A Context-aware, Intelligent and Flexible Ambient Assisted Living Platform Architecture

Hendrik Kuijs, Carina Rosencrantz, Christoph Reich
Faculty of Computer Science
Furtwangen University of Applied Science
Furtwangen, Germany
Email: {Hendrik.Kuijs, Carina.Rosencrantz, Christoph.Reich}@hs-furtwangen.de

Abstract—**Ambient Assisted Living (AAL) solutions support older people in remaining longer in their own environment. For a system to be successful on the market, it is essential to be flexible and adaptable to the individual needs of the elderly person. In this paper, we present an approach for a context-aware, intelligent and flexible AAL platform architecture, that integrates existing concepts for home automation environments with an extendable platform for information, communication and learning to assist elderly users in their daily life. The platform has the intelligence to react on environmental changes, by including data provided by sensors or external services, as well as changes of the medical state of the user by using a person centered ontology to deliver adapted services at any time. The intelligence to do this is assured by lightweight and autonomous software agents. The custom platform itself is realized as a Platform as a Service (PaaS) in the cloud. The setup is a Private Cloud, that shares central services in the Public Cloud for better flexibility, scalability and maintainability. It features a PaaS management system to customize and preconfigure different environmental settings. The user is able to add new services on demand or adjust the configuration of the platform to his needs.**

Keywords*—*PaaS; AAL; Cloud; OSGi; software agents; context-aware.

I. INTRODUCTION

Due to the demographic change towards an aging population and the emerging shortage of care facilities, the field of AAL aims to support elderly people in their daily living to enable them to stay in their own homes as long as possible.

As part of the research project ZAFH AAL [1], the platform named Person Centered Environment for Information, Communication and Learning (PCEICL) has been designed [2]. PCEICL is a personal assistance system with the primary goal to assist elderly people in staying at home longer and in improving social participation in rural regions by delivering social services. An essential requirement for such a platform is an automatic adaptation and tailoring of the services to an elderly maybe handicapped person's needs.

In the field of AAL, Open Services Gateway initiative (OSGi) [3] is often used to provide an easy integration framework for sensors and actors in home-automation environments [4][5][6]. OSGi enables developers to build up modular systems or to reuse existing services and therefore supports the upgradeability and extensibility of AAL systems [7].

The key feature of PCEICL is the adaptation of functionality and presentation of information based on the medical state and the physical environment of the user. Information about the user is stored and retrieved by implementing the PCEICL ontology [8] and information about the environment is provided by attached sensors or external web-services. The platform makes intelligent decisions based on the provided information about the user and its environment. The PCEICL platform tackles the problem of intelligent decision making by a software agent platform, in this case Java Agent Development Framework (JADE) [9]. While other platforms have a strong focus on emergency detection and prevention, the main concept of PCEICL is to provide information and communication services to support social participation.

The outline of this paper is as follows: After presenting related work in Section II, Section III explains two different concepts for combining the OSGi component middleware and the software agent framework JADE. The architectural approach of the PCEICL platform with its different layers of agents, OSGi bundles and the PCEICL ontology is explained in Section IV and is put in relation to the cloud infrastructure in Section V. The flexibility, context awareness and intelligence of this architectural approach are worked out in Section VI. Section VII presents a first evaluation based on different scenarios, followed by a conclusion and a further outlook of upcoming research topics in Section VIII.

II. RELATED WORK

OSGi gained wide currency in the field of home automation and smart home as a middleware for different sensors and actors [3]. ProSyst [10] offers a *Home Gateway Middleware*, a *SDK* for developers and a *Remote Management Service* for smart home service providers. This middleware is used, for example, by Miele [11] to connect different household appliances to deliver automated services based on user interaction or environmental changes, like starting the vapor departure hood when the stove or oven are used or starting the washing machine when power is cheaper. ProSyst based environments are hosted locally in the user's home and can be monitored or controlled with mobile devices via web interfaces. PCEICL puts the user assistance in focus and wants to migrate the hosting environment to the cloud. ProSyst also supports E-Health scenarios [12] and is involved in several Ambient Assisted Living projects, like SOPRANO [4] or universAAL [5].

In the field of AAL, there are currently two projects also developing a platform or a middleware which realize an assistance-system for elderly people: *SOPRANO* and *universAAL*.

The *SOPRANO* ***S****ervice* ***O****riented* ***PR****ogrammable smA****r****t enviro****N****ments for* ***O****lder Europeans (SOPRANO)* project developed an open middleware for AAL solutions. The *SOPRANO Ambient Middleware (SAM)* receives user commands or sensor data, enriches them semantically and determines an adequate system response, which is then performed by the connected actors installed in the living environment. If, for example, SAM receives the information that a window is open, it analyses the remaining context information and can inform the user about the open window before he is leaving the house. The components communicate over semantic contracts and are based on a common domain ontology. This ontology is designed state-driven and every concept (device, person, location, etc.) of the ontology is represented by its actual state. The PCEICL platform, on the other hand, focuses on the user. The most important is to describe the user, since for information retrieval the user's condition is essential.

The ***UNIVER****sal open platform and reference* ***S****pecification for* ***A****mbient* ***A****ssisted* ***L****iving) (universAAL)* project [5] aims to join different approaches from lots of projects to a unique AAL solution. One of this included projects is SOPRANO [4]. The goal of universAAL is a platform, that makes it viable to develop AAL services. To meet this requirement, there will be developer tools, a store for distributing AAL services and a runtime environment to support all stakeholders. The universAAL platform is based on OSGi and ontologies are used as a common language for the components, too. But due to the OSGi-agent combination and the use of cloud technologies the PCEICL platform is more flexible and intelligent than the universAAL or the SOPRANO approach. The advantages of using agents within the OSGi framework and the usage of cloud technologies are described in Section VI.

Software agents are used in the *Emergency monitoring and prevention (EMERGE)* project [13] by the *Event-driven Activity Recognition System (EARS)* to process detected events by sensors to system reactions. Sets of detected events can be combined to assumed activities of the user. Self-StarMAS [14] uses agents to automate the configuration of devices in complex AAL scenarios. In the PCEICL approach JADE agents are used to deliver information from the user-centered ontology and to adapt the user experience and assistance by intelligent decision making.

III. COMBINING OSGI AND AGENTS

To tailor the functionality and information presentation according to the user needs, we integrated the software agent platform JADE for supporting intelligent behavior with the OSGi platform, which is often basis of smart home infrastructures.

There are several related studies that deal with the combination of OSGi and JADE. They can be divided into two main approaches: a) Deploying behavioral patterns of agents as bundles in OSGi on top of the agent system or b) setting up the JADE system as an OSGi bundle.

a) Gunasekera et. al introduced VERSAG (VERsatile Self-Adaptive aGents), a general architecture for lightweight flexible multi-agent systems (MAS) [15]. Based on this architecture they present a concrete implementation for JADE and OSGi [16] (see right part of Figure 1).

In the VERSAG approach, software agents have the ability to share their *Capabilities* with other agents. *Capabilities* are OSGi bundles, that run within the agents. The kernel is managing the agents and can pass the control over to other modules. The *Itinerary Service* manages the route for the mobile agents and provides them with the needed information about the distinct location, in which the agents have to operate. The *Capabilities Repository* holds all information about the application specific *Capabilities*. The OSGi container is located in the *Capabilities Execution Service* and runs *Capabilities* that are available in the *Capabilities Repository*. In addition, the *Capabilities Execution Service* is hosting an *Adaption Service* that contains the logic for adapting the agent according to the context, and a *Context Service* that influences the aforementioned adaptation by providing context information. The configuration and implementation of VERSAG requires several significant changes to the OSGi platform, to use it within the agent.

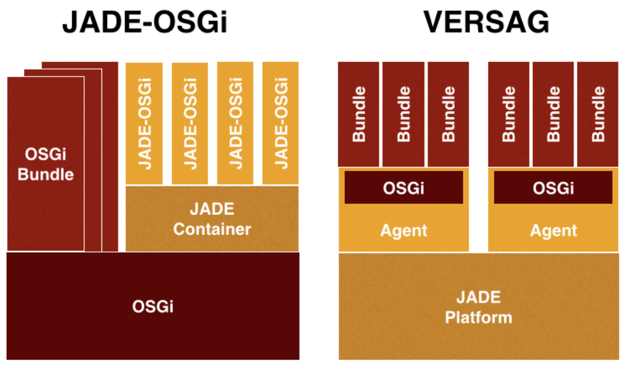

Fig. 1. Overview VERSAG Architecture

b) In Carneiro et. al [17], the JADE-system is running as an OSGi bundle and controls and manages other JADE-OSGi bundles that are independent software agents.

Jaszczyk and Król [18] follow the same approach, but divide the devices into three tiers: 1.) *Agentless Devices*, that do not have the power to provide a runtime environment with agents but have interfaces to receive FIPA-based [19] Agent Communication Language (ACL) messages [20], 2.) *Agent Devices* that are powerful enough to provide their own OSGi platform for software-agents or auxiliary OSGi bundles, and 3.) one *Main Device* that is a JADE Main Container where all other JADE Containers are registered.

Telecom Italia has developed an official JADE-OSGi bundle since JADE version 3.7, that is compatible with OSGi compliant frameworks since version 3.4 [21]. The installation and minimal configuration of JADE-OSGi is fairly easy and fast. The basic idea of JADE-OSGi is that each agent is located in a single bundle (see left part of Figure 1). The advantage of this is the possibility to use standardized OSGi actions, e.g., update agents separately during runtime by using the OSGi update functionality or deploying a *Management Agent* by using the *Configuration Admin Service*. Besides this, it is also possible to deploy other OSGi bundles in this environment and agents can access the services of these bundles. This leads to the possibility to keep the agents simple by reusing functionality that is provided by other bundles. The dynamic and modular architecture of OSGi is extended by JADE-OSGi and introduces additional intelligence by adding flexible agents to the framework.

The approach b) setting up the JADE system as an OSGi bundle, is also the approach taken in the PCEICL platform.

IV. Platform Architecture of PCEICL

The PCEICL platform combines the modular service platform OSGi (Equinox [22]) and intelligent software agents using JADE-OSGi [21]. The platform is built up on an infrastructure, that is provided by the Cloud Management System (see Section V). First an architecture overview is given, second an architecture layer model is described and finally it is shown, how the PCEICL ontology is integrated into the platform.

A. Architecture Overview

PCEICL architecture is based on the OSGi platform with several OSGi bundles for common services, like *Sensor Bundles*, *Smart Home Control Bundles*, an *Address Book Bundle* or a *Web-Interface Bundle* (see Figure 2).

JADE-OSGi is hosting the software-agents system but can also register and communicate with agents, that are OSGi bundles themselves. Only the agents have access to the PCEICL-Ontology [8] and pass the aggregated data to the service bundles. Authentication and access control is managed by the *Authentication* and *Security Module* that is also securing the OSGi framework. For example, a *Message Access Control Module* controls the messages flow between the various OSGi bundles in detail. Special OSGi bundles provide services for installing new bundles and functionality or for updating existing ones through a central *Bundle Repository* as described in Section V.

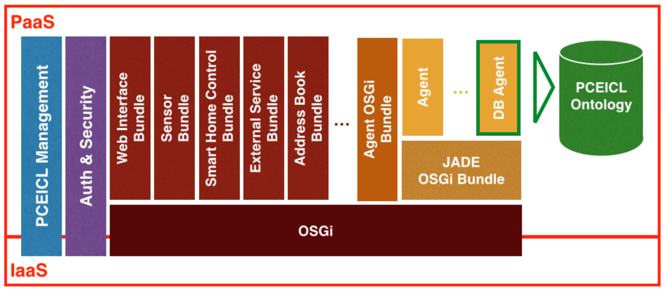

Fig. 2. Overview of PCEICL Architecture

B. Architecture Layer Model

The architecture of the PCEICL platform can be divided into three different layers (see Figure 3):

- *Agent Layer*
  All agent bundles are located in the *Agent Layer*. Communication between agents within this layer is performed by Agent Communication Language (ACL) [20].

  There are *Application Agents*, like the *Reminder Agent*, which can be developed by external developers and provide several services for the user. Other agents are *System* and *Smart Agents*, which are part of the PCEICL platform. *Smart Agents* are for example domain expert agents, which have access to the domain ontology data and are able to execute ontology reasoning. *System Agents* can be used by other agents for system matters, like access control or sensor data acquisition.

- *OSGi Bundle Layer*
  In this layer, all OSGi bundles are placed, that provide services. These bundles are usually provided by smart home service providers. They integrate sensors, like temperature sensor, window status, etc.

- *External Service Layer*
  This layer provides services, that can be accessed from outside of the OSGi platform, e.g., a weather web service.

The communication between the *Agent* and the *OSGi Bundle Layer* is implemented by the OSGi framework itself.

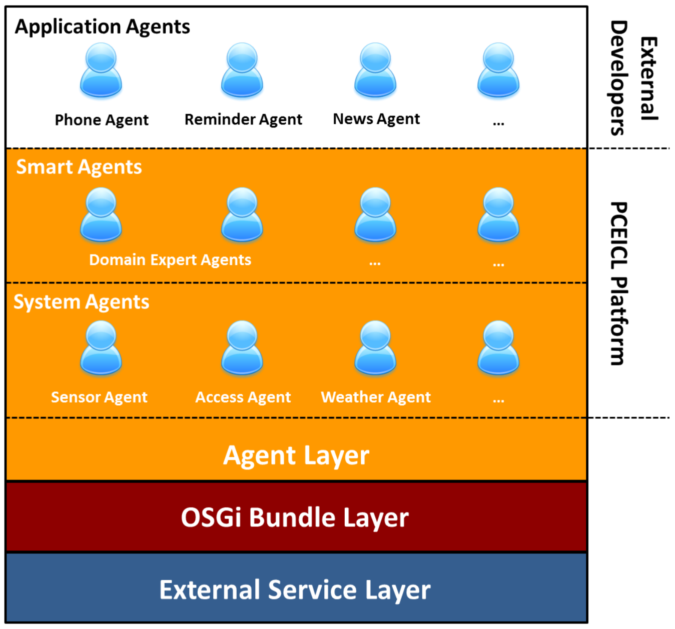

Fig. 3. Layers of the PCEICL Platform

C. User-Centered PCEICL Ontology Integration

Based on the PCEICL-Ontology [8], the user context can be semantically interpreted and can be used directly by the ACL for exchanging messages. This is used to adapt all services of the PCEICL platform to the needs of the user. The PCEICL ontology is, unlike in other approaches, user-centered. This means, that it models the user and his properties like personal information, interests, preferences, health condition but also the user's environment (social contacts or information about sensors, devices, weather, etc.). Additionally, the ontology is easy to expand and offers the possibility to have a historical view over the changing user data. To achieve access control of the PCEICL-Ontology only the *PCEICL-Ontology DB Agent* is allowed to access the stored information about the user.

V. Platform in the Cloud

PCEICL is a specialized platform, which delivers specific core functionalities in the field of Ambient Assisted Living as a service. It supports common home automation scenarios but also adds another layer of interaction by services and user interfaces to assist the user in his daily life.

Each environment is separated from other environments for privacy reasons and is managed by a *PCEICL PaaS Management System*. The *PCEICL PaaS Management System* has the following functions:

- *Multi-Tenant:* Multiple customers can use the PCEICL PaaS.
- *Customizable:* The PCEICL PaaS is customizable. For instance, if the system is used in a facility for assisted living, special devices need to be preconfigured during the installation of the PaaS. This is done by installing the needed bundles and services for these devices without the participation of the user. After this, the user can customize additional functionalities or install the services that assist him in his daily life. The user can proceed on his own or is helped by trained staff during the initial setup of the PCEICL platform.
- *Scalable:* The *PCEICL PaaS Management System* can also monitor the workload of the PCEICL PaaS instance and request additional resources at the *Cloud Management System* locally or remotely at the Public Cloud.
- *Granular Security Control:* The PCEICL PaaS offers a flexible access control system that allows detailed control over the user's data access and services access.

The private clouds share centralized services, hosted in the Public Cloud, like the aforementioned *Cloud Management System* or *Community Event Services*.

A. OSGi Bundle Repository

An *OSGi Bundle Repository* based on OBR [23] provides new services or updates existing services and agents (see Figure 4). In this example the Private Cloud is reporting a misbehavior or error inside a service/agent through a *Reporting Module*. This malfunction can be solved by an update of the affected services. After the update is released to the central *Bundle Repository*, the *PCEICL PaaS Management System* of each Private Cloud is notified that there is a new version of the service ready for deploying. The *PCEICL PaaS Management System* then triggers the update on the PCEICL platforms. The platforms evaluate if the update is part of a service, that is running in the platform, and if the bundle is still in use by an active service. If it is in the active bundle set but currently not in use, the PCEICL platform pulls the updated bundle from the Bundle Repository and deploys it. Otherwise the new bundle is installed alongside the existing and currently running bundle. When the deployment of the new bundle is finished, the old bundle is stopped and deleted and new requests will be processed by the new bundle. This update process is implemented to keep the availability as high as possible.

The central *Bundle Repository* is also accessed by the installation bundle for new services. The first time this is used, is during deployment of PCEICL by the *PCEICL PaaS Management System* and during setup of the individual PaaS instance through the primary user. Through the central *Bundle Repository*, all PaaS instances are on the same software version. This contributes to future development and maintainability, because the currently deployed bundles are the same across all PCEICL platforms and can be used as a basis for new bundles and services.

B. Reporting Module

Besides the already mentioned service bundle update reporting, the *Reporting Module* has an anonymized information interface. This interface is used to allow the improvement of software agent reasoning in the PCEICL platforms. For example, the suggestion of new services based on the installation setup of other PCEICL platforms can be realized. But, further ideas, like the collection of anonymized data about the user behavior, can help to improve the PCEICL services.

VI. Particularities of the PCEICL Platform

Based on the presented technology the PCEICL architecture is context aware and supports intelligent behavior. Especially the platform flexibility is worth mentioning and will be pointed out next.

- *Flexibility through Private Cloud:*
  New PCEICL service instances can be created or deleted on demand. This is relevant in settings of large facilities for assisted living that aim to keep their inhabitants living a self-determined life as long as possible. The data of the user (e.g., profile, configurations, etc.) are hosted locally.
- *Flexibility through PaaS:*
  Based on the load of the platform, it can react by requesting more resources through the PCEICL Management System and the attached monitoring. The PaaS is able to grow with the user's demand on assistance or services. Detailed access control is part of the platform.
- *Flexibility through Service Adaptability:*
  Each PCEICL platform can be set-up with different services and use-cases in mind. Home-automation scenarios based on OSGi are possible and can be extended with assistance systems to support the user in his daily life. The user has the tools to customize the user-experience by adding services or trying out new application offers. The system itself has the flexibility to support the user by responding to new requirements based on the behavior or health state of the user.
- *Flexibility through Functional Adaptability by Context Awareness:*
  The service functionality varies according to the context awareness [24] of the system. The system and its services have the ability to communicate with sensors in the living environment or with external services,

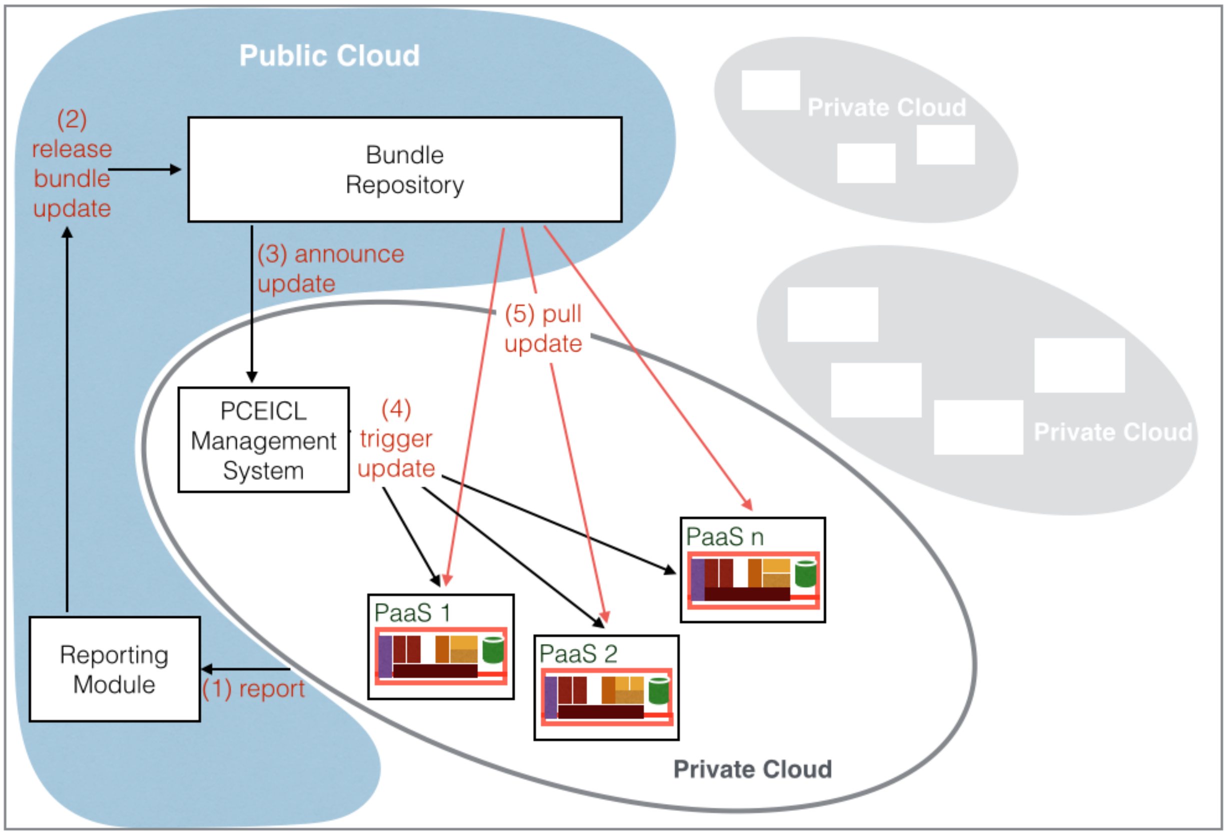

Fig. 4. Private clouds share centralized services in the Public Cloud. Example: Private Cloud updates bundles from the Public Cloud Bundle Repository based on the Reporting Module.

like, e.g., weather forecasts or calendars of events, to react with special features or services. It also has the ability to take the user's condition into account and adapt the services or to simply alert trained personnel for help.

PCEICL platform therefore uses the technology of agent systems to make the right decision at the right time. These lightweight and intelligent software agents can be updated and refined based on the anonymized data that can be retrieved by the Reporting Module.

VII. Evaluation

For a first evaluation of the architectural approach, we follow the Software Architecture Analysis Method (SAAM) [25] using the SAAM step: *Perform Scenario Evaluation*. The developed four scenarios try to evaluate the main features of the presented PCEICL architecture.

A. Scenario 1: Adding New Functionalities by the User

The direct scenario of adding new functionalities to a PCEICL instance by the user involves the private PaaS of the user and the *Bundle Repository*. The user has access to a central market place AAL OSGi bundle repository, that will list different applications, that can be added to the platform. The user chooses the new functionality. The *PCEICL PaaS Management System* requests for the new bundle at the *Bundle Repository*. This bundle requires a set of bundles. The platform evaluates, which required bundles are already running on it and which have to be added for the new functionality. If all required bundles and the bundle that contains the desired functionality, are deployed on the user's PCEICL platform, the functionality is started and ready for further configuration, e.g., specifying login details or changing color schemes based on the user's needs.

Involved PCEICL platform modules: *Bundle Repository*, *PCEICL PaaS Management System*

B. Scenario 2: Adding Resources to a PCEICL Instance

Adding resources to a PCEICL PaaS, when predicting performance shortcomings, is another direct scenario that is fully supported by the presented architecture. The *PCEICL PaaS Management System* continually monitors the platforms inside the private cloud. If an increase in demand for resources is measured or estimated, the *PCEICL PaaS Management System* can request additional resources, e.g., memory or CPU, at the *Cloud Management System*. The Cloud Management System is responsible for the IaaS layer and can then assign the resources to the platform.

Involved PCEICL platform modules: *PCEICL PaaS Management System*, *Cloud Management System*

C. Scenario 3: Updating a Bundle

As described in Section V and shown in Figure 4, the updating process of a bundle is supported by the architecture, which could imply a direct scenario. However, an updated bundle is changing functionality of the platform itself and is able to interact with other modules. This characterizes an indirect scenario in SAAM.

If a developer wants to update a bundle, he can use an instance of the PCEICL platform to test the functionality against it before introducing it in the Bundle Repository. Changes in bundles, which are only loosely coupled with other services, are simpler to update than closely coupled services. Changes in core services lead to a greater impact on other bundles and have to be treated very carefully. The difficulty of each change therefore has to be evaluated case by case. The modularity of the system however supports the process of updating bundles and the architecture can provide the aforementioned testing platform.

Involved PCEICL platform modules: *Bundle Repository*, *PCEICL PaaS Management System*

D. Scenario 4: Changing System Behavior Based on Context Changes of the User

This direct scenario describes the context awareness of the system using a simple *Ride Offering Service*. The user has added an event to his calendar that he wants to attend. After he had broken his leg in an accident, his health state in the PCEICL system (*PCEICL Ontology DB*) is updated by his doctor through the *PCEICL Ontology DB Agent*. The system reacts to this new state by offering a request for a lift to the event. This is also the adapted behavior for all new events that he would like to attend. If the health state changes again and the user is mobile again, the system can take back the changes and respond as before the accident.

Involved PCEICL platform modules: *PCEICL Ontology DB*, *PCEICL Ontology DB Agent*, (scenario specific: *Ride Offering Service*)

VIII. Conclusion and Future Work

In this paper, we presented an architecture approach for a context aware, intelligent and flexible PaaS for the use in Ambient Assisted Living.

The PCEICL platform is based on OSGi and extended by the JADE agent system. The agent system has access to the *PCEICL-Ontology* and reacts to changes by adapting the services, that are run on the platform. PCEICL is designed as a Platform as a Service, has a *PCEICL Management System* and runs in a private cloud for security and privacy reasons. The *PCEICL PaaS Management System* has the ability to manage the platform, preconfigure, start and stop services on demand, does load balancing and monitors the resources. Several common services, that are used by the PCEICL platform, run in the Public Cloud and can be accessed from all other platforms. The user has the ability to configure the platform to his needs and install new services from a central *Bundle Repository*. The Bundle Repository contains new software bundles, updates for existing bundles, special bundles with agents included, etc.

The paper discussed the improvement of flexibility by combining OSGi and cloud mechanisms. At all layers of the PCEICL architecture, this approach can improve the platform for the field of AAL.

Most of the data, that is stored in PCEICL Ontology DB, is only accessible by the user. Future work would be to refine the access of the data by the agents, based on a Role Based Access Control (RBAC), so that only services, that are trusted, get access to the data they need to deliver the service. These roles should be transparent to the users and developers of new services. One promising approach could be the integration of JADE-S [26] to dynamically assign permissions to agents.

The data privacy is also a matter for discussion, when collecting data by the *Reporting Module*. How can data be anonymous but still being significant across several private cloud infrastructures? This has to be considered, when trying to update services and agents to improve the experience of the platform.

Acknowledgment

The project ZAFH-AAL (“Zentrum für Angewandte Forschung an Hochschulen für Ambient Assisted Living”) is funded by the Ministry of Science, Research and the Arts of Baden-Württemberg, Germany. The funding program for the universities of applied science is called: Zukunftsoffensive IV “Innovation und Exzellenz” (ZO IV). The PCEICL project is a sub-project of the project ZAFH-AAL [1].

References

[1] “ZAFH-AAL - Zentrum für angewandte Forschung an Hochschulen für Ambient Assisted Living (Collaborative Center for Applied Research on Ambient Assisted Living),” http://www.zafh-aal.de, [retrieved: 2014.07.18].

[2] “PCEICL a Person Centered Environment for Information, Communication and Learning,” http://www.wolke.hs-furtwangen.de/currentprojects/pceicl, [retrieved: 2014.07.18].

[3] OSGi Alliance, “Smart home market,” http://www.osgi.org/Markets/SmartHome, [retrieved: 2014.08.02].

[4] M. Klein, A. Schmidt, and R. Lauer, “Ontology-centred design of an ambient middleware for assisted living: The case of soprano,” http://publications.andreas.schmidt.name/klein_schmidt_lauer_AIM-CU_KI07.pdf, [retrieved: 2014.07.11] 2007.

[5] R. Ram et al., “universaal: Provisioning platform for aal services,” in Ambient Intelligence - Software and Applications, ser. Advances in Intelligent Systems and Computing, A. Berlo, K. Hallenborg, J. M. C. Rodríguez, D. I. Tapia, and P. Novais, Eds. Springer International Publishing, 2013, vol. 219, pp. 105–112.

[6] Fraunhofer Institute for Open Communication Systems - FOKUS, “AAL-Kompetenz - The information portal for developers of intelligent assistance systems,” http://www.aal-kompetenz.de/, [retrieved: 2014.07.08].

[7] AALIANCE, Ambient Assisted Living Roadmap - AALIANCE Project - Deliverable 2.7, March 2010, ch. Enabling Technologies, pp. 95–96.

[8] C. Fredrich, H. Kuijs, and C. Reich, “An ontology for user profile modeling in the field of ambient assisted living,” in SERVICE COMPUTATION 2014, The Sixth International Conferences on Advanced Service Computing, A. Koschel and A. Zimmermann, Eds., vol. 5. IARIA, 2014, pp. 24–31.

[9] F. Bellifemine et al., “Java agent development framework,” http://jade.tilab.com/, [retrieved: 2014.07.12].

[10] ProSyst, “Smart home / smart energy,” http://www.prosyst.com/what-we-do/smart-home-smart-energy/products/, [retrieved: 2014.07.15].

[11] Miele & Cie. KG, "Miele@home," http://www.miele.de/haushalt/hausgeraetevernetzung-1912.htm, [retrieved: 2014.08.03].

[12] M. Petzold, K. Kersten, and V. Arnaudov, "Osgi-based e-health / assisted living," ProSyst, http://http://www.prosyst.com/fileadmin/ProSyst_Uploads/pdf_dateien/ProSyst_M2M_Healthcare_Whitepaper.pdf, Whitepaper, September 2013.

[13] H. Storf et al., "An event-driven approach to activity recognition in ambient assisted living," in AmI 2009, M. Tscheligi et al., Ed. Springer Verlag Heidelberg, 2009, pp. 123–132.

[14] I. Ayala, M. Amor, and L. Fuentes, "Self-starMAS: A multi-agent system for the self-management of AAL applications," in Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IEEE. IEEE International, 2012, pp. 901–906.

[15] K. Gunasekera, A. Zaslavsky, S. Krishnaswamy, and S. W. Loke, "VERSAG: Context-aware adaptive mobile agents for the semantic web," in COMPSAC '08. 32nd Annual. IEEE International, July 2008, pp. 521–552.

[16] K. Gunasekera, A. Zaslavsky, S. Krishnaswamy, and S. W. Loke, "Building ubiquitous computing applications using the VERSAG adaptive agent framework," Journal of Systems and Software, vol. 86, no. 2, pp. 501 – 519, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121212002695

[17] D. Carneiro, P. Novais, R. Costa, and J. Neves, "Developing intelligent environments with OSGi and JADE," in Artificial Intelligence in Theory and Practice III, ser. IFIP Advances in Information and Communication Technology, M. Bramer, Ed. Springer Berlin Heidelberg, 2010, vol. 331, pp. 174–183.

[18] P. Jaszczyk and D. Król, "Updatable multi-agent osgi architecture for smart home system," in Agent and Multi-Agent Systems: Technologies and Applications, ser. Lecture Notes in Computer Science, P. Jedrzejowicz, N. Nguyen, R. Howlet, and L. Jain, Eds. Springer Berlin Heidelberg, 2010, vol. 6071, pp. 370–379.

[19] "The foundation of intelligent physical agents," http://www.fipa.org/, [retrieved: 2014.07.08].

[20] "Agent communication language specifications," http://www.fipa.org/repository/aclspecs.html, [retrieved: 2014.07.12].

[21] E. Quarantotto and G. Caire, "JADE OSGi GUIDE," http://jade.tilab.com/doc/tutorials/JadeOsgiGuide.pdf, April 2010.

[22] The Eclipse Foundation, "equinox OSGi," http://www.eclipse.org/equinox/, [retrieved: 2014.07.14].

[23] W. J. Gédéon, OSGi and Apache Felix 3.0, 1st ed. Packt Publishing, November 2010, no. 978-1-84951-138-4, ch. Using the OSGi Bundle Repository.

[24] A. K. Dey and G. D. Abowd, "Towards a better understanding of context and context-awareness," in Computer Human Intraction 2000 Workshop on the What, Who, Where, When, Why and How of Context-Awareness, 2000, pp. 304–307.

[25] R. Kazman, L. Bass, G. Abowd, and M. Webb, "SAAM: A Method for Analyzing the Properties of Software Architectures," Software Engineering Institute, Carnegie Mellon University, White Paper, May 2007.

[26] JADE Board, "JADE Security Guide," http://jade.tilab.com/doc/tutorials/JADE_Security.pdf, February 2005.