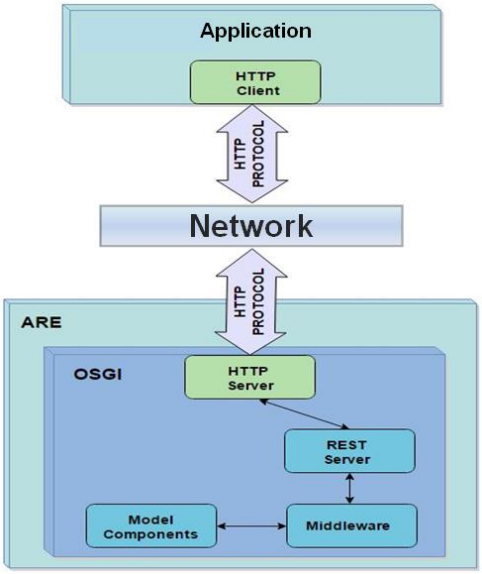

Fig. 3. The REST-enabled AsTeRICS architecture.

Due to the aforementioned reasons, Representational State Transfer (REST) API for the AsTeRICS Runtime Environment was defined and implemented [9]. REST is an architectural style that builds on the benefits of the Internet and the World Wide Web, such as scalability, remote communication, and easy access from everywhere and from any application. It is important to note here that the ASAPI protocol can also support remote communication, but since it's a proprietary protocol and difficult to implement, the widely-used REST architecture that is based on the well-known HTTP protocol is selected to perform the necessary architectural improvements to the runtime environment. The REST services were defined while respecting and utilising the OSGi implementation of the existing components. Therefore, the HTTP server where the REST services are deployed, was defined and implemented itself as an OSGi bundle/component. Exploiting the Grizzly NIO framework [10], an OSGi bundle was implemented that allows hosting an embedded HTTP server, through which the REST API can be accessed. Figure 3 presents the refined architecture of the runtime environment.

The main requirement for refining the runtime environment architecture was to provide the capability to integrate the very large set of assistive functionalities offered by the implemented components into existing applications. In fact, the target was to still facilitate the design and development of assistive applications through the use of the ACS, but at the same time enable integration of assistive functionalities into existing software applications implemented in different technologies. In specific, the ASAPI substitution offers platform and language independence. This means that a developer is able to reuse and integrate assistive functions into existing software applications, without any concerns about the language and/or the platform used to implement and deploy the applications. Finally, the development effort is reduced and

the development process is simplified since these assistive functionalities are implemented, reducing also the costs of integration.

4 The FlashWords Use Case

For the first scenario we have used an application called FlashWords developed by LIFEtool, which mainly targets at children with or without Down Syndrome who have insufficient speech abilities or special problems with their speech development (hearing, speech motor function, short auditory memory, comprehension). The application is based on an internationally acknowledged method called Early Reading and was developed by the special education centre at the Macquarie University in Sydney. The main idea is that the visualisation of words (see Figure 4) can help to compensate auditory problems, as the visual memory is available as working memory at an early stage and can support the auditory memory. This application is available in the Apple Store. [11]

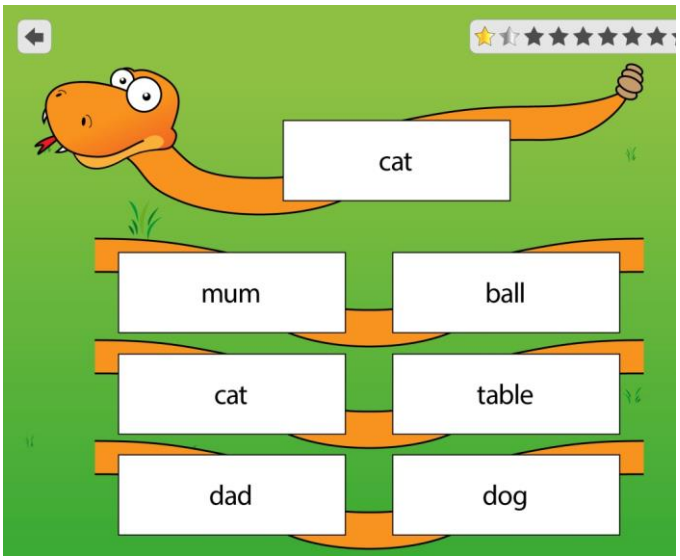

Fig. 4. Screenshot of the FlashWords App.

Through the AsTeRICS runtime environment REST API it was possible and easy to integrate the FlashWords application with the assistive functionalities offered by the components of the AsTeRICS platform, in order to achieve improvement of the accessibility of the FlashWords application. Although AsTeRICS features a MS Windows-only camera mouse library in Java, it was not possible for FlashWords to directly use it as FlashWords is developed on different technologies, namely Adobe ActionScript. In this aspect, the REST API was ideal to loosely couple these two totally different technologies running on either the same machine or even distributed.

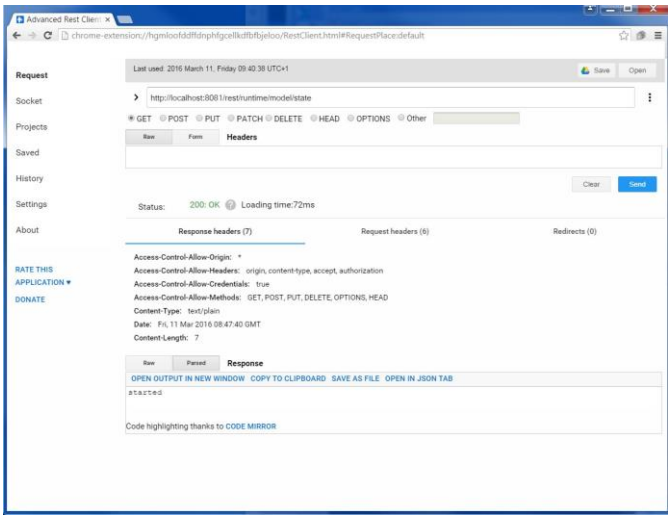

Fig. 5. The communication between AsTeRICS and the Advanced Rest Client.

4.1 AsTeRICS Camera-Mouse Model

The AsTeRICS Configuration Suite was used to design a model that captures the movement of the head to control the mouse. First it was confirmed that controlling the mouse by using the head is possible and that it functions adequately within the AsTeRICS framework. Next, the communication between the Runtime Environment and FlashWords was accomplished by developing a REST client. In specific, the needed functionality provided to FlashWords by the REST client was to load different models into the Runtime Environment via REST calls and to manage (start, stop and pause) models remotely so as to offer the assistive functions to the FlashWords software application.

For the above purposes a Google Chrome extension called “Advanced Rest Client” was used (see Figure 5). Other REST clients would also be an option; however, the “Advanced Rest Client” is easy to use with a clean interface and a good history function. The tool was used to precisely define the commands to be used to control the AsTeRICS Runtime Environment.

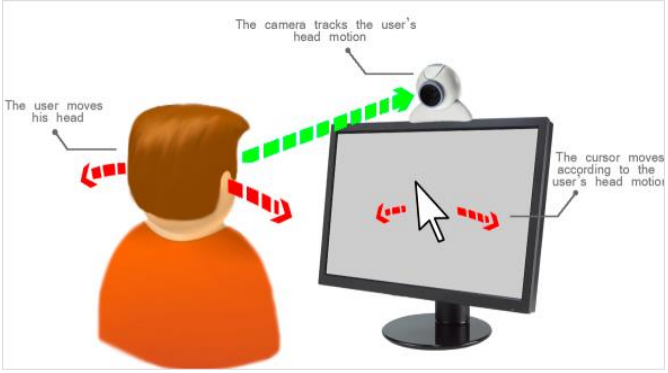

Fig. 6. People with limited movement abilities can control the mouse cursor via head movement and in this way use FlashWords.

4.2 The Integrated Application

The ability to communicate and utilise the assistive functionalities provided by the Runtime Environment opens up a new world of controlling FlashWords and providing enhanced user accessibility. With the help of the camera-mouse tracking and movement AT function, people with limited movement capabilities, for example people who can only move their head, can now easily control the mouse cursor and start to learn new words with FlashWords (see Figure 6). LIFEtool already supports a number of different ways to control the mouse (1-button scanning, 2 button scanning, dwelling, etc.). However this alternative user accessibility and interaction method is a valuable addition, which also satisfies assistive technology scenarios.

5 The EURO Use Case

The Runtime Environment was also tested and validated through the EURO application. EURO is an application that assists users to get to know money, not only the knowledge of the individual coins and banknotes, but also the knowledge of the value of money itself (see Figure 7). Target groups include children and adults with learning disabilities that use the EURO application to get a feeling for how much daily life costs, such as food and drug products, as well as electronic equipment and other everyday-life objects.

Fig. 7. By using this application users get to know individual coins and banknotes, as well as the value of money itself.

5.1 AsTeRICS Accelerometer Gyro Model

The REST API was used to achieve communication between the Runtime Environment and the EURO application in a similar manner as it was used for the FlashWords application described in Section 4. The AsTeRICS "Accelerometer" model was used to send mouse clicks by using the gyro sensor. As with the FlashWords application, the model is able to be loaded, started and stopped from within the EURO application via REST and offers the assistive functions within the application. The Gyro sensor control method was added to EURO options menu.

5.2 The Integrated Application

As with the FlashWords application, the REST API defines new ways of interaction and control of the EURO application. The user wears the Gyro sensor and is able to control the EURO application with simple wrist movements (see Figure 8). The main target group of the accelerometer input modality are people who are not able to control their hands or feet with precision in order to use an ordinary physical button.

Fig. 8. Controlling the EURO application with simple wrist movements - for people who are not able to control their hands or feet with precision.

6 Conclusions and Future Work

This paper describes how the novel REST-enabled runtime environment built over the AsTeRICS framework. This environment enables, simplifies and expedites integration of assistive functionalities into software applications that are built using different technologies. Also, the runtime middleware environment offers exploitation of the capabilities of sensors, actuators and other mobile devices deployed even on different machines for the development of assistive applications.

Through the two use cases it was demonstrated that new ways of AT interaction can be easily and rapidly integrated into existing software applications. This is supported by the REST-enabled runtime environment. The scenarios can be perceived as examples of rapid re-development of two AT applications, by including further important assistive functionalities and improving accessibility. In the first scenario the FlashWords application can now be used also by children with motor disabilities. Furthermore, the EURO application was upgraded in terms of AT capabilities to be able to be used also by people who are not able to control their hands or feet with precision in order to use ordinary physical buttons.

For future work, the target is to enrich the AsTeRICS Runtime REST API, so as to provide the capability for remote communication, not only at the level of the AsTeRICS Runtime Environment, but also at a lower level, the level of the AsTeRICS model components. By achieving this goal, models can be defined that use components available on different AsTeRICS-enabled machines. This will enable a component A available on one AsTeRICS machine to communicate with a component B available on another AsTeRICS machine. In this manner, advanced interaction capabilities between remote components will be provided for realising even more complex AT scenarios.

Acknowledgements. This work is supported by the European Commission as part of the Prosperity4All (Large Contribution) EU project funded by the Seventh Framework Programme – under grant agreement no 610510.

References

1. Eurostat: Population and Social Conditions: Percentual Distribution of Types of Disability by Sex and Age Group. Online. http://epp.eurostat.cec.eu.int
2. Assistive Technologies: Principles and Practice (2nd Edition) (15 December 2001) by Albert M. Cook, Susan Hussey
3. AsTeRICS Homepage (2015, April 17). Retrieved from http://www.asterics.eu/index.php
4. TOBI: Tools for Brain Computer Interaction, http://www.tobi-project.org
5. BRAIN: Brain-computer interfaces with Rapid Automated Interfaces for Nonexperts, https://www.brain-project.org/
6. BrainAble: Autonomy and social inclusion through mixed reality Brain-Computer Interfaces: Connecting the disabled to their physical and social world, http://www.brainable.org/
7. OSGi Alliance Homepage (2015, April 17). Retrieved from http://www.osgi.org/Main/HomePage
8. AsTeRICS Developers Manual, Chapter 8. Retrieved online: http://www.asterics.eu/download/DeveloperManual.pdf
9. Richardson, L., Amundsen, M., Ruby, S.: RESTful Web APIs. O'Reilly Media, September 2013.
10. Project Grizzly (2015, April 17). Retrieved from https://grizzly.java.net.
11. FlashWords App (2016, March 11) http://www.lifetool.at/assistive-technology/lifetool-developments/apps-for-tablet-computers/flash-words.html?L=1