AMBIENT ORCHESTRATION IN ASSISTED ENVIRONMENT

G. Brestovac[1*] – D. Marin[1] – T. Oroz[2] – A. Vidović[1] – S. Bozóki[3] – A. Grgurić[4] – M. Mošmondor[4] – T. Galinac Grbac[1]

[1]Department of Computer Engineering, Faculty of Engineering, University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia
[2]Department of Software Engineering and Information Systems, Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, 10000 Zagreb, Croatia
[3]Computer and Automation Research Institute, Hungarian Academy of Sciences, Hungary
[4]Research and Innovations Unit Ericsson Nikola Tesla, Krapinska 45, 10002 Zagreb, Croatia

ARTICLE INFO

***Article history*:**
Received: 5.7.2013.
Received in revised form: 2.12.2013.
Accepted: 2.12.2013.

Keywords:
Ambient assisted living
Smart home
Reference architecture model
Case study

Abstract:

Ambient Assisted Living (AAL) stands for information and communication technology enabled smart home environment that serves persons, especially the elderly and disabled in their independent living. Among many already developed AAL systems, technologies, resources and services, the main problem about their inherent interconnection still remains. One of the big research issues is to propose reference architecture and develop an open and standardized platform that should serve wider community as an enabler for cooperating concept – collaboration between competitors. This paper presents an experimental ambient orchestration in assisted environment on top of universAAL middleware that is based on the reference architecture under development within universAAL research project. Our contributions are the following: interconnection scenario using industry commercial products, verification of reference architecture and user guides, developed web services for ambient orchestration within case study and its demonstration in real environment.

1 Introduction

With a current demographical trend in Europe, projections for 2050 are two retired people per one working person. Furthermore, there is a growing amount of older people living alone and consequently a growing need for intensive care that is seriously questioning sustainability of the European Community. Information and Communication Technologies (ICT) are recognized as high potential coping with social and economic challenges imposed by this fact [1]. That is why the European Commission has defined a program within ICT research theme named *The European Ambient Assisted Living Innovation Platform called AALIANCE*. The main focus of this program is on the development of Ambient Assisted Living (AAL) solutions based on the advanced ICT technologies for the areas ageing at work, ageing at home and ageing in society [2]. According to their terminology, AAL refers to '*intelligent systems of assistance for a better, healthier, and safer life in*

* Corresponding author. Tel.: +385 91 150 4217 ; fax: +385 35 422 176
E-mail address: gbresta@hotmail.com

preferred living environment and covers concepts, products and services that interlink and improve new technologies and social technologies'.
Significant research has been devoted to home automation systems that control home appliances and features such as windows or lighting with the use of ICT and thus improve ease, security and energy efficiency. Another step forward is using *smart home technology* that can cope with the integration of technology and services through home networking for a better quality of living. Although there are numerous smart home automation systems developed and offered on the market, a complete exploration of ICT in this field is still limited [3]. There are two major obstacles. Firstly, the lack of technical convergence among different technologies and solutions complicate/s systems comparison and assessment of user experience while using these systems. Secondly, market fragmentation in sense that numerous devices capable of interconnecting in home environment systems are treated as separate market segments thus complicating offerings of such combined business models. Having removed these obstacles, ICT could be fully explored in AAL systems and would impact our daily lives heavily, maybe even more than the Internet revolution. Foreseen future directions are described in the long term technical vision in (http://aaloa.org/manifesto), [2]:

- evolution of hardware resources (displays, keyboards, storage, etc.) towards pluggable network resources causing the need for new programming languages that will be based on resource discovery paradigm;
- shift in developing software applications intended for particular operating system or resource towards developing software applications for AAL environments; and
- development of pluggable components intended for working in distributed environment thus contributing to distributed applications rather than development of software applications for specialized equipment having strong prerequisites for its operation.

All these reasons lead to setting the research direction towards the definition of reference architecture that would engage standardization of the resources available in AAL environments and their integration. The European Research Project called universAAL aims to develop an *open standardized* platform and *reference specification* for Ambient Assisted Living (AAL) [4]. The importance of this project lies in the creation of an open and unified environment to everybody that could enable rapid AAL solution development and deployment with the focus on *interoperability* and *standardization* [2]. A detailed exploration of research challenges in that direction is summarized in [5]. Besides technical innovation, the goal of the project is to convince all domain stakeholders of the quality of its outcomes. Therefore, evaluation framework has been developed [6] that is used to evaluate the outcomes of this project.
The work presented in this paper is contributing to this research field by exploring the possible interconnections among various commercial products and the universAAL platform. The point is to show not only that the universAAL platform is interoperable with commercial products of different producers, but to provide feedback for further development of universAAL platform as well. More precisely, our contribution to this research is as follows.

- Firstly, we developed interconnection scenarios where univerAAL platform as described in universAAL reference model [7] is integrated with commercial products such as Kinect and Ericsson IP solution ZyXEL STB.
- Furthermore, we verify universAAL platform interfaces and provide feedback on usability of technical documents and support.
- Finally, we illustrate the use of universAAL platform in a case study in real-network environment.

The Ambient orchestration environment system developed provides a simulation and useful tool for understanding the interaction between ourselves as individuals and our environment. It also provides an understanding of the impact of technology in people's daily life. The presented architecture in Section 2 is based on universAAL platform with commercial products IPTV, real lamps, Kinect™, Android device, Smart Home Simulator. The user is able to operate real devices. Interacting scenarios in between commercial devices over universAAL platform are presented in the form of sequence diagrams in Fig. 3, 4, 5, and 6 in Section 3. In the same section, the discussion of implementation details that were used in verification is described. Finally, in Section 4, we conclude the paper. The

working product is demonstrated in a short video accessible from http://www.seiplab.riteh.uniri.hr/?page_id=193&lang=en.

2 Related work

Numerous efforts have been devoted to the definition of reference architecture for ambient assisted living environments. Mostly, they were motivated by the idea of providing unified middleware architecture for heterogeneous networks that would bring numerous commercial and cost benefits. The problem has arisen/arises with the existence of a number of different technologies and each is represented as one closed network with limited integration ability in common home architecture possibilities. The technologies from heterogeneous networks are represented as a group of conforming devices that can talk and cooperate with each other and form a working and independent environment but closed within the home system.

Early works on design of the middleware that would bridge several heterogeneous closed network systems thus aiming to address interoperability issues failed mainly because of weak use of open standards [8], [9], [10]. A middleware framework for home environments that is based on the usage of open standards and Service Oriented Architecture (SOA) is presented in [11], [12]. In [12], a middleware solution based on SOA is proposed. However, the prototype demonstration is performed only within simple lighting domain. This paper shows an integration of several domains. By integrating new domains, a number of new issues are opened. For example, mobile computing technology integration has / arouses a problem of context diversity and its integration into a single context-aware solution. A service oriented middleware for integrating context-aware technologies has been proposed in [13] and a survey of context-aware systems is provided in [14]. The work presented in [11] is interesting because it addresses a dynamic discovery and a composition of Web services in the mobile context. Dynamicity is addressed by the application of Service Oriented Architecture as for example in OSGi platform [15]. Also, it is used in [16] within a domotic infrastructure based on Web services stack. In our example, the universAAL platform is built on SOA architecture and new technologies are integrated by simple dynamic Web service extension. Recent works in the field have explored human aspects and integration of people activities [17]. Experiments with gesture interaction within the specified universAAL environment are presented in [18].

Finally, the European Ambient Assisted Living Innovation Alliance called AALIANCE has proposed an AALIANCE system composition reference architecture within AAL Roadmap document [19] that aims to provide enabled innovation for defining an open middleware architecture for heterogenous networks. It consists of three layers that makes it suitable for building middleware platforms aiming to enable communication and connectivity between various devices and services. The universAAL architecture used in the case presented in this document is built on the basis of AALIANCE reference architecture.

3 Ambient orchestration environment and architecture

In this paper, we proposed an ambient orchestration environment that enables interactive actions in smart home environment. Ambient orchestration environment realizes communication among following devices within the house: real lamps, Smart Home Simulator, Android phone, Kinect and IPTV as presented in Fig. 1. In essence, it is a software communication system that implements web services (developed in this project) based on the usage of AAL technology. Software architecture could be functionally decomposed into following main building software components (see Fig. 2):

Lighting Server functional entity is a software application implemented on a personal computer that implements the lamp control service. In our implementation, this service offers control functions like turning on/off, dimming of lights and getting information about current lamp state. Lamps are controlled via X10 protocol that enables communication with various appliances through power line control. In the universAAL terminology, Lighting Server represents universAAL server that offers its services to various potential clients.

IPTV Client is a software application integrated with ZyXEL IPTV Set-Top Box (STB). It represents a universAAL client that connects to the Lighting server and utilizes its services. User operating IPTV client is able to view and control states of various

lamps (for example, existing lamps of their homes) and to turn them on and off with a simple click on the remote control of IPTV.

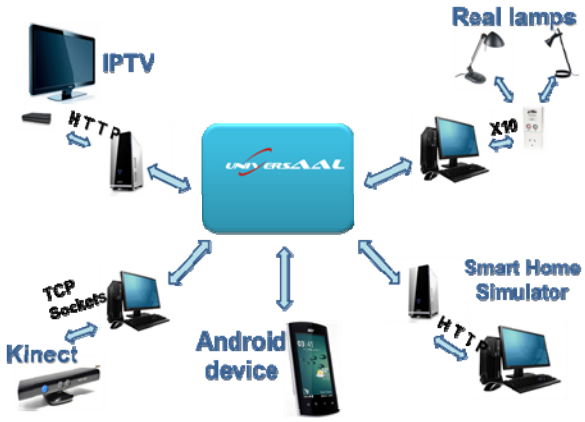

Figure 1. Smart home devices involved in ambient orchestration environment.

Android Client functional entity is a software application for Android based Smartphone. It represents a universAAL client that connects to the Lighting Server, giving the end user possibility to control any chosen lamp via his/hers Android device. Moreover, Android Client gives the end user possibility to record actions and to create *macros*. An example of a simple macro would be: *when Lamp1 is turned on, automatically turn on Lamp2.*

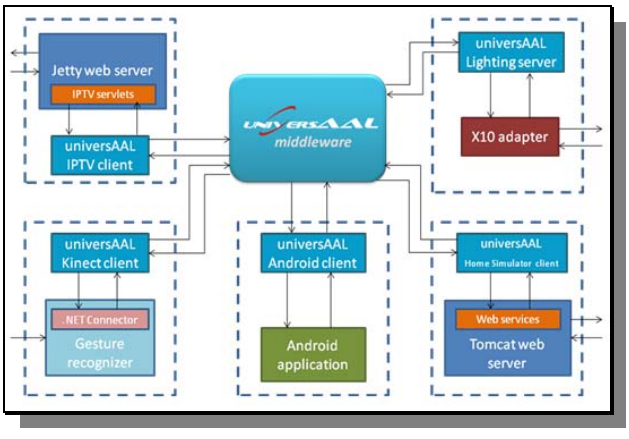

Figure 2. Functional decomposition of system.

Home Simulator is a software application for personal computer that simulates the real state of the user's house environment. On a simple layout of the house, the user is able to see all the lights in the house, their given names, states and rooms in which they are located. The user is also provided with the lighting control services for every light in the house. This functional entity utilizes services offered by Lighting Server, thus becoming a universAAL client.

Kinect Client is a software application for Kinect devices. Kinect is a motion sensing and gesture recognition input device which, in this case, allows the user to control lamps by simple body movement of his choice. For example, swiping the right hand from left to right could/ be instruction for turning on/refer to *turn on*. Thus, if the user choose a specific lamp and swipe his right hand from left to right, the lamp would be turned on. Kinect Client provides the end user with the same functionalities as other aforementioned universAAL clients. More details about research problems and implementation issues encountered during designing process of this application can be found in [18].

4 Use cases

Implementation of automation system presented in Fig. 1 could be described using Use Case notation. Each use case describes a communication interaction. Use cases are described in the following subsections.

4.1 Andriod Client

The Android part of ambient orchestration environment consists of two platform external functional modules. The former is the Android application – Android Client, and the latter is the universAAL Android middleware application.

Before using Android Client, it is necessary to start the universAAL Android middleware application, as shown in the Fig. 3. It will establish the communication between Android Client and universAAL server using a special Android component – a broadcast receiver. The broadcast receiver is an Android component through which an Android application is able to register itself for various system or application events. The registered Android application (it is Android Client application in our case) is called the receiver and it will be notified by the universAAL Android middleware application when the event from the universAAL server happens/occurs. This event can be a list of controlled lamps, the result of the request previously sent to the universAAL server, or the state of controlled lamps, as shown in the Fig. 3. Table 1 presents the most important methods used in the Fig. 3 through which an Android Client is able to get information and control any chosen lamp and to

create macros (e.g., if the state of Lamp1 is changed to *off*, automatically change the state of Lamp3 to *on*).

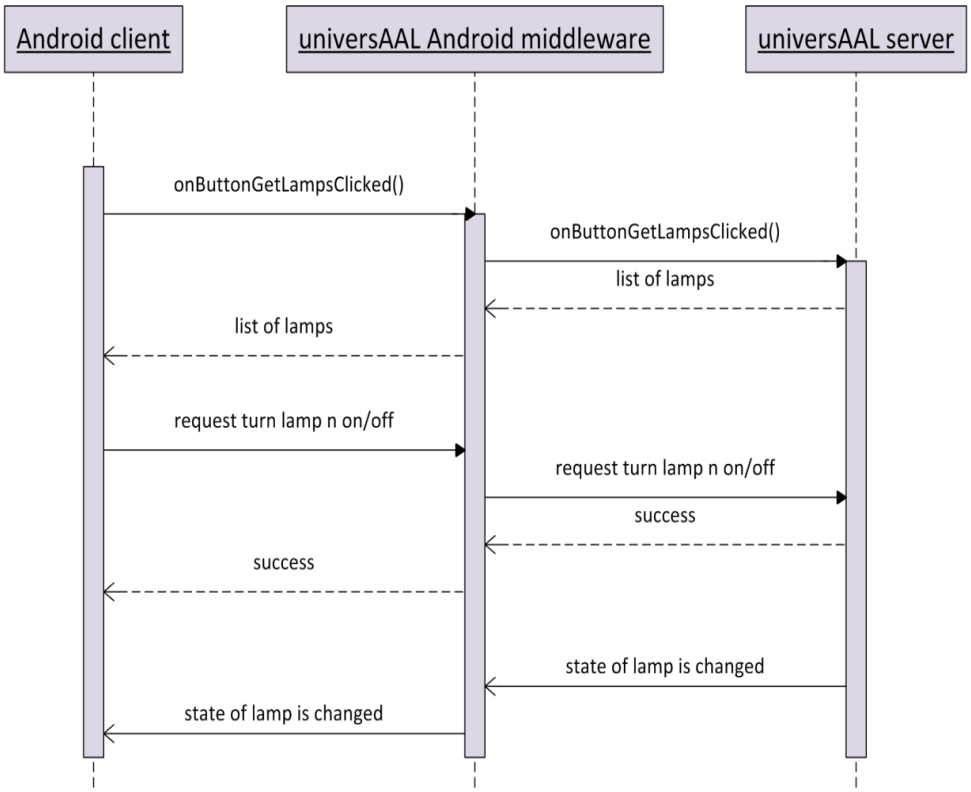

Figure 3. Sequence diagram of Android Client - universAAL Server communication through universAAL middleware.

4.2 IPTV Client use case

This section explains the use case for IPTV application presented in Fig. 4. The possibilities offered by IPTV are:

- getting the list off/of accessible lights (button Ok)
- selecting the proper light (remote buttons Up or Down)
- turning on or off the light on selected objects (remote buttons Right or Left)
- turning all lights on or off (remote buttons 7 or 9)

The listed functionalities are enabled on a simple user interface developed with JavaScript and HTML web development technologies. The background service is developed in Java in order to communicate with the universAAL middleware. Table 2 explains the most important methods in the background services and Table 3 explains the most important functions in JavaSript. Those functions are used to explain sequence diagram presented in Fig. 4 in the text below.

The first request (while starting application), that is sent from IPTV (set-top box) to a web server, is used to invoke the main index.html page. This web server is started inside the universAAL client. The universAAL client has registered a servlet that has the possibility to receive and respond to HTTP requests coming from outside. On the first call, IPTV gets the main index.html page which is interpreted as a home page. After this initial call, the IPTV application is waiting for human interaction. When a person presses the OK button on a remote controller, IPTV sends the HTTP request to get the list of all available lights. Registered servlet receives the request and forwards it to the universAAL middleware as a service call.

Table 1. List of methods in Android Client application

Method name	Description
onButtonGetLampsClicked()	Register a new broadcast receiver that will listen for/to a response from the universAAL server
analyzeGetLampsResponse(intent)	Update the GUI of Android Client with the list of lamps
onButtonOffClicked(View v)	Create the service request to turn the lamp off
addSelectedLampNumber(intent)	Incorporate the selected lamp number in an intent
invokeIntent(intent)	Send created intent in addSelectedLampNumber(intent) method to the universAAL Android middleware application
onButtonOnClicked(View v)	Create the service request to turn the lamp on
onReceive(Context context, Intent intent)	Capture information (or context events) from universAAL server about lamp state change
displayNotification(String message, Context context)	Display captured information from onReceive(Context context, Intent intent) method on the GUI of Android Client

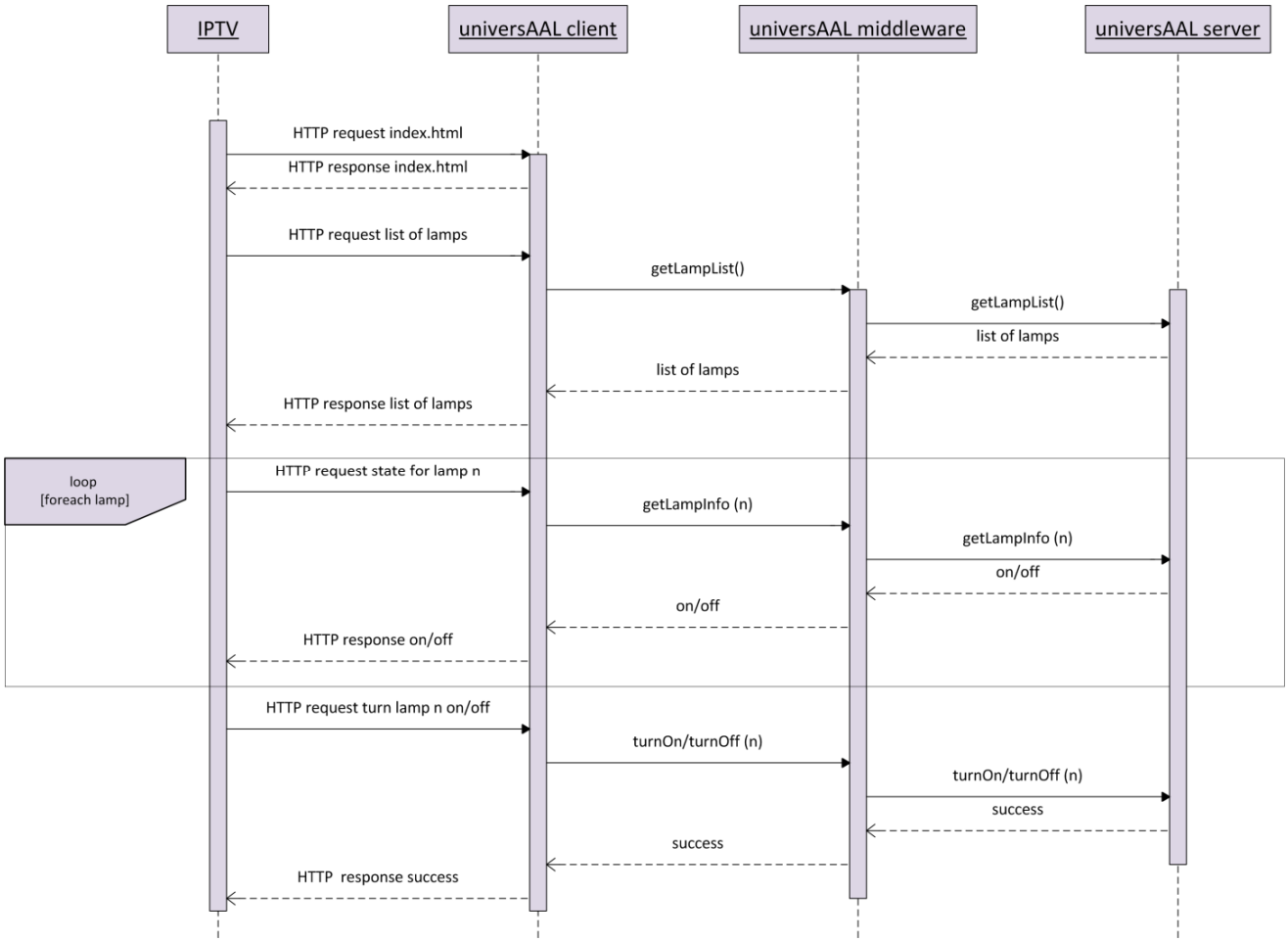

Figure 4. Sequence diagram of IPTV Client - universAAL server communication through universAAL middleware.

Table 2. List of methods in IPTV Client application (Java)

Method name	Description
turnOnRequest(String lampURI) / turnOffRequest (String lampURI)	Making requests to universAAL middleware for turning On / Off a lamp with the specific URI
getLampStateRequest(String lampURI)	Make request for the state of the light with the specific URI
getControledLamps()	Make request for the list off all controlled lights

Table 3. List of methods in IPTV GUI application (Javscript)

Method name	Description
move(step)	Function to enable moving through the list of lights on the screen with up and down arrow
getStates()	Reading light states and filling the appropriate space on an IPTV page with the exact HTML code for displaying lights states
startPooling()	Function which periodically makes calls to a web server to check for light state changes
turnOn() / turnOff()	Invoke the service for turning on or off the specific light selected on the user interface
allOn() / allOff()	Function for calling function turnOn()/turnOff() for each light available for controlling
readLampStates(lightId)	Function for making request for the light state of the specific light

The universAAL server serving the information about available lights generates the information and sends it to the universAAL middleware as a response. This response gets to the universAAL client that then forwards it as a HTTP response to the IPTV. This is the end of the first interaction. After the IPTV gets the names of available lights, it makes separate HTTP requests for each light. These requests are made in order to receive the states of each lamp. Each request contains the light name which enables getting the exact light state. For each request, registered servlet makes service calls to universAAL middleware, and waits for the response of a relevant universAAL server. Every received answer is then sent to IPTV for the proper display of the light state. The last important interaction shown on the sequence diagram changes the state of the selected lamp to on or off. This interaction consists of sending the HTTP request to the registered servlet, which then forwards it to universAAL middleware. In the end, the responsible universAAL server can execute the request and return a message as successful or unsuccessful actions.

4.3 Web application *Home Simulator* use case

Fig. 5 presents the use case responsible for communication between *home simulator*, universAAL client, universAAL middleware and universAAL server. For each lamp, the home simulator sends HTTP request to get the state of each lamp from the universAAL client and thus refreshes its state. To know the state (on or off) of each lamp, the universAAL client has to contact universAAL middleware. UniversAAL middleware then proceeds with a request to the universAAL server, which then sends the lamp state to the universAAL middleware, which then forwards it to the universAAL client. This client then sends HTTP response to the home simulator with the state of a particular lamp – on or off. The same process continues in a loop for every lamp.

To change the state of the selected lamp to on or off, the home simulator sends a HTTP request to the universAAL client that then uses TurnOn or TurnOff method. It then proceeds to the server, which sends the status- success to the universAAL middleware and then universAAL middleware forwards it to the universAAL client. This client

sends HTTP response to the home simulator, which then turns on or off and switches the selected lamp on or off. Also, the server's use as a context publisher is to publish context – the lamp state changed to the universAAL middleware. The client receives it and then sends HTTP request to the home simulator to refresh – change the state of a lamp.

4.4 Lighting Server use case

This section explains the use case for communication with real lamps presented in Fig. 6. Lighting Server offers the following services:

- listing all the lamps that are controlled by it and
- allowing the client to manipulate them (turns each of them on/off or dims it to a certain value).

For those purposes, the Server has to have knowledge about an initial lamp state, e.g., if the chosen lamp is on or off at the moment, the Server is started. Therefore, the first action of communication performed by the Server is "asking" the lamp about its current state.

The Server is not asking this directly, but sends a "lampState" request to the X10 controller. X10 controller has the role of a mediator in this communication because of the ability of establishing a direct connection with any appliance (in this case, the lamp) via power line.

On receiving "lampState" request from the Server (and in this request, Server must send some additional info - e.g., the state of the lamp requested by the actuator), X10 controller sends an X10 command (via power line) asking the given lamp its state. The response comes in the form of integer value 1 (if the lamp is turned on) or 0 (if the lamp is off). This action is repeated for every lamp controlled by the Server (in this simple sequence diagram, there is only one lamp, but that is not the real-life situation).

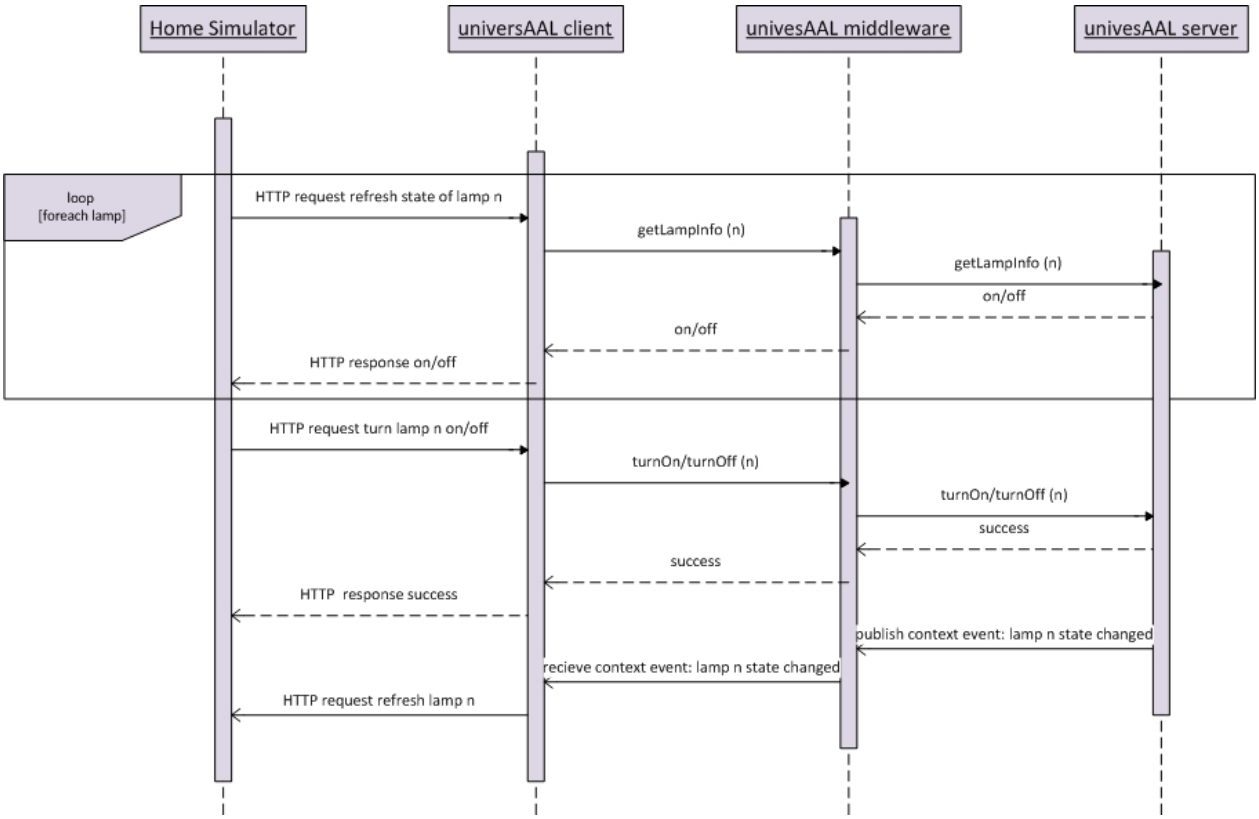

Figure 5. Sequence diagram of Home Simulator - universAAL server communication through universal middleware.

Table 4. List of methods in Home Simulator

Method name	Description
start()	Create the GUI , place the buttons on specific coordinates, sets size
getGetLampsAction()	Gets accessible lamps
LightClient()	Calls initGUI method that has GUI dimensions
initGUI()	Starts GUI with dimensions
getOn()	Get new on state
getOff()	Get new off state
getScale()	Scales - dims the lamp to specific value
updateHomeSim()	Updates Home Simulator, gets list of lamps, calls the method from Lighting Consumer which turns a specific light on Simulator
dimToValue	this method dims the light of a specific lamp
isOn	calls and handles the getLampStateRequest
getLampStateRequest	registration, creation and definition of a service request for the purpose of getting the lamp state
getLampInfo	Used to find out whether the lamp is turned on or off

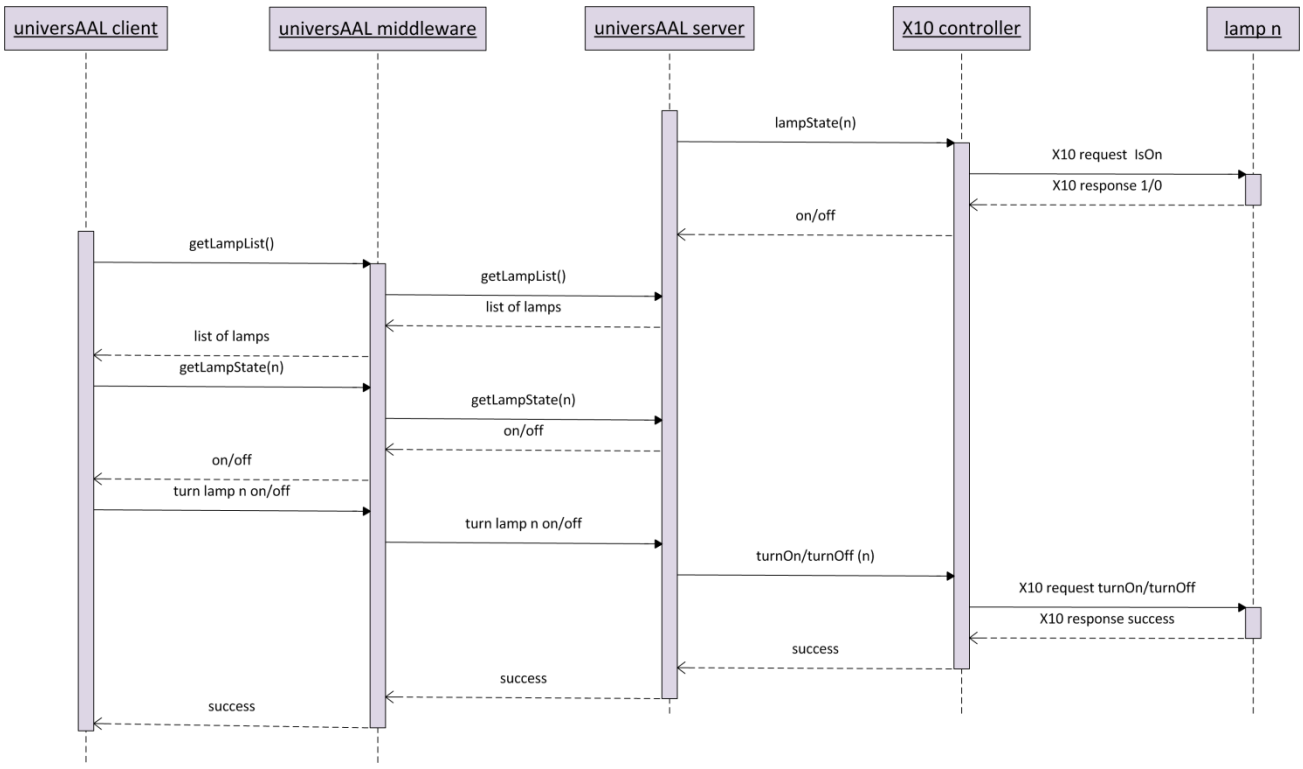

Figure 6. Sequence diagram of universAAL client - universAAL server (Lighting Server) communication through universAAL middleware.

Once the states of all lamps are updated at the Server, they are ready to offer their services to any potential Client that needs them (when talking about the "Server" and the "Client", it needs to be understood that it is meant in the context of universAAL - the Server represents the universAAL server and the Client represents the universAAL client; they communicate through universAAL middleware – the Server offering its services and the Client calls on performing them).

Once the certain Client is registered as a node on universAAL middleware, it can call on any service

that Server offers. For example, if it calls on the "GetLampList" action, the Server will respond with the list of lamps. If the Client calls on "GetLampState", the Server will respond to universAAL middleware with its services existing for the request to go all the way to the real lamp. However, in case the Client wants to change the state of the lamp, communication stretches through several layers: first, the Client calls on the action "turn lamp on/off", then this request is forwarded to the Server through universAAL middleware, then the Server calls on X10 controller to perform this action and X10 controller sends the "turnOn/turnOff" command to the chosen lamp.

5 Conclusion

This paper presents (our work on) the implementation of the AAL reference model on an ambient assisted orchestration environment proposed by universAAL research project that is still in progress.. Interactive actions among commercial products from several vendors are enabled in a smart home environment. The research presented in this paper was concentrated around the universAAL platform and the reference AAL architecture. The main results are about/deal with the interconnectivity among different commercial products using universAAL platform. In particular, the features of the reference platform are enhanced with commercial products such as Kinect™ and ZyXEL STB, thus contributing to the reference architecture development itself. This is achieved by developing new software applications that operate in distributed environment explained in detail in this paper.. Another important contribution is in verifying the platform interfaces and in providing feedback on usability of technical documents and support. In this paper, both the ambient orchestration architecture and use cases have been discussed in detail. In addition to the material presented here, we provide demonstration of operable ambient orchestration environment and all the applications and services that have been studied. Numerous technical details were discussed and solved during the project presented in this paper. It is worth mentioning that this project has been realized with the help of students with no previous programming background in universAAL architecture for this 6- week project. Thus, we have proved usability of universAAL platform documentation and support and we are able to easily create new services that could be fully integrated in heterogeneous home environment.

References

[1] Kubitschke L., Cullen K.: *ICT & Ageing – European Study on Users, Markets and Technologies*, European Commission, Directorate General for Information Society and Media, Brussels, 2010.

[2] Aquilano, M., van der Broek, G., Cavallo, F.: *Ambient Assisted Living – Strategic Research Agenda*, VDI/VDE-IT AALIANCE Office, Berlin, 2010.

[3] Fagerberg, G., Kung, A., Wichert, R., Tazari, M., Jean-Bart, B., Bauer, G., Zimmermann, G., Furfari, F., Potort, F., Chessa, S., Hellenschmidt, M., Gorman, J., Alexandersson, J., Bund, J., Carrasco, E., Epelde, G., Klima, M., Urdaneta, E., Vanderheiden, G., Zinnikus, I.: In: Lukowitz, P., Kunze, K., Kortuem, G. (Eds.): *Proc. of the 2010 European conference on Platforms for AAL applications. 5th European conference on Smart sensing and context* (EuroSSC'10), Berlin, Germany, 2010, pp.177-201.

[4] Hanke, S., Mayer, C., Hoeftberger, O., Boos, H., Wichert, R., Tazari, M.R., Wolf, P., Furfari, F.: *universAAL - An Open and Consolidated AAL Platform*, 4. AAL-Kongress 2011, In Advanced technologies and societal change: Ambient Assisted Living, Berlin, Germany, 2011, pp. 127-140.

[5] Grgurić, A.: *ICT towards elderly independent living*, Zagreb, 2012, http://www.fer.unizg.hr/_download/repository/A._Grguric,_rad_KDI.pdf

[6] Salvi, D., Montalvá Colomer, J.B., Arredondo, M.T., Walderhaug, S.: *Evaluation of AAL platform and services: The universAAL case*, AAL Forum 2012, Eindhoven, Netherlands 2012.

[7] Tazari, M., Furfari, F., Fides-Valero, A., Hanke, S., Hoeftberger, O., Kehagias, D., Mosmondor, M., Wichert, R., Wolf, P.: *The universAAL Reference Model for AAL, Ambient Intelligence and Smart Environments. Handbook of Ambient Assisted Living*, IOS Press, 2012.

[8] Moon, K.D., Lee Y.H., Lee C.E., Son Y.S.: *Design of a Universal Middleware Bridge for Device Interoperability in Heterogeneous*

Home Network Middleware, IEEE Transactions on Consumer Electronics, 51 (2005), 1, 314-318.

[9] Tokunaga, E., Ishikawa, H., Morimoto, Y., Nakajima, T.: *A Framework for Connecting Home Computing Middleware*, *ICDCSW Conference*, Vienna, Austria, 2002, 765-770.

[10] Moon, K.D., Lee, Y.H., Son Y.S., Kim, C.K.: *Universal home network middleware guaranteeing seamless interoperability among the heterogeneous home network middleware*, IEEE International Conference on Consumer Electronics, Los Angeles, USA, 2003, 306 - 307.

[11] Issarny, V., Sacchetti, D., Tartanoglu, F., Sailhan, F., Chibout, R., Levy, N., Talamona, A.: *Developing Ambient Intelligence Systems: A Solution based on Web Services,* Automated Software Engineering, 12 (2005), 1, 101-137.

[12] Miori, V., Tarrini, L., Manca, M., Tolomei, G.: *An open standard solution for domotic interoperability*, IEEE Transactions on Consumer Electronics, 52 (2006), 1, 97-103.

[13] Gu, T., Pung, H. K., Zhang, D.Q.: *A service-oriented middleware for building context-aware services.* Journal of Network and computer applications 28 (2005), 1, pp.1-18.

[14] Baldauf, M., Dustdar S., Rosenberg, F.: *A survey on context-aware systems,* International Journal of Ad Hoc and Ubiquitous Computing, 2 (2007), 4, 263-277.

[15] OSGi Alliance group, *Open Service Gateway initiative (OSGI), OSGI Service Platform Version 4*, http://www.osgi.org, 2008.

[16] Aiello, M., Dustdar, S.: *Are our homes ready for services? A domotic infrastructure based on the Web service stack.* Pervasive and Mobile Computing 4 (2008), 4, pp. 506-525.

[17] Kahri, P.: AAL-2009-2, *ICT based solutions ICT based solutions for Advancement of Social for Advancement of Social Interaction of Elderly People Interaction of Elderly People*, ICT 2008, Lyon, 2008.

[18] Broek, G., Cavallo, F., Odetti, L., Wehrmann, C.: *AALIANCE Ambient Assisted Living Roadmap*. VDI/VDE-IT AALIANCE Office, Berlin, Germany, 2009.

[19] Grgurić, A., Mošmondor, M., Kušek, M, Stocklow, C., Salvi, D.: *Introducing Gesture Interaction in the Ambient Assisted Living Platform universAAL*, 12th International Conference on Telecommunications ConTEL 2013, Zagreb, Croatia, 2013, pp. 215-222.