Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung - AW2 - SS 2010

Benedikt Johannsen

Ambient Intelligence Networks
current work

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

1 Einführung

Im Rahmen des Aufbaus des „Living Place Hamburg“ an der HAW wird eine intelligente Wohnung entwickelt und aufgebaut. Hierfür werden eine große Anzahl von Sensoren, Aktoren und Kontrollern benötigt. Damit diese miteinander interagieren können wird eine Kommunikationsinfrastruktur benötigt. In dieser Ausarbeitung wird der Aspekt der Vernetzung näher erläutert und Ansätze aus verwandten Projekten vorgestellt und mit den Anforderungen des Living Place Hamburg verglichen. (vgl. von Luck u. a. [2010])

Abbildung 1.1: Logo des Living Place Hamburg

1.1 Rückblick AW1

Im vorherigen Semester wurden die Grundlagen für dieses Themengebiet erarbeitet. Es wurde unter anderem eine Anforderungsanalyse an Bussysteme in Ambient Intelligence Anwendungen beschrieben. Dort wurden physikalische, leistungs- und sicherheitstechnische Anforderungen formuliert die nun als Basis für die Bewertung vergleichbarer Projekte dienen. Besonderes Augenmerkt liegt hierbei auf der dynamischen Konfiguration und der Handhabung der Heterogenität der Systeme. (vgl. Johannsen [2009])
Zudem wurden verschiedene Bussysteme auf ihre speziellen Eigenschaften hin untersucht und verglichen um deren prinzipiellen Einsatzmöglichkeiten auszuloten. (vgl. Pautz [2009])

2 Vergleichbare Projekte

Es gibt zahlreiche Projekte die sich mit der Thematik „Ambient Intelligence“ beschäftigen. Der Großteil setzt beim Thema Vernetzung auf ein homogenes System aus der heutigen Hausvernetzung wie beispielsweise KNX. Nur wenige Projekte wagen einen unabhängigen Ansatz. Zwei davon sind die hier vorgestellten, die Hydra middleware und das Amigo project.

2.1 Hydra middleware

Die Hydra middleware ist ein Projekt das im Rahmen des „Sixth Framework Programme in the area of Networked Embedded Systems“ der Europäischen Union mitfinanziert wurde. Unter der Organisation des Frauenhofer Instituts beteiligten sich 16 Unternehmen und Hochschulen wie beispielsweise die Telefónica S.A. oder die Siemens AG. Es war auf vier Jahre ausgelegt und wurde im Jahr 2009 beendet.

Abbildung 2.1: Logo der Hydra Middleware

2.1.1 Zielsetzung

Zielsetzung des Projekts war die Entwicklung einer Middleware die die Hardware- und Systemunabhängige Kommunikation zwischen Embedded Systemen ermöglichen soll. Dies soll zudem mittels einer automatischen Konfiguration beziehungs-

weise Rekonfiguration geschehen. Zusätzlich soll der Ressourcenverbrauch gering gehalten werden, vor allem im Hinblick auf Speicher und Stromverbrauch, um auch leistungsschwächere Systeme integrieren zu können.
Hydra bezieht sieht in diesem Punkt allerdings auf Geräte wie PDAs und Mobiltelefone und weniger auf mikrokontrollerbasierte Systeme.

2.1.2 Funktionsweise

Hydra verwendet eine Service-orientierte Architektur kombiniert mit einem Modelgetriebenen-Architektur. Jede Funktionalität und jedes Gerät wird durch einen Service repräsentiert. Als Technologie wird Webservices verwendet, wobei die Beschreibung der Services in WSDL und die Kommunikation per SOAP realisiert wird. Die Applikationen und Geräte werden durch Ontologien beschrieben.(vgl. Hydra Consortium [2007a])

Device- und Service-Discovery

Um einen Dienst beziehungsweise ein Gerät zu finden, setzt Hydra einen OSGi-Proxy als zentrale Instanz ein. An diesem melden sich neue Geräte und Dienste an. Da der Discovery-Mechanismus von OSGi in seinem Funktionsumfang sehr eingeschränkt ist wird er mittels eines Adapters erweitert um die Kompatibilität zu den Ontologien zu gewährleisten und eine genauere Suche zu ermöglichen.

Wird ein Hydra-Device in eine Infrastruktur eingebunden, werden folgende Schritte ausgeführt:

- 1. Das Gerät registriert sich beim OSGi-Proxy
- 2. Das Gerät meldet seine Services an und registriert diese beim OSGi-Proxy

Es ist nun durch andere Geräte auffindbar, ebenso wie seine Dienste. Soll eine Aktivität beziehungsweise eine Aktion ausgeführt werden, muss folgender Ablauf abgearbeitet werden:

1.Device-Discovery

Es muss mindestens ein Device gefunden werden, das den gesuchten Anforderungen entspricht. Kriterien können zum Beispiel die Lokalität oder die Leistungsfähigkeit sein. Eine entsprechende Anfrage wird an den Service-Proxy gestellt.

2.Service-Discovery

Wenn es zutreffende Geräte gibt, wird nun nach einem passenden Service gesucht.

3.SOAP-Aufruf

Es wird die entsprechende Methode aufgerufen.

Durch diese Struktur kann nun sich jede Applikation ,beziehungsweise jeder Agent, als Service registrieren und somit seine Aufrufe starten.

Modellierung durch Ontologien

Um Services und Devices allgemeingültig zu beschreiben verwendet Hydra Ontologien. Diese wurden im nachfolgend beschriebenen Amigo Projekt entwickelt und von Hydra nur leicht modifiziert. Neben Ontologien für Geräte und Lokalität wurde auch der Kontext dementsprechend modelliert. Die folgende Abbildung zeigt einen Ausschnitt aus der Device-Ontologie. (vgl. Sarnovský u. a. [2007])

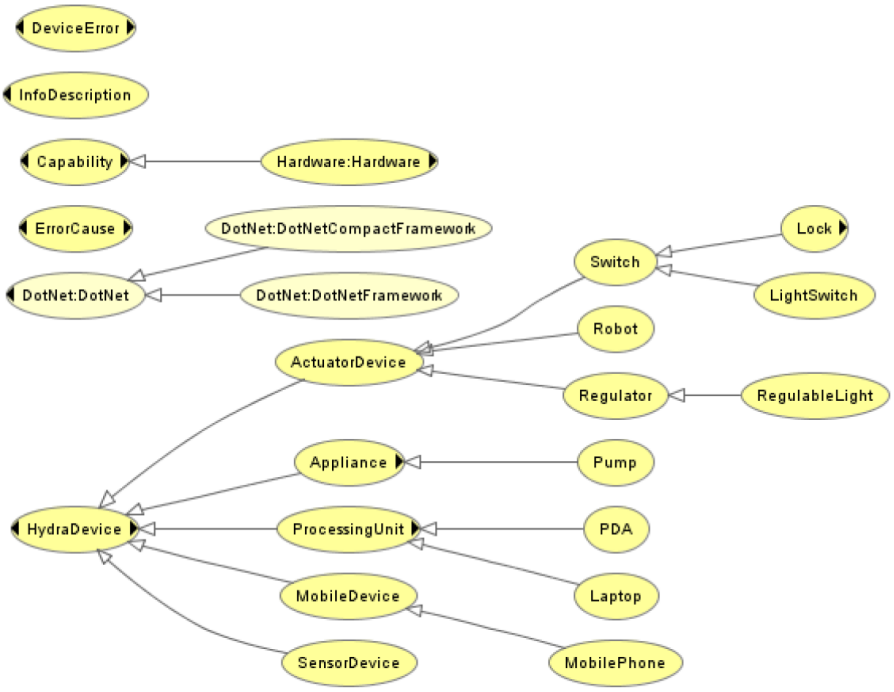

Abbildung 2.2: Ausschnitt der Device-Ontologie von Hydra (Quelle Hydra Consortium [2007b])

Hier wird deutlich das jeder Device Typ beschrieben werden kann und durch diese Struktur auf Software-Seite jedes Gerät integriert werden kann.

Semantic Devices

Um jedes Gerät integrieren zu können nutzt Hydra ein Konzept namens „Semantic Devices“. Dieses ermöglicht es mehrere Geräte zu einem Sematic Device zusammenzufassen. So lassen sich zum Beispiel die Geräte „Pumpe“, „Heizelement“ und „Ventil“ kappseln zu einem Device „Heizung“. Somit muss der Anwendungsentwickler nur noch die Heizung als ganzes ansprechen und braucht nicht mehr jede Subkomponente kontrollieren.
Eine weitere Möglichkeit ist die Einbindung von Geräten, die zu leistungsschwach sind oder keine Programmierbarkeit bieten. Sie können über einen Proxy-Service eingebunden werden.(vgl. Sarnovský u. a. [2008])

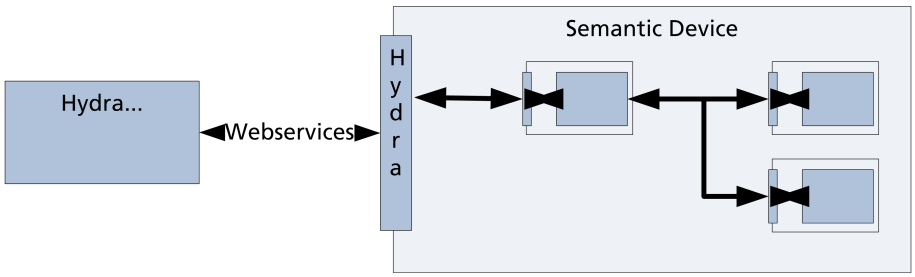

Abbildung 2.3: Beispiel Semantic Device

2.1.3 Bewertung

Hydra bietet einen sehr flexiblen und dynamischen Ansatz. Durch den Service-orientierten Ansatz und die Verwendung von verbreiteten, offenen Standards und dem Konzept der Semantic Devices lassen sich Geräte jeglicher Art in die Infrastruktur integrieren. Hauptvorteil ist allerdings die Kappselung der Struktur des Netzwerks von der Anwendungsebene. Ein Anwendungsentwickler braucht keine Kenntnisse von konkreten Geräten zu haben, sondern benötigt nur noch deren Eigenschaften. Somit kann sich die Struktur des Netzes ändern ohne das die Anwendungen davon betroffen sind.

Als Nachteil ist zu bewerten, das der Einsatz von „Web2.0-Technologien“ wie Webservices, WSDL oder SOAP sehr rechenintensiv ist. Zu der Skalierbarkeit und zu den Reaktionszeiten sind leider keine Mess- ode Erfahrungswerte bekannt. Es ist jedoch anzunehmen das hier schnell spürbare Latenzen auftreten, vor allem auf den seitens Hydra anvisierten leistungsschwachen Zielgeräten.

2.2 Amigo project

Das Amigo Project wurde ähnliche wie Hydra durch das „Sixth Framework Programme in the area of Networked Embedded Systems“ der Europäischen Union mitfinanziert. Es wird von Philips koordiniert und von Unternehmen wie der France Telecom und Microsoft unterstützt. Es wurde 2004 ins Leben gerufen und lag daher zeitlich vor Hydra. Die Zielsetzung und Herangehensweisen der beiden Projekte sind weitestgehend identisch.

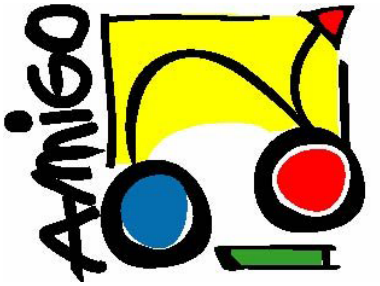

Abbildung 2.4: Logo des Amigo Projekts

2.2.1 Funktionsweise

Analog zu Hydra setzt auch das Amigo Project auf einen Ansatz der auf „Web 2.0“-Technologien basiert. Es wird ebenfalls eine Middleware entwickelt und auf einen Service-Orientierten Ansatz gebaut. Der grobe Ansatz ist absolut identisch. Die Modellierung sowie die Service-Discovery Mechanismen sind ebenfalls analog zur Hydra Middleware. Aus diesem Grund wird hier auf eine detailiertere Beschreibung verzichtet.

3 Vergleich mit dem Living Place Hamburg

Hydra wie auch Amigo bieten ein komplettes Framework, das neben der Kommunikation auch direkt das Kontext-Managment übernimmt. Ein Einsatz von Teilkomponenten ist nicht möglich. Somit bleibt der Blick auf die Ansätze beider Lösungen, insbesondere der Ontologien beziehungsweise des Service-Orientierten Designs.

Durch die Service-Orientierte Architektur können Geräte beliebig hinzugefügt oder entfernt werden. Das System kann somit jederzeit erweitert und modifiziert werden ohne das die Anwendungen hiervon betroffen sind. Die Anwendungen kennen nur die Attribute für die Geräte, die sie benötigen und nicht die Geräte selbst. Soll nun zum Beispiel die Beleuchtung in einem Zimmer der Wohnung angeschaltet werden, sucht die Applikation nach Beleuchtungs-Aktoren mit der entsprechenden Lokalität. Wird nun eine neue Lampe in dem Zimmer installiert, muss die Anwendung nicht angepasst werden, da die neue Lampe automatisch durch die Abfrage erfasst wird.

Diese Eigenschaft bietet das derzeitige System im Living Place Hamburg nicht. Zur Zeit muss dort jede Anwendung konkrete Informationen über die Geräte haben, die es nutzen will. Um diesen Aspekt in Zukunft besser begegnen zu können, wäre also eine Adaption der Ansätze von Hydra eine Möglichkeit.

4 Modellierung des Netzwerks

Die Vorteile der Hydra Middleware basieren auf der Modellierung der Komponenten mittels Ontologien. Um die daraus resultierende Flexibilität auch für das Living Place Hamburg nutzen zu können, wird im folgenden ein möglicher Lösungsansatz auf Basis des Common Information Modells aufgezeigt.

4.1 Ontolien und Modelle

Eine Ontologie beschreibt Bezeichnungen und deren Zusammenhänge allgemeingültig und unabhängig von der speziellen Anwendung oder ihrer Domäne. Ein Modell hingegen ist üblicherweise anwendungsbezogen und nur bedingt allgemeingültig. (vgl. Spyns [2002])

4.2 Common Information Model

Das Common Information Model, kurz CIM, wurde 1997 von der Distributed Managment Task Force (DMTF) veröffentlicht. Die DMTF ist eine Organisation mehrerer Unternehmen um gemeinsame Standards zu entwickelt. Ihr gehören unter anderem Firmen wie Cisco, Compaq, Dell, HP, IBM, Intel und Microsoft an.
CIM bietet ein Datenmodell mit dem Systemkomponenten, Hardware wie auch Software, beschrieben werden können. Die Zielsetzung ist es eine einheitliche Modellierungsmöglichkeit zu schaffen um große, verteilte Systeme wie beispielsweise Rechenzentren und Serverfarmen besser verwalten zu können. (vgl. Distributed Management Task Force, Inc. [2010])

4.3 Struktur und Eigenschaften von CIM

CIM basiert auf den Eigenschaften der objektorientierten Programmierung mit den entsprechenden Eigenschaften wie Polymorphie. Es ist in UML Klassendiagrammen spezifiziert und ist hierarchisch aufgebaut. Es besteht aus vier Schichten:

Meta-Schema

Das Meta-Schema spezifiziert das Metamodell und damit die Semantik von CIM. Es enthält somit die grundlegenden Elemente.

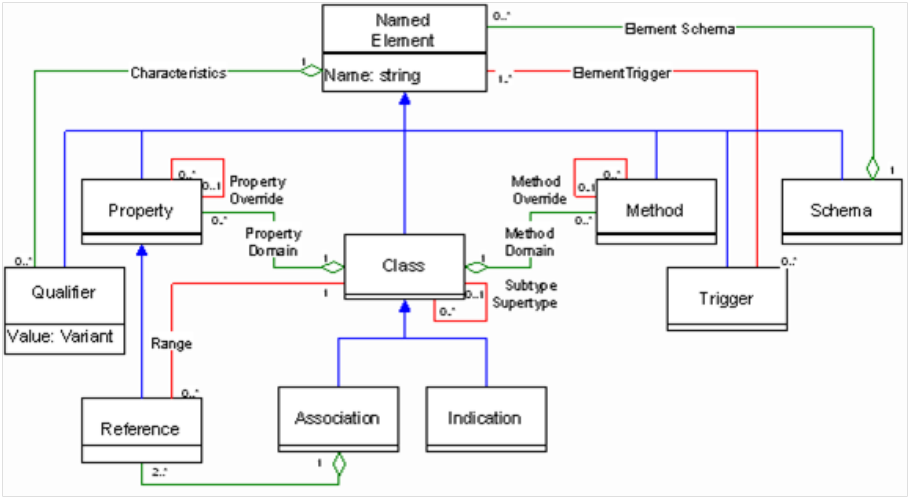

Abbildung 4.1: Ausschnitt des CIM Meta-Schemas

Wie in Grafik 4.1 erkennbar ist handelt es sich um ein leicht modifiziertes UML Klassendiagramm mit Klassen, Methoden und Attributen. Hinzugekommen sind Elemente wie „Indications“, die Benachrichtigung definierten. Diese können beispielweise als Warnung definiert werden, wenn ein Grenzwert eines Attributs erreicht wird.

Core-Schema

Dieses Schema stellt generalisierte Basisklassen bereit von denen letzendlich alle weiteren Klassen abgeleitet werden.

Common-Schema

Das Common-Schema stellt spezielle Klassen für jede Anwendung bereit. So gibt es Common-Schemas für Hardware, Software, Peripherie, Netzwerkelement und Ähnliches. Insgesamt enthält die aktuelle CIM-Spezifikation über 700 Klassen.

Extension-Schema

Sollte dennoch eine Eigenschaft eines Geräts nicht mit dem Common-Schema abgedeckt sein, so sieht CIM ein Extension Schema vor. Dies sind unspezifizierte Erweiterungen. Üblicherweise wird hierbei die passendste Common-Schema Klasse vererbt und entsprechend angepasst.

Desweiteren gibt es ein Format für die textuelle Beschreibung dieser Klassen, sogenannte Managed Object Files (MOF). Diese ermöglichen eine maschinenlesbare Form der Spezifikation und Instanziierung der Klassen. Das folgende Listing zeigt beispielhaft die Definition der neuen Klasse $WindowDriver$ die von der Klasse CIM_Device abgeleitet ist. Danach wird eine Klasse dieses Typs instanziiert.

Listing 4.1: Beispiel für ein Managed Object File

```
1  Class WindowDriver:CIM_Device {
2    [read, Units("cm")] uint8 max_position;
3    [read, Units("cm")] uint8 position;
4    uint8 close();
5    uint8 gotoPosition(IN uint8 position);
6  };
7
8  Instance of WindowDriver {
9    DeviceID = "iFlat-window03";
10   Description = "Suedseite Oberlicht 3";
11   max_position = 20;
12 };
```

4.4 Eignung für das Living Place

Das Datenmodell von CIM bietet sehr detailierte Modelle und die Möglichkeit, durch eigene Extension-Schema-Klassen anwendungsspezifische Attribute zu ergänzen. Durch diese Erweiterbarkeit ist es denkbar eine Klasse „Living Place Object“ zu definieren, die die living-place-spezifischen Attribute enthält. Zudem könnten die „Indications“ für eine einfache Signalisierung von Aktivitäten oder Kontextänderungen genutzt werden.
Dies wäre ein alternativer Ansatz zu den von Hydra oder Amigo eingesetzten Ontologien. Wie bereits in 4.1 Ontolien und Modelle erwähnt wäre dieser Modell-Basierte Ansatz allerdings anwendungsbezogen und nicht allgemeingültig. Die Funktionalität wird hierdurch allerdings nicht eingeschränkt.

5 Zusammenfassung und Ausblick

Sowohl Hydra als auch Amigo bieten ein umfangreiches Komplettpaket um ein flexibles Netzwerk zu betreiben das alle Vorraussetzungen erfüllt um kontextbasierte Ambient Intelligence Anwendungen zu ermöglichen. Allerdings bringt die Verwendung von „Web2.0“-Technologien wie Webservices einen erhöhten Ressourcenbedarf mit sich. Kleingeräte oder Sensorknoten, die eigentlich auf kleinen und stromsparenden Mikrokontroller-Plattformen realisiert werden könnten, können so nicht direkt eingebunden werden und benötigen einen zusätzlichen Proxy, der für sie die Middleware übernimmt.
Allerdings bietet die Ontologie-basierte Flexibilität der Lösungen viele Vorteile, insbesondere im Hinblick auf die Unabhängigkeit der Anwedungsentwicklung. Diese Aspekte lassen sich möglicherweise auch für das Living Place Hamburg nutzen. Hierzu müsste die Wohnung und deren Teilnehmer modelliert werden. Ein möglicher Ansatz wäre hier die Nutzung des Common Information Model.
Es könnte perspektivisch ein Modell des Living Place mit Hilfe von CIM erstellt und eine entsprechende API entwickelt werden. Die bisherige Kommunikationsschnittstelle könnte weiter verwendet werden. Dadurch könnte ein Testlauf auch parallel zum bisherigen Betrieb laufen.

Literaturverzeichnis

[Amigo Consortium 2005a] Amigo Consortium: *D2.1 Specification of the Amigo Abstract Middleware Architecture*, 2005. – URL http://www.hitech-projects.com/euprojects/amigo/deliverables/Amigo_WP2_D2.1_v10%20final.pdf. – Zugriffsdatum: 24.05.2010

[Amigo Consortium 2005b] Amigo Consortium: *D3.1 Detailed Design of the Amigo Middleware Core*, 2005. – URL http://www.hitech-projects.com/euprojects/amigo/deliverables/Amigo_WP3_D31_intropdf.pdf. – Zugriffsdatum: 24.05.2010

[Distributed Management Task Force, Inc. 2010] Distributed Management Task Force, Inc.: *Common Information Model - CIM Schema: Version 2.25.0*, 2010. – URL http://www.dmtf.org/standards/cim/cim_schema_v2250. – Zugriffsdatum: 18.05.2010

[Eisenhauer u. a. 2009] Eisenhauer, M. ; Rosengren, P. ; Antolin, P.: A Development Platform for Integrating Wireless Devices and Sensors into Ambient Intelligence Systems. In: *Proc. 6th Annual IEEE Communications Society Conf. Sensor, Mesh and Ad Hoc Communications and Networks Workshops SECON Workshops '09*, 2009, S. 1–3

[Hydra Consortium 2007a] Hydra Consortium: *D3.4 Initial architectural design specification*, 2007. – URL http://www.hydramiddleware.eu/hydra_documents/D3.4_Initial_Architectural_Design_Specification.pdf. – Zugriffsdatum: 18.05.2010

[Hydra Consortium 2007b] Hydra Consortium: *D4.2 Embedded Service SDK Prototype and Report*, Dezember 2007. – URL http://www.hydramiddleware.eu/hydra_documents/D4.2_Embedded_Service_SDK_Prototype_and_Report.pdf

[Hydra Consortium 2008a] Hydra Consortium: *D4.8 Self-* properties DDK prototype and report*. 2.0. Hydra Consortium (Veranst.), 12 2008. – URL http://www.hydramiddleware.eu/hydra_documents/D4.8_Self-star_properties_DDK_prototype_and_report.pdf

[Hydra Consortium 2008b] Hydra Consortium: *D6.3 Semantic Web Services Design Document*. 1.1. Hydra Consortium (Veranst.), 08 2008. – URL http://www.hydramiddleware.eu/hydra_documents/D6.3_Semantic_Web_services_Design_Document.pdf

[Johannsen 2009] Johannsen, Benedikt: Die Anforderungsanalyse an Feldbussystemen für Ambient Living / HAW Hamburg. HAW Hamburg, 2009. – Forschungsbericht. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master09-10-aw1/johannsen/bericht.pdf. – Zugriffsdatum: 09.03.2010. AW1 Ausarbeitung WS2009/10

[Kostelnik u. a. 2008] Kostelnik, Peter ; Sarnovsky, Martin ; Hreno, Jan ; Ahlsen, Matts ; Rosengren, Peter ; Kool, Peeter ; Axling, Mathias: Semantic Devices for Ambient Environment Middleware. In: *ACM* 1 (2008), S. 1–6

[von Luck u. a. 2010] Luck, Prof. Dr. K. von ; Klemke, Prof. Dr. G. ; Gregor, Sebastian ; Rahimi, Mohammad A. ; Vogt, Matthias: Living Place Hamburg – A place for concepts of IT based modern living / Hamburg University of Applied Sciences. URL http://livingplace.informatik.haw-hamburg.de/content/LivingPlaceHamburg_en.pdf, Mai 2010. – Forschungsbericht

[Pautz 2009] Pautz, Alexander: Analyse von Feldbussystemen in Hinblick auf Ambient Living / HAW Hamburg. HAW Hamburg, 2009. – Forschungsbericht. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master09-10-aw1/pautz/bericht.pdf. – Zugriffsdatum: 12.03.2010. AW1 Ausarbeitung WS2009/10

[Sarnovský u. a. 2007] Sarnovský, Martin ; Butka, Peter ; Kostelnik, Peter ; Lackova, Dasa: HYDRA - Network Embedded System Middleware for Ambient Intelligent Devices. In: *ACM* 1 (2007), S. 1–4

[Sarnovský u. a. 2008] Sarnovský, Martin ; Kostelník, Peter ; Hreno, Ján ; Butka, Peter: Device Description in HYDRA Middleware. In: *ACM* 1 (2008), S. 1–4

[Spyns 2002] Spyns, Peter: Data modelling versus Ontology engineering. In: *SIGMOD Record* 31 (2002), S. 12–17

[Voskuhl 2009] Voskuhl, Sören: Bereitstellung einer Sensorwolke / HAW Hamburg. HAW Hamburg, 2009. – Forschungsbericht. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master09-10-aw1/Voskuhl/folien.pdf. – Zugriffsdatum: 20.02.2010. AW1 Ausarbeitung WS2009/10

[Zhang und Hansen 2008] Zhang, Weishan ; Hansen, Klaus M.: Towards Self-managed Pervasive Middleware using OWL/SWRL ontologies. In: *acm* 1 (2008), S. 1–6