Technologieunabhängige Geräteintegration des OSAmI-Projekts
Technology independent device integration of the OSAmI project

Torsten Gorath[1], Marco Eichelberg[1], Andreas Hein[1], Elmar Zeeb[2], Frank Golatowski[2] und Dirk Timmermann[2]
[1] OFFIS – Institut für Informatik, Escherweg 2, 26121 Oldenburg, Deutschland,
{torsten.gorath, marco.eichelberg, andreas.hein}@offis.de
[2] Institut für Angewandte Mikroelektronik und Datentechnik, Universität Rostock, 18051 Rostock, Deutschland,
{elmar.zeeb, frank.golatowski, dirk.timmermann}@uni-rostock.de

Kurzfassung

Eine zentrale Komponente vieler AAL-Systeme ist die Anbindung unterschiedlichster Sensoren und Aktuatoren an eine Plattform, auf der die Daten zusammengeführt und gemeinsam ausgewertet werden. Dabei kommen unterschiedlichste Geräte aus den Bereichen Gebäudeautomation und Medizintechnik zum Einsatz. Die Anbindung der Komponenten erfolgt drahtgebunden oder drahtlos, und häufig verwendet derselbe Gerätetyp von unterschiedlichen Herstellern unterschiedliche Protokolle oder inkompatible Datenformate. Dies verursacht bei der Entwicklung und Pflege von AAL-Systemen einen erheblichen Aufwand für die Geräteintegration. Im Rahmen des Projekts OSAmI (Open Source Ambient Intelligence) wird eine flexible Diensteinfrastruktur geschaffen, welche für unterschiedlichste Anwendungen in den Bereichen AAL und Ambient Intelligence eingesetzt werden kann. Eine Kernkomponente dieser Diensteinfrastruktur ist die in diesem Beitrag beschriebene Lösung für eine technologieneutrale Geräteintegration in einem OSGi-Framework.

Abstract

A central component of many AAL systems is the integration of various sensors and actuators into a platform where data is collected and evaluated as a whole. For this purpose different devices from home automation and medical device technology are used. Connections are wired or wireless, and frequently the same type of device is available from different vendors with different, incompatible protocols and data formats. This causes a significant effort for the development and maintenance of the device integration in AAL systems. In the framework of the OSAmI (Open Source Ambient Intelligence) project a flexible service infrastructure is being developed that can be used for a wide range of applications in AAL and ambient intelligence. A core component of this service infrastructure is the approach for a technology neutral device integration into an OSGi framework described in this article.

1 Einleitung

Die Anbindung von Geräten, Sensoren und Aktoren ist für die meisten AAL-Systeme von zentraler Bedeutung und wesentlich für den Erfolg eines AAL-Systems. Neben medizinischen Geräten, Sensoren und Aktoren, wie z.B. EKG, EEG oder auch Ergometer, werden auch Geräte aus vielen anderen Bereichen, wie z.B. der Gebäudeautomatisierung, dem Mobilfunk, der Konsumgüter-Elektronik oder dem IT-Sektor, eingebunden. Die Anbindung der Komponenten erfolgt über drahtgebundene (USB, serielle Schnittstelle) oder drahtlose (z. B. Bluetooth, Zigbee) Übertragungstechnologien.

Einerseits verwenden dieselben Gerätetypen, die von unterschiedlichen Herstellern angeboten werden, häufig inkompatible Protokolle und Datenformate. Wenn das EKG des einen Herstellers in eine Anwendung eingebunden ist, kann man dieses zumeist nicht ohne großen Anpassungsaufwand durch ein EKG eines anderen Herstellers austauschen. Andererseits gibt es auch das Phänomen, dass, obwohl Protokolle und Datenformate kompatibel sind, unterschiedliche Übertragungstechnologien einer einheitlichen Einbindung in das AAL System entgegenstehen.

Beide Fälle haben bei einer direkten Anbindung der Geräte zur Folge, dass das AAL-System abhängig vom verwendeten Gerätetyp ist und der Austausch eines Gerätes nur gegen ein baugleiches erfolgen kann. Für den Einsatz eines gleichartigen Gerätes eines anderen Herstellers ist eine Anpassung notwendig. Anpassungen sind auch dann notwendig, wenn ein Gerät durch ein anderes vom gleichen Hersteller ersetzt wird, wenn dieses eine andere Übertragungstechnologie verwendet.

Dies alles verursacht bei der Entwicklung und Pflege von AAL-Systemen einen erheblichen Aufwand bei der Geräteintegration und widerspricht modernen IT-Architekturkonzepten wie der losen Kopplung. Die lose Kopplung ist ein Konzept aus den service-orientierten Architekturen und hat zum Ziel, dass Änderungen in Teilkomponenten keine Änderungen im Gesamtsystem nach sich ziehen.

In diesem Beitrag wird ausgehend von der Beschreibung bestehender Lösungen ein neuer Ansatz zur Geräteintegration beschrieben. Dieser erweitert die *OSGi Device Access* Spezifikation um die Fähigkeit, Anwendungen, die Geräte einbinden, von den verwendeten Übertragungstechnologien zu entkoppeln. Das ist eine grundlegende Verbesserung mit deren Hilfe die Entwicklung von AAL-Systemen bedeutend vereinfacht wird.

2 Andere Ansätze

Feldbusse für die Gebäudeautomation wie LON (Local Operating Network, DIN EN 14908), KNX (DIN EN 50090) oder BACnet (Building Automation and Control Network, DIN EN ISO 16484) definieren jeweils eine herstellerunabhängige Möglichkeit, um Sensoren und Aktoren anzubinden [1]. Meistens werden sogenannte Geräteprofile definiert, die die Funktion einer Geräteklasse beschreiben. Alle Geräte, die ein solches Profil unterstützen, können beliebig ausgetauscht werden. Dieser Ansatz ist jedoch auf die jeweiligen Feldbusse als Netzwerktechnologie sowie auf die im jeweiligen Standard definierten Geräteklassen eingeschränkt.

Diese Problematik wird im Eclipse-Projekt SODA [2] adressiert. Das dort bereitgestellte Framework stellt Implementierungen für unterschiedliche Kommunikationsprotokolle (beispielsweise eine serielle Schnittstelle) bereit. Die Architektur sieht vor, dass der Entwickler fünf Schichten zu implementieren hat. Dabei ist jedoch wichtig, den Implementierungsaufwand zwischen dem Geräteherstellers und dem Anwendungs- oder Dienstanbieter aufzuteilen. Zwischen diesen Schichten werden Profile definiert, welche die gleichbleibenden Schnittstellen darstellen. Dieser Ansatz ist hersteller- und technologieunabhängig und steht als OSGi-Implementierung (Open Service Gateway Initiative) bereit.

3 Eigener Ansatz

Im Rahmen des ITEA2-Projektes OSAmI (Open Source Ambient Intelligence) [3,4] wird eine flexible und offene OSGi-basierte Diensteinfrastruktur geschaffen, welche für Anwendungen im Bereich Ambient Intelligence eingesetzt werden kann.

Das vom BMBF geförderte deutsche Teilprojekt hat den Anwendungsbereich Telemonitoring und häusliche Pflege. Es nimmt sich der Herausforderung an, AAL Systeme auf kleinen, kostengünstigen Plattformen in das häusliche Umfeld zu bringen. Eine Komponente, welche in diesem Teilprojekt entwickelt wird, ist die OSAmI Geräteintegration als eine schlanke, erweiterbare, herstellerunabhängige und auf OSGi basierende Lösung.

3.1 Grundlagen

Das OSAmI-Projekt hat zum Ziel, vorhandene Standards zu nutzen, zu erweitern oder dort, wo es noch keine Lösung gibt, neue zu schaffen. Dem entsprechend ist die OSAmI Geräteintegration als Erweiterung der *OSGi Device Access* Spezifikation [5, 6] konzipiert worden. In dieser Spezifikation ist definiert, wie Geräte im OSGi-Umfeld automatisch eingebunden werden können. Abbildung 1 verdeutlicht dieses Vorgehen und stellt zusätzlich die Erweiterungen für die OSAmI Geräteintegration dar.

Entsprechend der *OSGi Devices Access* Spezifikation werden neue im Netzwerk sichtbare Geräte vom *Base Driver* erkannt. Dieser erzeugt eine Instanz eines *OSGi Device Service*. Der *Device Manager* registriert das neue Gerät

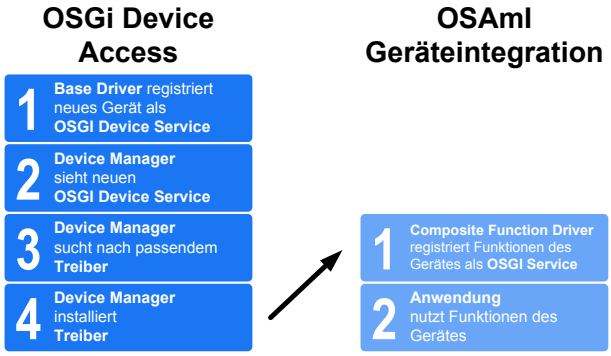

Abbildung 1: Sequenzen für das Hinzufügen und Entfernen von Geräten

mit Hilfe des *Whiteboard Pattern* [7] und versucht, den besten Treiber für dieses Gerät zu identifizieren und zu installieren. Jedes physische Gerät wird durch einen oder mehrere OSGi-Services wiedergespiegelt, welche die Interface-Spezifikation für Devices implementieren. Die Einbindung dieser Devices erfolgt über Gerätetreiber.

Die OSGi Device Access Spezifikation definiert mehrere Gerätetreiberarten, die um zwei neue Arten erweitert werden: der *Discovery Driver* und der *Composite Function Driver*.

Der *Discovery Driver* ist eine Erweiterung des *Base Driver*, der benötigt wird, um spezielle Netzwerktechnologien wie etwa Bluetooth in vollem Umfang einzubinden. Wie in Abbildung 2 dargestellt ist, ermittelt der *Composite Function Driver* die Funktionen des Gerätes (die ihm die Technologie zur Verfügung stellt) und installiert mehrere zugehörige *Function Interfaces*. Ein *Function Interface* bindet eine spezielle Funktion eines Gerätes in das *OSGi Framework* in Form von *OSGi Services* ein.

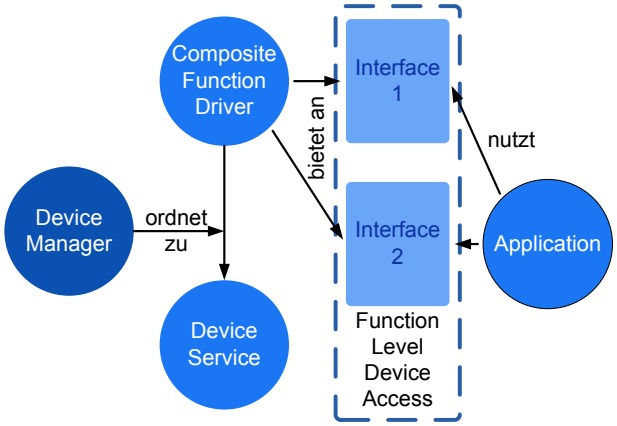

Abbildung 2: Zusammenhänge in der OSAmI Geräteintegration

Die *Device Access* Spezifikation definiert einen *Device Manager*, welcher von jeder OSAmI Geräteintegration realisiert werden muss. In der OSAmI Geräteintegration ordnet der *Device Manager* neuen Geräten den *Composite Function Driver* zu. Danach können die Funktionen, die ein Gerät anbietet im *OSGi Framework* verwendet werden. Somit ist es möglich, innerhalb von OSGi unabhängig von der Übertragungstechnologie auf die Funktionen von Geräten zuzugreifen.

3.2 Anwendungssicht

Die OSAmI Geräteintegration erweitert die OSGi Device Access Spezifikation, da diese nicht regelt, wie Anwendungen auf Gerätetreiber zugreifen. Das ist aber Voraussetzung dafür, die Anwendungen von den Gerätereibern zu entkoppeln, und wichtig für die herstellerunabhängige Einbindung von Geräten.

Durch die Festlegung von Schnittstellen für Gerätetreiber ist es möglich, dass Anwendungen unabhängig von den Gerätetreibern entwickelt werden können. Das hat den entscheidenden Vorteil, dass die Entwicklung von neuen Systemen beschleunigt werden kann, da es keine direkten Abhängigkeiten zwischen dem Gerätetreiber und der Anwendung mehr gibt. Somit können diese Komponenten gleichzeitig entwickelt werden. Hersteller von Anwendungen können in der Entwicklungsphase evtl. auf bereits vorhandene Geräte und Treiber zurückgreifen.

Entscheidend für die Verwendung von Geräten unabhängig von konkreten Gerätetreibern ist das Vorhandensein einer allgemeingültigen Definition von Schnittstellen (API) für unterschiedliche Dienste von Geräten. Im Rahmen des deutschen OSAmI-Teilprojekts werden solche Schnittstellen für Elektrokardiogramm (EKG), Blutdruck, Herzfrequenz, Blutsauerstoffsättigung und ein Fahrradergometer definiert. Weitere Schnittstellen können definiert werden, wobei es jedoch wichtig ist, dass diese Definitionen allen Interessierten frei zugänglich sind. Deshalb wird das OSAmI-Projekt vordefinierte Schnittstellen für Geräte veröffentlichen, die in den Bereichen Telemonitoring und häusliche Pflege zum Einsatz kommen.

3.3 Treibersicht

Die Treiber werden in zwei Teile aufgeteilt, was die Wiederverwendbarkeit fördert und somit zur Kostensenkung beiträgt. Es wird zum einen die Kommunikation auf unterer Abstraktionsschicht implementiert. Auf dieser Ebene gibt es noch keine Services, es werden lediglich Daten ausgetauscht. Das spezifische Protokoll und damit auch ggf. proprietäre Datenstrukturdefinitionen des Herstellers werden in einem *Composite Function Driver* realisiert. In Abbildung 3 ist diese Aufteilung in die unterschiedlichen Komponenten dargestellt. Bei dem zu erstellenden *Discovery Driver* sind in der OSAmI Geräteintegration folgende Erweiterungen vorgenommen worden: Da viele ältere Übertragungstechnologien kein automatisches Discovery von Geräten unterstützen, ist der OSGi-Basedriver zu einem *Discovery Driver* erweitert worden. Der Discovery Driver stellt Methoden für das manuelle Verbinden oder Trennen von Geräten mit bzw. von dem System bereit. Diese Methoden können auch verwendet werden, um den Discovery-Prozess manuell zu starten. Somit ist es möglich, die Mechanismen der Device-Access-Spezifikation von OSGi beispielsweise für Geräte mit einer RS232-Schnittstelle zu verwenden.

Da der *Discovery Driver* unabhängig von den Geräten bereitgestellt werden kann, ist es nicht notwendig, diesen immer zusammen mit neuen Geräten zu entwickeln. Vielmehr müssen diese Treiber nur einmal für eine Netzwerktechnologie (Bluetooth, Zigbee, RS 232 usw.) bereitgestellt werden. Dieses steigert zum einen die Effizienz bei der Entwicklung von Treibern, da große Teile wiederverwendet werden können, zum anderen werden Fehlerquellen vermieden bzw. reduziert.

Jedes Gerät kann einem oder mehreren Diensten zugeordnet werden. Die Dienste werden in den funktionalen Treibern (*Composite Function Driver*) von Geräteherstellern implementiert. Die Hersteller besitzen natürlich sehr genaue Kenntnisse der jeweiligen Geräte sowie des verwendeten Protokolls und können somit eine schlanke und performante Implementierung realisieren. Neben dem rein funktionalen Teil ist für jeden funktionalen Treiber das *Driver Interface* der *OSGi Device Access* Spezifikation zu

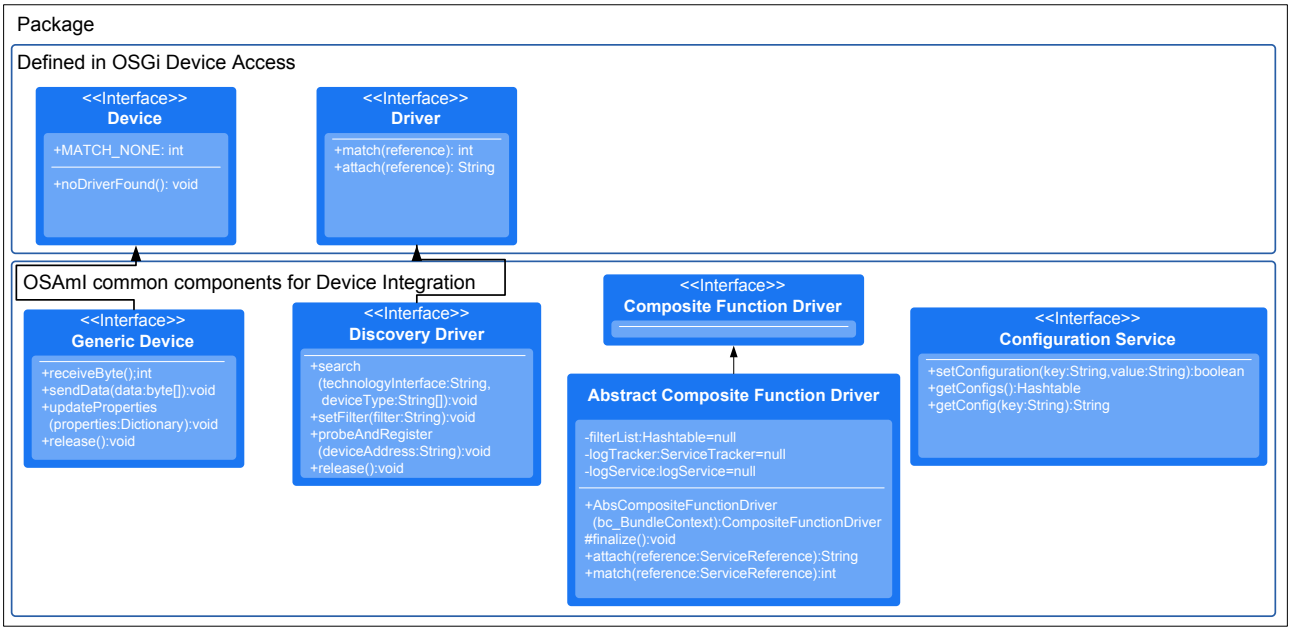

Abbildung 3: Klassendiagramm der OSAmI Geräteintegration

realisieren. Dieses Interface enthält die Funktionen, mit denen der *Device Manager* den passenden Treiber auswählt und bei Bedarf automatisch zur Laufzeit installiert. Die Implementierung des *Driver Interface* stellt einen minimalen Mehraufwand dar, welcher durch die Ersparnisse im technologieabhängigen Teil kompensiert werden kann.

3.4 Vereinfachung funktionaler Treiber

In Anlehnung an *OSGi Declarated Services* [6] wurde ein generisches Verfahren definiert, das die Zuordnung des Treibers über den *OSGi Device Manager* stark vereinfacht. In einer XML-Datei werden Regeln hinterlegt, die ausgewertet werden, um die Eignung des Treibers für ein gefundenes Gerät zu bestimmen. Synonym zu den *Declarated Services* ist diese Beschreibung in dem OSGi-Bundle des Treibers hinterlegt. Der bereits oben beschriebene *Composite Function Driver* wird als abstrakte Basisklasse mit dem *Abstract Composite Functional Driver* (siehe Abbildung 3) bereitgestellt. Diese Klasse wertet die XML-Beschreibung aus und leitet die darin enthaltenen Regeln an den *OSGi Device Manager* weiter. Dadurch ist es sehr einfach möglich, einen bereits vorhandenen Treiber auf neue Geräte zu erweitern, wenn diese dasselbe Protokoll basierend auf einer anderen Übertragungstechnologie verwenden. Wenn die Anbindung eines EKGs bereits für Bluetooth realisiert worden ist, kann sie somit einfach für die Technologien ZigBee und W-LAN angeboten werden. Sofern die *Discovery Driver* für die zusätzlich zu unterstützenden Netzwerktechnologien verfügbar sind, ist lediglich eine Erweiterung in der XML-Beschreibung notwendig und die neuen Geräte können sofort verwendet werden. Neben der Möglichkeit der sehr einfachen Erweiterung eines Treibers um eine neue Kommunikationstechnologie ist auch die Implementierung für Gerätehersteller deutlich einfacher geworden. Ein großer Teil der Erweiterungen, welche für den *Device Access* notwendig sind, ist bereits implementiert und kann direkt verwendet werden. Dieses reduziert den Mehraufwand bei der Implementierung von Treibern deutlich und steigert die Effizienz und die Qualität bei der Realisierung dieser Treiber.

3.5 Verbindung mit DPWS

Mit der zunehmenden Verbreitung des Internet Protokolls (IP) auch zur Vernetzung von Geräten werden neue Anwendungsprotokolle benötigt. Die Protokolle für bestehende Technologien wie z. B. Bluetooth wurden nicht für den Einsatz in IP-Netzen entworfen. Web Services basieren auf HTTP, XML und weiteren XML-basierten Standards. Daher können sie gut in IP-Netzen eingesetzt werden. Allerdings werden Web Services bisher vorrangig in Enterprise-Umgebungen eingesetzt, in denen andere Rahmenbedingungen als in Geräteumgebungen gelten. Aus diesem Grund wurde der Standard *Devices Profile for Web Services* (DPWS) [8] geschaffen. Der Standard wird bei OASIS im technischen Komitee *Web Services Discovery and Web Services Devices Profile* (WS-DD) verwaltet. Im Juni 2009 ist die Version 1.1 des Standards erschienen. DPWS erlaubt den Einsatz von Web Services auf Ressourcen-beschränkten Geräten. Es ermöglicht das dynamische Auffinden von Geräten im Netzwerk, den sicheren Nachrichtenaustausch mit einem Gerät, die Abfrage der Beschreibung eines Gerätes zur Laufzeit sowie das Anfordern für und das Empfangen von Benachrichtigungen eines Gerätes. Dadurch ermöglich DPWS einen weitaus abstrakteren Zugriff auf Gerätefunktionen als ältere Technologien. Ein Ziel der OSAmI Geräteintegration ist, eine Schnittstelle zu schaffen, über die sowohl alte Technologien also auch neue Technologien in Anwendungen eingebunden werden können. Dabei sollte die Schnittstelle nicht die Möglichkeiten neuerer Technologien wie DPWS einschränken, aber auch nicht die Einbindung alter Technologien wie RS232 ausschließen. Daher wurden in der Geräteintegration Konventionen definiert, die beim Entwickeln von Gerätetreiberschnittstellen mit der Geräteintegration beachtet werden müssen. Diese Konventionen sind so angelegt, dass die Funktionalität moderner Vernetzungsmöglichkeiten ausgeschöpft werden kann und ältere Technologien durch entsprechende Erweiterungen im Treiber eingebunden werden können.

Als Ausgangsbasis für Treiberschnittstellen und die Konventionen gelten Java-Interfaces. Da Java-Interfaces allerdings spezifisch für die Programmiersprache Java und damit programmiersprachenabhängig sind, wird eine programmiersprachenunabhängige Schnittstellensprache unterstützt. Die Konventionen sind so festgelegt, dass sich die resultierenden Java-Interfaces in *Web Service Description Language* [9] (WSDL)-Dokumente umwandeln lassen. WSDL-Dokumente werden dazu genutzt, die Schnittstellen von Web Services zu beschreiben. Diese Dokumente werden von Entwicklern erstellt, um ein Protokoll zu definieren, und werden von Werkzeugen verwendet, um Programmcode aus dieser Beschreibung zu generieren. Dadurch muss der Entwickler nicht mehr das Protokoll implementieren, sondern kann sich auf der Server-Seite auf die Funktionalität konzentrieren bzw. auf der Client-Seite den Service direkt in seine Anwendung integrieren. Dies ermöglicht es, dass der zuvor beschriebene funktionale Treiber nicht mehr vom Entwickler erstellt werden muss, sondern von Werkzeugen automatisch erstellt wird. Dadurch können Technologien wie DPWS durch einen generischen Treiber integriert werden.

Der zweite bedeutende Vorteil der Kompatibilität zu WSDL ergibt sich durch die OSGi-DPWS-Integration [10]. Diese ermöglicht es, OSGi-Dienste mit Hilfe von DPWS und Web Services zu verteilen. Diese Verteilung funktioniert für den Programmierer völlig transparent, solange er bestimmte Regeln bei den Schnittstellen beachtet. Diese Regeln entsprechen den Schnittstellen-Konventionen, die in der Geräteintegration definiert werden. Das bedeutet, dass Geräte, die über die OSAmI Geräteintegration eingebunden sind, auch von entfernten OSGi-Plattformen angesprochen werden können.

4 Diskussion

Verglichen mit den Ansätzen früherer Projekte ist bei OSAmI ein bewusst schlanker Ansatz gewählt worden. Die Anbindung eines Gerätes wird in zwei Schichten vorgenommen, der Kommunikationstechnologie und dem gerätespezifischen Protokoll. Dort setzt die Anwendung an. Die Abstraktion besteht in erster Linie aus festen Schnittstellen und ist somit sehr Ressourcen sparend und performant.

Eine besondere Bedeutung liegt aber auch in der ausschließlichen Verwendung von OSGi-Standards. Dies ist ein deutlicher Unterschied zu existierenden Ansätzen. Diese versuchten, Geräte ohne die Verwendung existierender OSGi-Standards zu integrieren, und bildeten dadurch Insellösungen.

Darüber hinaus werden Anwendungs- und Diensteentwicklern klare, fest definierte Schnittstellen bereitgestellt. Es wird dabei aber auch den Geräteherstellern die Möglichkeit gelassen, proprietäre Erweiterungen zu realisieren und damit auch Besonderheiten der eigenen Geräte über die Treiber zugänglich zu machen.

Bei fehlenden Schnittstellen für Gerätedienste können mit etwas Grundlagenwissen über Java-Schnittstellen neue Definitionen erstellt werden. Diese können dann in die Codebasis der OSAmI Geräteintegration aufgenommen werden oder auf anderem geeigneten Weg veröffentlicht werden.

5 Ausblick

Im Rahmen des OSAmI-Projektes ist eine Basis-Implementierung des in diesem Beitrag beschriebenen Systems realisiert worden. Dieses wird nun in einigen Forschungsprojekten erprobt, um das System weiter optimieren zu können. Dabei erkannte Vereinfachungen für die Anwender können in eine neue Version der OSAmI Geräteintegration einfließen.

Der Erfolg des Konzepts ist darüber hinaus von der Breite der vordefinierten Schnittstellen für Gerätedienste abhängig. Daher werden weitere solche Schnittstellen definiert und bereitgestellt. Hier sei auf die weiteren internationalen OSAmI-Teilprojekte hingewiesen, welche ebenfalls Geräte unterschiedlichster Hersteller in ihren Systemen verwenden und die Problematik der Herstellerunabhängigkeit berücksichtigen werden. Daher werden hier weitere API-Definitionen für diverse Geräteklassen entstehen.

6 Literatur

[1] Eichelberg M (Hrsg.): *Interoperabilität von AAL-Systemkomponenten, Teil 1: Stand der Technik,* VDE-Verlag, Berlin, 2009

[2] Deugd Sd, Carroll R, Kelly K, et al., *SODA: Service Oriented Device Architecture.* IEEE Pervasive Computing 5, 3 (Jul. 2006), 94-96

[3] Eichelberg M, Stewing FJ, Thronicke W, et al., *OSAMI Commons: Eine Softwareplattform für flexible verteilte Dienstesysteme über Geräten und eingebetteten Systemen,* Tagungsband Ambient Assisted Living, 2. Deutscher AAL-Kongress, Berlin, Germany, January 27-28, 2009, S. 269-272

[4] Lipprandt M, Eichelberg M, Thronicke W, et al., *OSAMI-D: An Open Service Platform for Healthcare Monitoring Applications.* Proceedings 2nd International Conference on Human System Interaction (HSI '09), 21-23 May 2009, Catania, Italy, pp. 139-145

[5] OSGi Alliance, *OSGi Service Platform, Core Specification, Release 4, Version 4.1,* April 2007

[6] OSGi Alliance, *OSGi Service Platform, Service Compendium, Release 4, Version 4.1,* April 2007

[7] OSGi Alliance, *Listeners Considered Harmful: The "Whiteboard" Pattern,* Technical Whitepaper, Revision 2.0, August 17 2004

[8] OASIS *Devices Profile for Web Services, Version 1.1,* 1 July 2009, http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.html

[9] Christensen E, et al, *Web Services Description Language (WSDL) 1.1*, March 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[10] Fiehe C, et al., *Location-Transparent Integration of Distributed OSGi Frameworks and Web Services*, Advanced Information Networking and Applications Workshops, International Conference on, pp. 464-469, 2009