Smart Environments als selbst konfigurierende Systeme

Malte Heidenreich

Hochschule für Angewandte Wissenschaften Hamburg
malte.heidenreich@haw-hamburg.de

Abstract. Die Verwaltung von Smart Environments kann durch Selbstkonfiguration vereinfacht werden. Ziel ist es daher eine Vielzahl von Diensten dazu zu bringen, ohne einen manuellen Konfigurationsschritt untereinander Informationen auszutauschen. Es wird gezeigt, welche technischen Möglichkeiten gegeben sind, um dieses Ziel zu erreichen. Eine der gezeigten, wissenschaftlichen Arbeiten stellt ein Softwaresystem vor, das auf Ontologien basiert und weitestgehend automatisiert arbeitet. Eine weitere Arbeit zeigt den technischen Stand des Living Place der HAW Hamburg. Dort ermöglicht eine Art der Gruppenkommunikation den Austausch von Informationen zwischen den Diensten.

Keywords: self-configuration, smart environment, smart space, autonomic computing, ubiquitous computing, pervasive computing

1 Einleitung und Motivation

Dienste sind heute allgegenwärtig. Der Begriff "Everything as a Service" geht durch die Schlagzeilen der Technik-Magazine. Das Magazin t3n schreibt: "Noch nie war es so einfach, neue Produkte aus verschiedenen XaaS zu bauen" [4]. Besonders schnell steigt die Anzahl der Dienste für Smart Environments, da dort Dienste in vielen verschiedenen Ausprägungen angeboten werden. So müssen Dienste nicht nur kontextbasiert arbeiten, sondern auch auf den individuellen Benutzer eingehen können. Die Kontextualisierung berücksichtigt bei der Auswahl eines Dienstes für eine Anwendung beispielsweise den Ort, die Jahreszeit oder das Wetter. Eine Auswahl geeigneter Dienste für die Anwender wird durch die Individualisierung über Profile ermöglicht. Durch die steigende Anzahl von Diensten und deren Ausprägungen steigt der Aufwand zur Konfiguration sehr stark. Die heterogene Anwendungslandschaft wird komplex und die Installation kompliziert. Selbst konfigurierende Systeme können dieses Problem lösen. Dadurch können die Kosten und der Zeitaufwand verringert und der Nutzen und die Produktivität erhöht werden. In diesem Fall ist dies für Entwickler und Endanwender gleichermaßen nützlich, denn wenn sich etwas selbst konfigurieren kann, muss es kein anderer machen.

2 Selbst konfigurierende Systeme

In diesem Abschnitt wird erklärt, was selbst konfigurierende Systeme sind und woher der Begriff stammt.

2.1 Autonomic Computing

Die Ursprünge selbst konfigurierender Systeme finden sich im Autonomic Computing. Der Begriff tauchte erstmals in einem technischen Report von IBM auf. Die Autoren nennen vier Selbstmanagement-Eigenschaften für Systeme, die dem Paradigma des Autonomic Computing folgen [1]:

- Self-configuration
- Self-healing
- Self-protection
- Self-optimization

2.2 Organic Computing

Das Organic Computing wird als Weiterentwicklung des Autonomic Computing gesehen. Es kapselt verschiedene Forschungsdiziplinen, wie beispielsweise die Neurowissenschaften, die Molekularbiologie und die Informatik. Der Begriff ist aus der Biologie abgeleitet [9]. Zu diesem Forschungsbereich wurde von der Informationstechnischen Gesellschaft im VDE (ITG) und Gesellschaft für Informatik (GI) im Jahr 2003 die Organic Computing Initiative gegründet. Die Organic Computing Initiative nennt zwei weitere Selbstmanagement-Eigenschaften, die das ursprüngliche Autonomic Computing ergänzen:

- Self-explanation
- Self-describing

Selbst konfigurierende Systeme haben das Ziel die Komplexität technischer Systeme mit Hilfe von Technologie handhabbarer zu machen. Ein solches System soll, ähnlich wie der menschliche Körper den Blutzucker, die Herzfrequenz oder die Temperatur reguliert, weitestgehend eigenständig arbeiten, sodass nur selten manuelle Eingriffe durch den Menschen notwendig werden.

Die Weiterentwicklung des Organic Computing ist die Service-oriented architecture (SOA) [7]. Dort werden elementare Dienste dynamisch zu komplexeren Diensten zusammengefasst.

2.3 Multiagentensysteme

Ein Multiagentensystem, als selbst konfigurierendes System, besteht aus Diensten, die durch aktive Agenten repräsentiert werden. Ein Agent erzeugt und nutzt Wissen. Um dieses Wissen erzeugen zu können benötigt der Agent Sensoren. Die Kommunikation mit der Umwelt wird über eine geteilte Datenstruktur

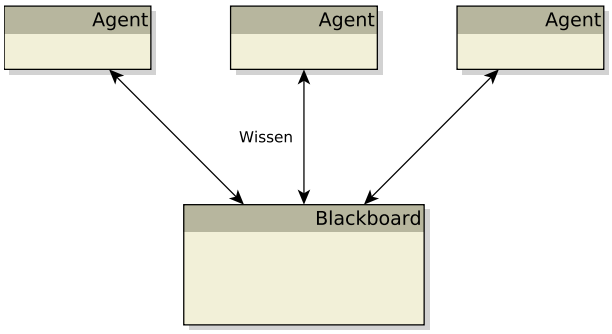

Fig. 1. Agenten tauschen Wissen über ein Blackboard aus

ermöglicht. Die Blackboard-Architektur ist geeignet, um Wissen zwischen den Agenten auszutauschen.

Die Abbildung 1 zeigt die Kommunikation der Agenten mit dem Blackboard. Die Agenten sind Wissensquellen, sie erzeugen und nutzen Informationen, die über das Blackboard geteilt werden.

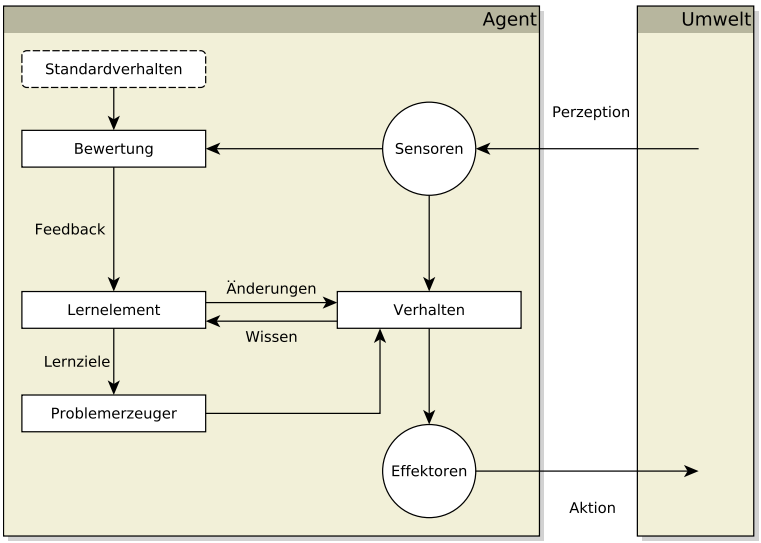

Fig. 2. Ein Selbstlernender Agent, wie er für das Autonomic/Organic Computing benötigt wird (nach [8])

Die Abbildung 2 zeigt das Schema eines selbstlernenden Agenten. Dieser kann durch das Lernelement alle Eigenschaften des Organic Computing erfüllen.

3 Smart Environments

In diesem Abschnitt wird erklärt, was Smart Environments sind und woher der Begriff stammt.

3.1 Ubiquitous und Pervasive Computing

Mark Weiser hatte im Jahr 1991 die Vision, dass "Spezialisierte Elemente aus Hardware und Software, verbunden durch Drähte, Radiowellen und Infrarot, so allgegenwärtig sein werden, dass niemand ihre Anwesenheit bemerkt" [11]. Der Begriff *ubiquitous*, den er für *allgegenwärtig* verwendete, hat sich in diesem Fachgebiet durchgesetzt. Im Gegensatz dazu wurde das Pervasive Computing, das *alles durchdringende Computing*, durch die Industrie geprägt [6]. Die Hard- und Softwareelemente sind dort untereinander stark gekoppelt.

3.2 Smart Environment

Das *Smart Environment* ist ein Konzept, das seinen Ursprung im Ubiquitous/Pervasive Computing hat. Diesem Konzept liegt *Sense Plan Act* zugrunde, einer Steuerungsarchitektur aus der Robotik. In einem Smart Environment wird die reale, physikalische Welt mit situationsabhängig genutzter Technologie kombiniert (Enriched Reality) [10]. In einem Smart Environment findet man Sensoren, Computer, Aktoren und Displays. Beispiele für Smart Environments sind *Smart Home*, *Smart Car* und *Smart Store*.

4 Verwandte Arbeiten

In diesem Abschnitt werden wissenschaftliche Arbeiten vorgestellt, die Grundlagen für eigene Arbeiten liefern sollen.

4.1 Toward a Self-Configuration Middleware for Smart Spaces [3]

Diese Arbeit wurde 2009 von Gouin-Vallerand et al. im DOMUS Laboratory an der Université de Sherbrooke in Kanada verfasst.

In dieser Arbeit werden das Autonomic Computing und das Pervasive Computing kombiniert. Die entwickelte Software basiert auf der serviceorientierten Architektur (SOA). Von den von IBM genannten Eigenschaften wird nur die Self-Configuration-Eigenschaft implementiert. Es werden mehrere Quellen genannt, in denen die anderen drei Eigenschaften behandelt werden.

Die Abbildung 3 zeigt die Komponenten der Self-Configuration Middleware, die im Folgenden erklärt werden:

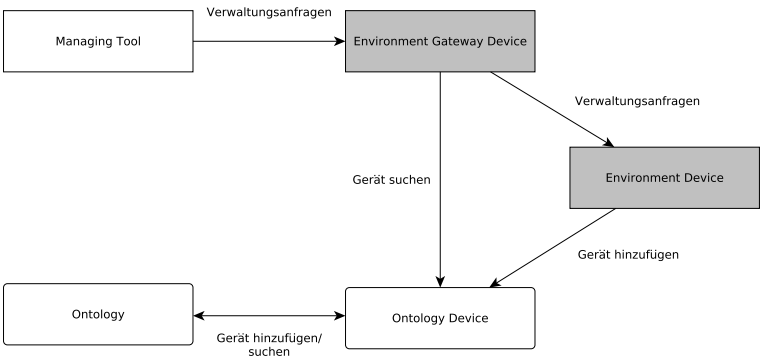

Fig. 3. Komponenten der Self-Configuration Middleware (nach [3])

Managing Tool
Das Managing Tool ist eine Anwendung mit grafischer Benutzeroberfläche, um den Smart Space zu verwalten und zu konfigurieren. Es kann die aktiven Geräte und die dort bereitgestellten Anwendungen in jedem Raum des Smart Space zeigen. Eine Karte vereinfacht die Übersicht. Das Managing Tool kommuniziert mit sogenannten Environment Gateway Devices, die in jedem Smart Space installiert sind, über das Simple Object Access Protocol (SOAP). Über das Protokoll können Informationen zum Kontext des Smart Space, beispielsweise der Standort von Geräten angefragt werden. Das Managing Tool kann Logdateien der Dienste auf den Geräten im Smart Space anzeigen.

Environment Gateway Device
Die Environment Gateway Devices empfangen die Verwaltungsanfragen des Managing Tools. Eine Anfrage wird analysiert und ein passendes Gerät wird abhängig vom Kontext der Anfrage herausgesucht. Anschließend wird sie an dieses Gerät weitergeleitet. Ist die Bearbeitung abgeschlossen, wird das Ergebnis über den gleichen Weg zurück an das Managing Tool gegeben.

Environment Device
Das Environment Device stellt die Software-Komponente dar, die auf fast allen Geräten im Smart Space installiert ist. Sie sorgt dafür, dass die angebotenen Dienste auf diesem Gerät erkannt und für andere Anwendungen bereitgestellt werden können. Zudem bietet diese Komponente eine Schnittstelle zur Verwaltung.

Ontology Device und Ontology
Das Ontology Device verwaltet die Ontologie des Smart Space. Es wird vom Environment Gateway Device genutzt, um ein Routing von Anfragen auf Geräte zu ermöglichen, die diese Aufgabe erledigen können.

Die Ontologie wurde auf Basis der Web Ontology Language (OWL) entwickelt. Sie beschreibt den Kontext des Smart Environment, indem die Menge von Entitäten und ihre Beziehungen untereinander formal dargestellt werden. Über die Ontologie kann automatisch ein geeignetes Gerät oder ein Dienst für den Kontext einer Anwendung gefunden werden.

OSGi Framework
Die implementierte Software basiert auf dem OSGi Framework. Dieses Framework ist eine Spezifikation für eine Softwareplattform auf Java-Basis. Es wurde von der OSGi Alliance (früher Open Services Gateway initiative) entwickelt. Das zugrunde liegende Komponentenmodell besteht aus zwei Komponenten. Sogenannte Bundles, auch Services genannt, repräsentieren die Anwendungen, die im Smart Space verteilt werden. Die Verwaltung dieser Anwendungen übernimmt die Service Registry. Zum Auffinden von Diensten werden unter anderem die P2P-Protokolle UPnP (Universal Plug and Play) und Jini (ein Java P2P Protokoll) genutzt. UPnP basiert auf SOAP und SSDP. Jini basiert auf RMI und JXTA (JuXTApose). Eine zusätzliche Komponente übernimmt die Konvertierung in Protokolle außerhalb der OSGi Spezifikation.

Beispielszenario
Eine neue Anwendung soll im Smart Space eingesetzt werden (siehe Abbildung 4). Dazu wird über das *Managing Tool* eine Anfrage an den *Environment Manager Service* gestellt. Dieser sucht über den *Ontology Search Service* und der zugehörigen *Smart Space Ontology* geeignete Dienste, die mit der neuen Anwendung zusammenarbeiten. Hat der *Environment Manager Service* die Liste der Dienste erhalten, wird die Anwendung installiert. Über das *Managing Tool* bekommt der Nutzer Rückmeldungen zum Status der Installation.

Die *Smart Space Ontology* wird ständig aktualisiert (siehe Abbildung 5). Dazu sucht ein *Device Manager Service* in regelmäßigen Abständen und mit Hilfe eines *Device Discovery Service* nach den vorhandenen Diensten. Der *Device Manager Service* verwaltet die Beschreibungen der Dienste. Der *Device Discovery Service* hat die Aufgabe, die gefundenen Dienste in die *Smart Space Ontology* einzutragen.

4.2 Agentenbasierte Middleware zur Entwicklerunterstützung in einem Smart-Home-Labor [2]

Die Masterarbeit von Tobias Eichler aus dem Jahr 2014 wurde im Umfeld des Living Place der HAW Hamburg verfasst.

Einige Anforderungen, die an die dort entwickelte Middleware gestellt wurden, sind ebenfalls nützlich für selbst konfigurierende Systeme:

Framework
Es sollte ein Framework entwickelt werden, dass einem neuen Entwickler einen

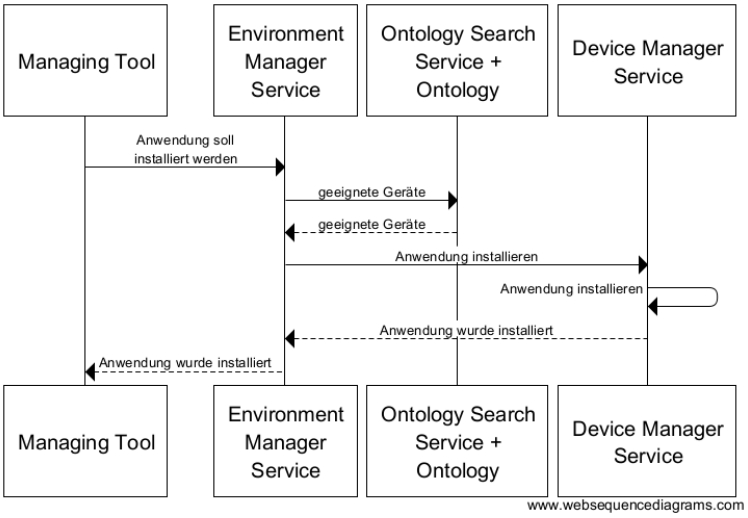

Fig. 4. Sequenzdiagramm zum Beispielszenario (nach [3])

schnellen Einstieg in das Projekt bietet. Um die Entwicklungszeit zu verringern, sollten wichtige Funktionen, die häufig benötigt werden, bereits in der Middleware verfügbar sein.

Übersicht und Kontrolle
Die Middleware bietet den Nutzern über ein Webinterface die Möglichkeit, einen Systemüberblick zu bekommen. Für die Agenten steht zu diesem Zweck ebenfalls eine Schnittstelle bereit. Die Reproduzierbarkeit von Fehlern wird durch Dienste ermöglicht, die einen definierten Stand der Umgebung herstellen und beispielsweise automatisierte Tests ermöglichen. Zentral gesammelte Logdateien erlauben es dem Entwickler schnell Informationen zu aufgetretenen Problemen zu bekommen.

Zugriff auf Entwickler-Ressourcen
Alle Entwickler und die Middleware sollen auf den Quellcode und die dazugehörige Dokumentation zugreifen können. In der Quellcodeverwaltung können zusätzlich Beispielprojekte zusammen mit Anleitungen veröffentlicht werden.

Middleware soll sich dynamisch an neue Anforderungen anpassen
Ein weiteres Ziel dieser Arbeit war, die Middleware so zu konstruieren, dass sie sich ohne manuelle Änderungen an neue Anforderungen anpassen kann. Dazu sollte eine Plattform geschaffen werden, die das Überwachen und Kontrollieren der laufenden Komponenten übernimmt.

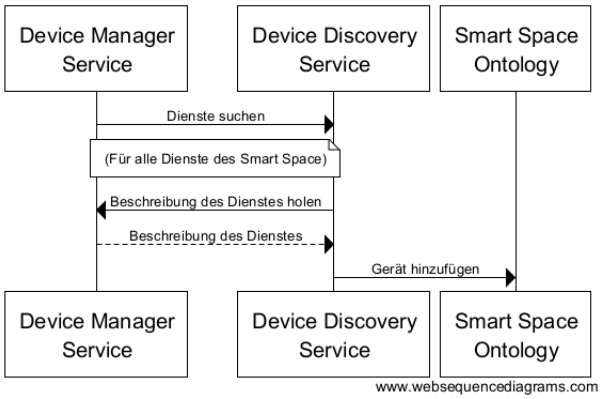

Fig. 5. Sequenzdiagramm zur Aktualisierung der Ontologie (nach [3])

Design

In der Implementierung wurde als Nachrichtenformat JSON ausgewählt. Die Kontextverarbeitung wurde über Complex-Event-Processing (CEP) realisiert. Das CEP besteht aus Methoden, Techniken und Werkzeugen, um Ereignisse kontinuierlich und zeitnah zu verarbeiten, während sie passieren. Die Laufzeitumgebung nutzt das OSGi Framework. Eine Bibliothek übersetzt Objekte in JSON-Nachrichten, um die Schnittstellen der Agenten untereinander zu vereinfachen. Die Agenten wurden nach dem Aktor-Modell von Akka implementiert. Zudem gibt es eine Komponente, die das Monitoring von Agenten übernimmt, um die Fehlertoleranz des Systems zu erhöhen. Das System soll in der Lage sein, Fehler selbst zu erkennen und darauf zu reagieren. Dies erfüllt die Anforderung *Übersicht und Kontrolle*.

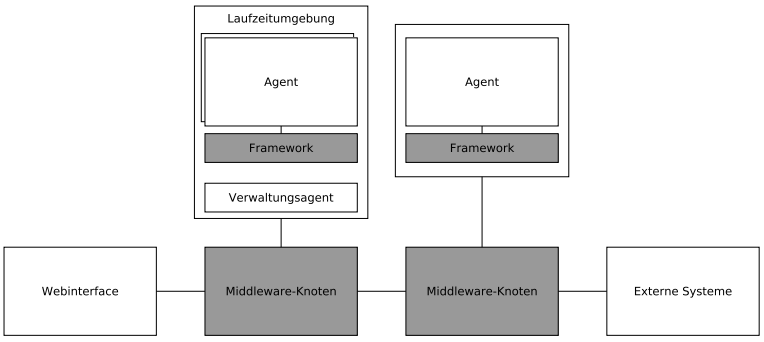

Fig. 6. Komponenten der agentenbasierten Middleware zur Entwicklerunterstützung (nach [2])

Die Abbildung 6 zeigt die Komponenten der Architektur. Die Agenten kommunizieren über das Framework mit der Middleware. Es kann mehrere Middleware-Knoten geben, um eine Lastverteilung zu realisieren. Die Aufgabe der Middleware-Knoten ist die Realisierung der Gruppenkommunikation. Die Kommunikation zwischen der Middleware und dem Framework findet über JSON-Nachrichten statt. Das Webinterface nutzt ebenfalls JSON-Nachrichten. Um externe Systeme einzubinden, wird Java Message Service verwendet. Die Verwaltung und das Complex-Event-Processing ist in die Laufzeitumgebung integriert.

5 Projektumfeld

Das Projektumfeld soll das Smart Home Labor der HAW oder der Creative Space for Technical Innovations (CSTI) sein, der sich seit Oktober 2015 im Aufbau befindet. Die Eröffnung soll im Sommersemester 2016 sein. Dort wird neben der Erforschung von Smart Environments auch an Fragestellungen aus dem Bereich der "Human Computer Interaction" (HCI) gearbeitet [5]. Im Umfeld des Smart Home Labor wurden bereits viele Master-, Bachelor- und Diplomarbeiten verfasst, unter anderem auch die Masterarbeit von Tobias Eichler. Viele Arbeiten aus dem Bereich der Smart-Home-Technologie sind auf der UbiComp-Seite[1] der HAW zu finden oder werden auf der Konferenz UbiComp behandelt, die 2016 in Heidelberg stattfindet.

Ergänzende Arbeiten finden sich unter anderem in der IEEE Conference on Self-Adaptive and Self-Organizing Systems (SASO) und der International Conference on Autonomic Computing (ICAC), die den Bereich der selbst konfigurierenden Systeme abdecken. Das DFG Schwerpunktprogramm 1183 setzt sich mit dem Thema Organic Computing auseinander. Das International Journal of Smart Home zielt darauf ab, die Erforschung der Smart-Home-Technologie und deren Anwendung zu erleichtern und zu unterstützen.

6 Eigene Arbeit

Für die eigene Arbeit gibt es viele verschiedene Ansätze. Die *Agentenbasierte Middleware zur Entwicklerunterstützung in einem Smart-Home-Labor* löst viele Probleme in einem Smart-Home-Labor. Dennoch gibt es Fälle, in denen die vorhandenen Möglichkeiten nicht ausreichend sind. Der Überblick über das System wird erschwert, wenn Entwickler zusätzliche Komponenten verwenden. Dies kann beispielsweise eine Datenbank sein, die er selbst installiert, obwohl er auch eine vorhandene nutzen könnte. Solche Redundanzen können vermieden werden, wenn das gesamte Smart Environment als ein großes, zusammenhängendes Softwaresystem gesehen wird. Mit einer Beschreibungssprache und einigen Werkzeugen könnte der Zustand festgehalten werden.

Der eigene Ansatz wäre in diesem Fall die Entwicklung der Beschreibungssprache

[1] `http://users.informatik.haw-hamburg.de/~ubicomp/`

für Smart Environments und einem Werkzeug, das es ermöglicht eine beschriebene Konfiguration auf das Smart Environment anzuwenden. Die Abgrenzung zur Arbeit *Toward a Self-Configuration Middleware for Smart Spaces* ist die Möglichkeit des Werkzeugs neue Ressourcen zu erzeugen, sollten sie nicht vorhanden sein. So kann das Werkzeug beispielsweise in einem Cluster automatisch eine Datenbank erzeugen und diese der Konfiguration des Smart Environment als neue Ressource hinzufügen. Diese kann anschließend von der neuen Anwendung genutzt werden, ohne einen manuellen Eingriff durch den Entwickler.

7 Fazit

Selbst konfigurierende Systeme und Smart Environments sind spannende Themengebiete, die eine Vielzahl von möglichen Forschungsaufgaben versprechen. Die Kombination aus beiden ist ideal, da dynamische, kurzlebige Anwendungen auf eine flexible Infrastruktur aufgebaut werden. Die Konzepte dahinter sind gut durchdacht und ausgereift und erlauben es in kurzer Zeit neue Softwaresysteme aufzubauen.

Die gezeigten Arbeiten bieten eine gute Grundlage für eigene Experimente, um Smart Environments schlussendlich als selbst konfigurierende Systeme zu entwickeln.

References

1. An architectural blueprint for autonomic computing. Tech. rep., IBM (Jun 2005)
2. Eichler, T.: Agentenbasierte Middleware zur Entwicklerunterstützung in einem Smart-Home-Labor (2014)
3. Gouin-Vallerand, C., Giroux, S., Abdulrazak, B., Molhtari, M.: Toward a self-configuration middleware for smart spaces. In: Future Generation Communication and Networking, 2008. FGCN '08. Second International Conference on. vol. 2, pp. 463–468 (Dec 2008)
4. Janschitz, M.: Everything as a Service: Der Trend der Cloud- und Hosting-Branche (2015), `http://t3n.de/news/whd-global-2015-everything-as-a-service-602429/`
5. von Luck, K., Draheim, S.: Creative Space for Technical Innovations (2015), `http://creative-space.haw-hamburg.de/`
6. Mattern, F.: Informatiklexikon der GI: Pervasive/Ubiquitous Computing (2001), `http://www.gi.de/index.php?id=77`
7. Müller-Schloer, C., Schmeck, H., Ungerer, T. (eds.): Organic Computing - A Paradigm Shift for Complex Systems. Birkhäuser (2011), `http://dblp.uni-trier.de/db/books/collections/Muller2011.html`
8. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson Education, 2 edn. (2003)
9. VDE/ITG/GI-Arbeitsgruppe Organic Computing: Organic Computing (2003), `https://www.vde.com/de/InfoCenter/Seiten/Details.aspx?eslShopItemID=d4d727a9-a596-44e4-aaaa-4aaa23fcaeb1`

10. Weiser, M., Gold, R., Brown, J.S.: The origins of ubiquitous computing research at parc in the late 1980s. IBM Syst. J. 38(4), 693–696 (Dec 1999), http://dx.doi.org/10.1147/sj.384.0693
11. Weiser, M.: The computer for the 21st century. Scientific American 265(3), 66–75 (January 1991), http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html

Alle Links wurden zuletzt am 22. Februar 2016 besucht.