Electronic Communications of the EASST
Volume 27 (2010)

Workshop über
Selbstorganisierende, adaptive, kontextsensitive
verteilte Systeme
(SAKS 2010)

A Middleware for Self-Organising Distributed Ambient Assisted Living Applications

Jan Schaefer

12 pages

Guest Editors: Klaus David, Michael Zapf
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

A Middleware for Self-Organising Distributed Ambient Assisted Living Applications

Jan Schaefer

Distributed Systems Lab
Hochschule RheinMain, University of Applied Sciences
Kurt-Schumacher-Ring 18, D-65197 Wiesbaden, Germany
jan.schaefer@hs-rm.de, http://wwwvs.cs.hs-rm.de

Abstract: This paper presents a middleware approach for self-organising distributed applications in the area of *Ambient Assisted Living* (AAL) as part of a home service platform. The middleware uses real time data processing techniques to identify complex events based on sensor input and service logs. This low-level and high-level information represents the platform's context model, which forms the basis for self-organisation properties: Controllers use context information to manage running services while adhering to strategies defined by users or inferred by the system itself.

Keywords: self-organisation adaptation context-awareness middleware AAL

1 Motivation

Flexible applications that are able to adapt their behaviour to an always changing environment at runtime are becoming more and more important. This need for runtime flexibility is caused predominantly by an increasing mobility of the devices executing applications (e.g. mobile devices) and growing user interest in applications that suit themselves to the current context (e.g. the user's location). Here, self-organising applications can increase the usability and the user experience and enable new application features and capabilities.

Contrary to static applications whose structure, features and behaviour are defined prior to execution, adaptive applications can react to external stimuli to support new or alternating capabilities without the need to restart the system. In addition, automated application adaptation reduces the need for manual intervention, which increases the manageability of adaptive systems profoundly. Especially in environments with a constantly changing number of interacting distributed software components running on devices using and providing services, a formalised adaptation process and an accompanying adaptation middleware can increase the benefit and overall stability of the system. Questions such as *"What functionality can the system provide right now using the currently available components?"* or *"Which additional components must I add to enable feature X?"* cannot be answered easily at the moment.

One field of research that heavily benefits from context-aware adaptive applications are service platforms for *Ambient Assisted Living* (AAL) environments. In recent years, several service platforms have been developed with different focuses. However, distributed computing and, especially, self-organisation properties such as self-management and structural adaptation have not been focused so far. Generally, it is either assumed that a technician installs and manages each component and the system as a whole, whenever the user likes to change something, or the platform consists of a fixed set of components provided by a single vendor (or a group of vendors) and cannot easily be changed. But being able to gradually build and extend their platform without being bound to a certain vendor and continuing services availability as resource availability fluctuates are major acceptance factors for users of such platforms, and as such these problems have to be addressed accordingly, if a service platform wants to succeed.

Instead of integrating the computation required for the decision to adapt into each application running on the platform, this computation has to be provided by the platform itself, more specifically its middleware. Apart from the already discussed manageability and platform stability aspects, platform-provided adaptation also enables new types of applications: applications (and user interfaces) that react to the current position of a mobile device and/or user, or applications that react to even more complex information that is made available by the platform (e.g. the detection of emergency situations derived from fused sensor data).

To be able to adapt to changes quickly, the service platform must enable applications to react automatically with minimal delay. As applications have to adapt to a changing execution environment (e.g. triggered by a change in location or resource availability), the service platform itself must incorporate self-management methods. However, user surveys have shown that they still want to be able to intervene and control the system themselves whenever they like.

In the face of these challenges, this paper proposes a middleware for self-managing adaptive distributed applications, whose application area will be AAL living environments. Section 2 shortly introduces the ongoing research related to Ambient Assisted Living. Section 3 analyses related AAL projects and feasible approaches for aspects of AAL systems subsequently. The proposed service platform architecture is presented in Section in Section 4. Although research and development for the architecture are still in an early phase, Section 5 presents the results of preliminary work of the Distributed Systems Lab. Section 6 completes the paper with an outlook on the next steps.

2 Ambient Assisted Living

The emerging demographic change towards an ageing population has begun to introduce drastic changes in the European society. Therefore, solutions have to be developed to motivate and assist older people to stay active for longer in the labour market, to prevent social isolation and to help people stay independent for as long as possible. *Information and Communication Technologies* (ICT) are designated to play a major role in helping to achieve these goals. It can help elderly individuals to improve their quality of life, stay healthier, live independently for longer, and counteract reduced capabilities which are more prevalent with age. ICT can enable them to remain active at work, in their community and at home. However, the problems cannot be solved by means of ICT alone. Thus, interdisciplinary work is required to identify the actual problems and to develop and evaluate solutions together with the involved parties. It is important to note that ICT technologies are supposed to work in the background, as many users simply will not be able to deal with the attached technical complexity. This is a major factor to ensure the acceptance of developed solutions. The research field dealing with the above presented problems is called *Ambient Assisted Living*[1] (AAL).

The European Ambient Assisted Living Innovation Alliance called *AALIANCE*, which is funded within the specific programme *Cooperation* and the research theme ICT of the 7th European Framework Programme, focuses on AAL solutions based on advanced ICT technologies for the areas of ageing at work, at home and in the society. In 2009, AALIANCE released an *Ambient Assisted Living Roadmap* [BCOW09], which presents a detailed overview into the prospected future of AAL topics, concepts and technologies until the year 2025. Of special importance for the work presented in this paper, this roadmap contains the *AAL Systems Composition Reference Architecture* shown in Figure 1, which details the common requirements for AAL systems.

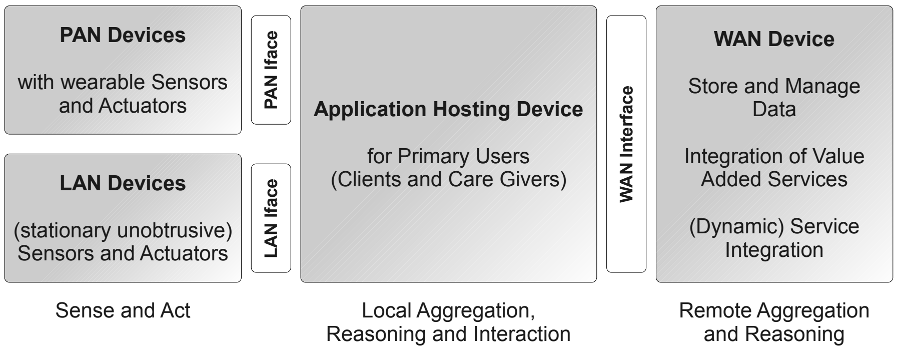
PAN Devices
with wearable Sensors and Actuators
LAN Devices
(stationary unobtrusive) Sensors and Actuators
PAN Iface
LAN Iface
Application Hosting Device
for Primary Users (Clients and Care Givers)
WAN Interface
WAN Device
Store and Manage Data
Integration of Value Added Services
(Dynamic) Service Integration
Sense and Act
Local Aggregation, Reasoning and Interaction
Remote Aggregation and Reasoning

Figure 1: AALIANCE Systems Composition Reference Architecture [BCOW09]

The reference architecture describes a three-layer networking approach to enable communication and connectivity between devices and services. It defines an AAL system consisting of seven component types:

- *PAN Devices* represent sensors and actuators that are either in-body, on-body or wearable and collect raw data. Some devices are powerful enough to process the collected data locally and aggregate collected data. Depending on its type, a device may be able to give basic feedback to the user in case of emergencies or failure (e.g. if the sensor cannot contact the application hosting device).
- The *PAN Interface* provides communication and data exchange facilities between PAN devices and one or more application hosting devices supporting dynamic configurations, which requires standardised discovery and communication protocols. Security and data safety are equally important as health care data may be transmitted between nodes. If devices are battery powered, they should give a timely warning before they run out of power.
- *LAN Devices* represent rather stationary devices, although they may be moved between rooms, quarters or public spaces. They also may be able to process data locally or collect data from multiple sensors. Like PAN devices, LAN devices should give user feedback, if they cannot contact the application hosting device.

[1] http://www.aal-europe.eu

- The *LAN Interface* is similar to the PAN interface, although energy consumption respectively power loss usually is a lesser issue for devices connected to power outlets.
- An *Application Hosting Device* (AHD) represents the central component of an AAL system communicating with PAN and LAN devices to collect sensor data and control other devices. It relays the sensor data to relevant services and processes their reaction. In addition, it stores data gathered from devices for processing and reasoning and provides a user interface for information and interaction of system users.
- The *WAN Interface* provides access to external (Internet) services for the AHD. It does not matter, whether this connection is established via (A)DSL, cable, mobile or (public) wireless networks. In public networks, encryption is especially important to protect private information.
- *WAN Services* are external services that are integrated into the AAL system remotely. They may, for example, provide teleconferencing or health monitoring services.

3 Related Work

Home service platforms have not only been objects of research in the wake of increasing AAL activities. For many years vendors have tried to develop hardware, software and services for home automation purposes that achieve broad acceptance in the consumer market (often called *Smart Home*). Several German and European projects targeting selected aspects described in Section 1 are presented subsequently.

Funded by the German Federal Ministry of Economics and Technology (BMWi), the *SerCHo*[2] (Service Centric Home) project was one of the first AAL system projects focusing on the development of a platform, models and tools for service development in the area of health care and home automation. As a result, a proprietary closed platform was developed, which assumes a completely outfitted apartment and is not planned for incremental growth.

The *MUSIC*[3] (Self-adapting applications for Mobile USers In ubiquitous Computing environments) project is funded as part of the 6^{th} European Framework Programme and primarily targets developers of modular, context-aware and adaptive mobile applications based on OSGi. For these, the project provides methods and tools that support the development of mobile applications. This approach is based on a model defining the applications adaptation behaviour, from which source code is generated. The MUSIC middleware is released as open source software. The methodology developed by the MUSIC project is highly relevant for this paper, although its focus lies on mobility rather than AAL.

Also funded as part of the 6^{th} European Framework Programme is the *SOPRANO*[4] project that aims to develop an affordable OSGi-based service platform, which promises ease of use for the elderly and support for social interactions and external health monitoring. The project plans to release its source code as open source software under the name *openAAL*[5].

[2] http://www.sercho.de
[3] http://www.ist-music.eu
[4] http://www.soprano-ip.org
[5] http://www.openaal.org

The German subproject *OSAMI-D*[6] of the European *ITEA 2* (Information Technology For European Advancement) project *OSAMI* aims to develop an OSGi-based service platform for IT and health care called *OSAMI Commons* that uses the *Devices Profile for Web Services* (DPWS) for sensor integration. The German research institute OFFIS, which is the primary developer of the service platform, is also pushing standardisation of interoperable AAL components within the German AAL research community.

Although several AAL and smart home projects featuring service platforms have been developed in recent years or are still under development, they do not focus on principles of self-organisation and coordinated distributed computing – especially self-management, adaptation and context-awareness. For example, [GPZ04] proposes an infrastructure for context-aware applications, which relies on a static application structure. The MUSIC project features several of the self-organising properties desirable for AAL systems, although it does not specifically target AAL environments. In addition to MUSIC, openAAL or OSAMI Commons might prove to be suitable building blocks for the implementation of the approach described in this paper, once the latter two become publicly available, as interoperability will likely become a major factor for acceptance and success of AAL service platforms.

4 Service Platform Architecture

The main goal of the service platform architecture presented in this paper is to enable development, deployment and execution of self-organising services in living environments consisting of distributed computing resources, which are connected by wired or wireless networks. The system features a constantly updating information model, which is based on system (e.g. logs) and environment (e.g. sensors) data gathered at runtime. The model relies on formal, ontology-based descriptions [HWDC09] of users, devices, services and their relationships. Although the system has to cope with potentially high dynamics caused by tens or hundreds of interacting nodes ranging from small sensors to PC-like workstations that can connect to or disconnect at any time, the service platform has to maintain a stable execution environment for executed services.

4.1 System Level

To be able to connect suitable communication partners with each other automatically (e.g. cameras producing video and TVs consuming video), this approach relies on formal descriptions of capabilities and dependencies in the form of ontologies provided by each device and service. Ideally, this description is aligned with the platform's core ontologies, which defines the basic terminology and relationships of relevant entities. The supplied descriptions are merged with the core ontologies and stored in an *Ontology Repository*, which can be queried by services.

Once a device has connected to the platform or a service has been deployed and provided its description, the platform stores the relevant mediation information in the repository and uses it to find appropriate communication partners henceforward. In addition, if a service specifies *Quality of Service* (QoS) requirements for potential partners, the platform also regards their performance guarantees and history, before coupling service consumer and provider.

[6] http://www.osami-commons.org

Figure 2 illustrates the elements of the service platform. Basically, the service platform runs arbitrary services while processing events according to the preferences of its user – the person or persons currently staying in the living quarter, in which the system is deployed. The *Activities* can consist of automatic services, which manage available devices like lights, shutters or heating based on the users' preferences, and of services interacting with users like a medicine reminder or teleconferencing application. Each activity and each service can introduce new capabilities to the platform (described by ontologies), which can be shared between activities and services.

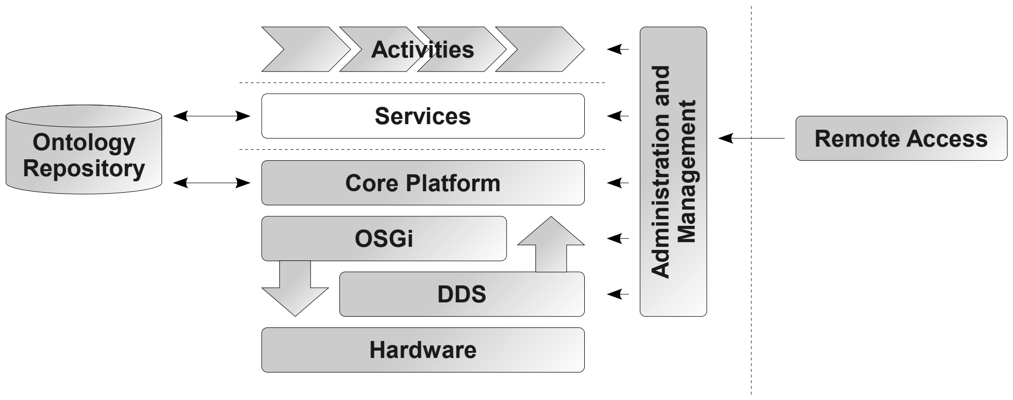

Figure 2: System Level Overview

Depending on their computing capabilities, devices (e.g. hardware sensors) are integrated into the system using either *OSGi* [OSG09] standardised by the OSGi Alliance (including *OSGi Remote Services* for discovery and remote service access) and the less resource-demanding *Data Distribution Service* (DDS) [Obj07] by the Object Management Group. Both middleware technologies are used to integrate devices into the platform to set the foundation for the *Core Platform*, which provides basic facilities required by services:

- A *Service Registry*, which relies on semantic descriptions and performance data for its mediation
- Access to the *Ontology Repository*, which allows services to query the context model
- *Security* features (e.g. user authentication and code signing) to ensure data safety and platform integrity
- *Performance Instrumentation* capabilities to enable reporting and claiming runtime performance characteristics

Although the platform employs several self-organisation properties (which are described in the subsequent Section 4.2), it also contains a subsystem for manual *Administration and Management*, which allows defining user preferences and analysing, tracing and overriding decisions the system reached autonomically. The management subsystem also supports enabling performance instrumentation for selected services to gain a deeper insight of currently active services. The subsystem can be accessed locally by the system's users or remotely by, for example, authorised service partners. In addition, it is able to list the ontologies stored in the ontology repository and to display the context model, so that administrators and tech-savvy users can analyse the current state and capabilities of the system. This knowledge is vital for adding new devices and services to extend the system's functionality.

4.2 Service Level

The self-organisation properties of the service platform heavily depend on its service layer, as this layer contains the *Data Processing* and *Control* subsystems. Because of its importance, the service layer is illustrated in further detail in Figure 3 (cf. *Services* in Figure 2).

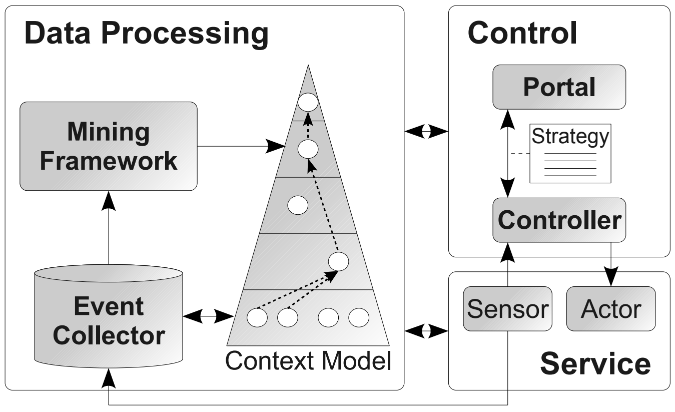

Figure 3: Service Level Overview

The *Data Processing* subsystem is responsible for collecting and processing events gathered from other services at runtime (e.g. instrumentation logs and sensor data). It consists of three components: the *Event Collector*, the *Context Model* and the *Mining Framework*. The first merely collects, sorts and categorises the gathered information and, thus, is not considered here any further.

The platform supports data processing modules to derive complex, higher level context information from low-level events (e.g. to detect emergency or failure situations). To achieve this, different approaches for data analysis are integrated into the service platform: In addition to existing *Complex Event Processing* [LF98] (CEP) engines like *ESPER* [Esp09, Hit10], custom *Data Stream Mining* (DSM) [GZK05] and knowledge reasoning approaches are used to process incoming information. To integrate the different analysis approaches, the event processing subsystem features a *Mining Framework*, into which additional processing modules can be plugged.

The raw logging and event data and the processed complex, higher level information are provided to services as an ontology-based *Context Model*, which is partitioned into different layers of ontologies depending on the quality of the information: Raw output from integrated small hardware sensors supplies the lowest layer of what is also referred to as the *Context Pyramid* information model. The second to lowest layer already requires processing of some sort (e.g.

statistical calculations); Information on higher layers may require elaborate processing (e.g. multiple sensor data fusion). But, hardware sensors do not necessarily deliver low-level information only. Specialised appliances might provide already processed information to a higher layer of the context model (e.g. an optical tracker consisting of several cameras may deliver coordinates of a user's location instead of a mere collection of single tracks). The higher in the pyramid an information element is placed (the more abstract it is), the less likely it can be associated directly to a specific service or hardware sensor output. Services can subscribe to the elements of any layer in the context pyramid and are notified in case of changes caused by events.

The more complex information contained in the context pyramid is primarily accessed by the *Control* subsystem, which features a *Portal* and *Controllers*. The portal provides *Strategies*, which can consist of global goals affecting all devices connected to the platform (e.g. *"Minimise energy consumption"*), and local goals only affecting a subset of devices at a time (e.g. *"Always illuminate the room user X is in"*). Further details on the control subsystem and its capabilities are presented in Section 4.4.

A controller bundles services and devices with common functionality trying to meet strategy's requirements. For example, a strategy may concern a light source controller able to control all light sources in a room. For this controller, the current locations of persons in the room are important as well as their individual illumination preferences.

4.3 Horizontal and Vertical Integration

The complexity of controlling the service platform and securing data integrity is increased by the services' and devices' distribution throughout the users' living environment and beyond. Thus, not every subsystem and its services has to be replicated to every device. Rather, depending on the computing power of a device, services can be executed either locally on devices, or devices can consume necessary services remotely from peers. Especially event processing, being computing-intensive, is provided by powerful devices, and the resulting context information is merely replicated horizontally to other nodes, which subscribed to the information, using DDS.

An AAL system consists of devices with highly heterogeneous features, especially regarding computing power, power consumption and network connectivity. Thus, not all devices can be integrated into the service platform in the same way, and not all devices will be connected to the Internet directly. Using both OSGi and DDS allows supporting a much broader range of devices, their deployment and vertical integration into the system. Adoption of OSGi and related technologies like *Universal Plug and Play* (UPnP) or *Devices Profile for Web Services* (DPWS) by technology vendors in smart homes has increased significantly, so that more and more devices can be integrated as OSGi services easily. In addition, DDS is used to enable communication between small sensors (or even sensor networks) and high-level services and devices. The details of the DDS subsystem are not in the focus of this paper and, therefore, not presented further.

4.4 Self-Organisation Aspects

Whenever a service connects to the platform for the first time, it provides an ontology describing its capabilities and public interfaces offered to other services. If the new service enables new platform capabilities in combination with existing services and devices, the new functionality is enabled subsequently, which may require reconfiguration of existing services. New services also must provide management interfaces, which can be used by controllers to set up and manage the service at runtime, thus forming a control loop. For validation purposes, sensor feedback allows controllers to evaluate the effectiveness of their decisions.

With regard to OSGi, the application adaptation is not merely limited to service starts and stops. Services might even be moved between containers (structural adaptation), in case of hardware or power failure, or if OSGi bundles have to relocate to a mobile (e.g. before a user leaves the living quarter). Especially mobile usage of services provides a wide range of adaptation scenarios, as the actions a user wants to take with his mobile can greatly differ between being at home and abroad.

The portal provides strategies representing user preferences (*"User X likes a room temperature of 22°C"*) or resulting from machine learning approaches used on runtime data. The latter, however, is not within the focus of this work). This setup can lead to conflicts of interest (e.g. between a user's will and system stability), which have to be resolved automatically (e.g. through strategy prioritisation).

The main purpose of strategies is, however, to define a set of policies for controllers, which enforce the instructions concerning their area of influence (e.g. rules or parameters). For this, controllers do not have to be connected to the portal permanently. If they do not receive strategy or policy updates, they continue their current course of operation. Similar to the portal, controllers also have to deal with conflicting strategies, which have to be resolved (e.g. if users with contradicting preferences are in the same room). Controllers are also responsible for monitoring changes in the systems structure (i.e., service or device availability), which may affect it or its controlled services. In case a relevant change occurs, it has to trigger the affected services' adaptation (e.g. by activation or deactivation of interfaces).

In addition to strategies, the portal may set *Application Modes* (e.g. *Emergency*, *Night*, *Party*, *Absence*), which trigger an adaptation in active services and influence how controllers and services react to sensorial input, if services know these modes and have the ability to adapt their behaviour (i.e., react differently for each mode). For example, the event *"A window has been opened."* does usually not pose a problem generally. If, however, the current application mode was set to "Absence" (i.e., no one is supposed to be home), this event could be an indication for an intrusion attempt. If a service does not know an announced application mode, it simply ignores the switch command.

5 Preliminary Work

The development of the Ambient Assisted Living service platform is still in an early phase, but fundamental work on several aspects has already begun. The results from the work presented subsequently are used in the development of the AAL system.

5.1 Data Processing

This section presents three master thesises and a diploma thesis, which were completed in the Distributed Systems Lab.

[Mar09] changes application modes (cf.4.4) based on changing context information. Using a finite state machine, applications can adapt their behaviour to context changes by switching rule sets. In addition, the adaptation is influenced by the new and previous application modes. This work is based on MUSIC.

[Ign09, Mol09, Gro09] developed three IT management approaches using data stream mining based on UML models, OWL ontologies and a Bayesian network respectively. The first approach uses *Object Constraint Language* (OCL) to annotate UML models with performance constraints. Using the previously developed *Constraint Monitor* [TSSK09], application logs are processed and workflows reconstructed. If a workflow violates its constraints, an appropriate notification is generated. The second approach is based on OWL ontologies in combination with SWRL rules. Here, facts are extracted from application logs and update the knowledge base. The last approach uses log data to parametrise a Bayesian network. Afterwards, the network analyses log data to detect imminent failures.

5.2 OSGi Management

Being able to manage (e.g. monitor, analyse and configure) an application at runtime greatly eases the development process, especially when dealing with a potentially high number of distributed interacting nodes. Thus, two student projects established a basis for generic, application-independent management and performance instrumentation of OSGi-based applications.

The developed prototypical management solution enables remote access to an OSGi platform and all deployed OSGi bundles using *Java Management Extensions* (JMX). To achieve this, all JMX-enabled bundles register themselves with a local JMX agent, which can be accessed remotely using an JMX connector. This solution is used to dynamically alter the configuration (e.g. attributes) of JMX-instrumented bundles.

The developed prototypical instrumentation solution relies on *Aspect-Oriented Programming* (AOP) to weave instrumentation code into OSGi bundles. The instrumentation, which adds logging, call counter and execution time measurement capabilities, contains switches to enable or disable parts of the instrumentation dynamically at runtime using the management solution presented in the previous paragraph.

5.3 AAL System Demonstration Environment

OSGi integrations for several home appliances have been developed in the Distributed Systems Lab at Hochschule RheinMain, so that they can be used to test, develop and demonstrate selected details of the distributed AAL service platform. The integrated devices include:

- *IP Power Outlets*, which can be switched and queried
- A *Wireless Camera*, which can be rotated and delivers video, audio and pictures
- Wired *Door Opening Sensors*, which can be queried for their status

So far, the IP power outlets are mostly used to control light sources, which typically only require the on/off-states. The device services can be accessed via Web Service interfaces. In addition, mobile clients for Google Android and Windows Mobile have been developed, which offer graphical user interfaces for device interaction.

5.4 Study of Acceptance of Mobile Devices by The Young Elderly

In a joint study between the Distributed Systems Lab and the Social Welfare Department at RheinMain University, two workshops with two groups of men and women between the age 50 and 65 – the so-called *Young Elderly* – were held in May and June 2009. During each workshop, requirements, interests and fears of the participants were collected and discussed. This input was used to develop a small prototypical application based on the integrated appliances presented in Section 5.3, which were presented to the participants during another workshop in October 2009.

The demonstration featured a complex activity, in which the system is configured to monitor all door opening sensors. Whenever one of the sensors is activated (a door is opened), the camera turns to the respective door, takes a picture and sends it to a selected mobile. This demonstration was well-received by all workshop participants.

The summarising report for this study is pending.

6 Next Steps and Open Issues

This paper presents a middleware approach for self-organising distributed applications in the area of Ambient Assisted Living. As presented in Section 5, the current work focuses on enabling features such as data processing for the construction of the layered context model. The next step will concentrate on context subscription and evaluation by services and further self-organising properties such as approaches for the structural adaptation of distributed OSGi applications. With regard to the implementation, the evaluation of existing service platforms will continue focusing on candidates supporting the integration of the approach presented here.

This paper has not discussed how new devices or services are integrated into the AAL system, nor how the describing ontologies are defined as these topics have not been within the focus of this paper. Development and initial deployment of an AAL system with its devices and services present a problem by itself, which will be addressed at a later stage.

The creation of communication groups (e.g. between controllers and services) is another topic not covered in this paper. Tightly related to this area of research are feasible approaches for resolving conflicts between strategies and controllers, which have yet to be dealt with.

Bibliography

[BCOW09] G. van den Broek, F. Cavallo, L. Odetti, C. Wehrmann (eds.). *AALIANCE Ambient Assisted Living Roadmap*. VDI/VDE-IT AALIANCE Office, Berlin, Germany, September 2009.
http://www.aaliance.eu/public/documents/aaliance-roadmap/

[Esp09] EsperTech Inc. ESPER - An Event Stream Processing and Event Correlation Engine (Version 3.3.0). December 2009.
http://esper.codehaus.org

[GPZ04] T. Gu, H. K. Pung, D. Q. Zhang. Toward an OSGi-Based Infrastructure for Context-Aware Applications. *IEEE Pervasive Computing* 3(4):66–74, October 2004.
http://dx.doi.org/10.1109/MPRV.2004.19

[Gro09] P. Großmann. Klassifizierung von Laufzeit-Zuständen kritischer Anwendungen mit Hilfe von Bayes-Netzen [German]. Master's thesis, RheinMain University of Applied Sciences, November 2009.

[GZK05] M. M. Gaber, A. Zaslavsky, S. Krishnaswamy. Mining data streams: a review. *SIGMOD Rec.* 34(2):18–26, June 2005.
http://dx.doi.org/10.1145/1083784.1083789

[Hit10] J. Hitchings. Flexible Log Monitoring with Scribe, Esper, and Nagios. January 2010.
http://eng.kaching.com/2010/01/flexible-log-monitoring-with-scribe.html

[HWDC09] M. Hadzic, P. Wongthongtham, T. Dillon, E. Chang. *Introduction to Ontology*. Studies in Computational Intelligence 219, pp. 37–60. 2009.
http://dx.doi.org/10.1007/978-3-642-01904-3_3

[Ign09] A. Ignat. Ein Data-Stream-Mining-Ansatz zum proaktiven Management von kritischen IT-Anwendungen [German]. Master's thesis, RheinMain University of Applied Sciences, November 2009.

[LF98] D. C. Luckham, B. Frasca. Complex Event Processing in Distributed Systems. Technical report, 1998.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.56.876

[Mar09] B. A. Marinescu. Eine Architektur für kontextsensitive und adaptive verteilte Anwendungen in AAL-Umgebungen [German]. Master's thesis, RheinMain University of Applied Sciences, October 2009.

[Mol09] G. Moldovan. Proaktives Management einer kritischen IT-Anwendung unter Nutzung von Semantic Web-Ansätzen [German]. Master's thesis, RheinMain University of Applied Sciences, November 2009.

[Obj07] Object Management Group. Data Distribution Service for Real-time Systems (Version 1.2). January 2007.
http://www.omg.org/technology/documents/formal/data_distribution.htm

[OSG09] OSGi Alliance. OSGi Service Platform Release 4 (Version 4.2) Core Specification. May 2009.
http://www.osgi.org/Download/Release4V42

[TSSK09] A. Textor, M. Schmid, J. Schaefer, R. Kroeger. SOA Monitoring Based on a Formal Workflow Model with Constraints. In *Proceedings of the QUASOSS'09 Workshop co-located with the ESEC-FSE'09 Conference*. Pp. 47–53. ACM, Amsterdam, The Netherlands, August 2009.