Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Sören Voskuhl

Modellunabhängige Kontextinterpretation in einer Smart Home Umgebung

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Sören Voskuhl

Modellunabhängige Kontextinterpretation in einer Smart Home Umgebung

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang Informatik (Master)
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer. nat. Kai von Luck
Zweitgutachter : Prof. Dr. rer. nat. Gunter Klemke

Abgegeben am 16. Dezember 2011

Sören Voskuhl

Thema der Masterarbeit

Modellunabhängige Kontextinterpretation in einer Smart Home Umgebung

Stichworte

Intelligente Wohnbereiche, modellunabhängige Kontextinterpretation, Kontextsensitive Anwendungen, Aktivitätserkennung, Kontextsimulation, Blackboard-Architektur, Indoor-Positioning

Kurzzusammenfassung

Mit dem Übergang in das Computerzeitalter des *Ubiquitous Computing* geht einher, dass sich Menschen zunehmend in intelligenten Umgebungen bewegen. Mit dem Ziel, sich exakt an die Bedürfnisse des Anwenders anzupassen, versuchen die Applikationen in diesen Bereichen in der Regel den aktuellen Kontext in ihre Reaktionen einzubeziehen. In diesem Zusammenhang stellt die Kontexterstellung einen essentiellen Aspekt dar, weshalb im Rahmen dieser Masterarbeit eine Architektur für *Context-Aware Systeme*, im Hinblick auf intelligente Wohnbereiche, entwickelt worden ist. Darüber hinaus wurde untersucht, inwieweit eine modellunabhängige Kontextinterpretation Informationen über die zugrunde liegende Situation liefern kann. Außerdem ist eine Applikation zur Simulation verschiedener Kontexte umgesetzt worden.

Sören Voskuhl

Title of the paper

Model-independent context interpretation in a smart home environment

Keywords

Smart home, context awareness, model-independent context interpretation, activity recognition, context simulation, blackboard architecture, indoor-positioning

Abstract

In the age of *ubiquity computing* people increasingly find themselves in intelligent environments. Driven by the aim to accurately adapt to the needs of the users, these areas often try to comprise the current context in their reactions. In this regard, the creation of a context is an essential aspect. Within this thesis an architecture for context aware applications in smart homes has been developed. Furthermore it was examined, what contribution a model-independent context interpretation can provide to the context. Additionally, an application for simulating different contexts has been implemented.

Inhaltsverzeichnis

1 Einführung

Ein Trend in der Informatik zeichnet sich durch kontinuierlich kleiner werdende, leistungsfähigere und kostengünstigere Computer aus. Diese Charakteristiken führen dazu, dass Informationstechnologien einen stetig wachsenden Einfluss auf das Leben des Menschen haben. Mark Weiser hat in dieser Entwicklung bereits Anfang der neunziger Jahre die Einleitung in ein neues Computerzeitalter gesehen und dieses als *Ubiquitous Computing* bezeichnet.

> *„The third wave of computing is that of ubiquitous computing, whose cross-over point with personal computing will be around 2005-2020. The „UC“ era will have lots of computers sharing each of us. Some of these computers will be the hundreds we may access in the course of a few minutes of Internet browsing. Others will be imbedded in walls, chairs, clothing, light switches, cars – in everything. UC is fundamentally characterized by the connection of things in the world with computation.“* – Weiser und Brown (1997)

Nachdem sich die Menschen im ersten Computerzeitalter einen Großrechner mit weiteren Personen geteilt haben, waren die Systeme der zweiten Ära auf einzelne Benutzer und die Interaktion mittels Tastatur und Maus ausgelegt. Im obigen Zitat beschreibt Mark Weiser das bevorstehende dritte Zeitalter, in dem Rechenleistung omnipräsent im Alltag des Menschen ist. Dies beinhaltet, dass der Anwender die ihn umgebenden Computer lediglich unbewusst verwendet und aus seiner gesamten Umgebung digitale Informationen ziehen kann. Dieser technologische Fortschritt führt zu einem gesellschaftlichen Wandel, in welchem sich die heranwachsende Generation zu *Digital Natives* entwickelt, während die Altersgruppe der vorherigen Computerzeitalter traditionell *Digital Immigrants* repräsentieren, da sie die Interaktion mit Computern nicht bereits in der Kindheit als selbstverständlich empfunden haben (Prensky (2001)). Während ersterer Personengruppe die Nutzung neuartiger Technik in der Regel leichter fällt, haben beide Generationen gemeinsam, dass sie ihr Verhalten während der Kommunikation dem Computer anpassen. Dieser Aspekt unterbindet eine als natürlich wahrgenommene Mensch-Computer-Interaktion und erfordert perspektivisch eine Anpassung der Informationstechnologien an den Menschen.

Vor diesem Hintergrund versuchen multimodale Interaktionsformen, wie Sprachsteuerungen, gestenbasierte Steuerungen sowie Multitouch-Systeme eine intuitivere Benutzerschnittstelle

als bisherige Kommunikationsmechanismen für den Anwender zu schaffen. Weitere Systeme versuchen die explizite Kommunikation mit dem Benutzer zu minimieren und unter Einbeziehung äußerer Einflüsse einen Kontext zu erstellen, woraufhin sie ausschließlich relevante Informationen bereitstellen oder automatisch ihr gesamtes Verhalten an die Umwelt und die Bedürfnisse der Person adaptieren.

Die kontextsensitiven Systeme finden ihren Einsatz dabei zunehmend auch in den privaten Wohnräumen der Menschen, indem sich diese sukzessive zu *Smart Environments* entwickeln. Hier wird versucht, den Bewohner rechnergestützt beim Erledigen von alltäglichen Aufgaben zu unterstützen, um ihm hierdurch u.a. mehr Komfort und Sicherheit (wie z. B. das *Ambient Assisted Living*) im Eigenheim zu bieten. Ein kontextabhängiges Systemverhalten erfordert dabei im ersten Schritt eine rechnergestützte Erfassung der äußeren Bedingungen und die Modellierung eines konsistenten und allgemeingültigen Kontextes. Dabei ergeben sich bei der Implementierung von Applikationen dieser Art elementare Fragestellungen, die es für jedes System individuell zu beantworten gilt. In diesem Zusammenhang gilt es zu definieren, was den jeweiligen Kontext in der Mensch-Computer-Interaktion auszeichnet und zu eruieren, in welchem Maße einzelne Sensorwerte zu einem einzigen Faktum verschmolzen werden können.

Zielsetzung und Abgrenzung

Vom gesellschaftlichen und technologischen Wandel inspiriert, soll im Rahmen dieser Arbeit zunächst eine Architektur entstehen, auf deren Basis Applikationen für einen intelligenten Wohnbereich entwickelt werden können. Vor diesem Hintergrund soll ermittelt werden, welche Komponenten und Dienste eine Infrastruktur für kontextsensitive Systeme zur Verfügung stellen sollte, damit sie dem Entwickler neuer Anwendungen eine möglichst große Unterstützung liefert.
Ein weiteres Ziel dieser Arbeit liegt in der Evaluierung dieser Architektur, indem verschiedene Sensorwerte generiert und in die Kontextinterpretation einbezogen werden, mit dem Ziel, systemseitig adäquate Reaktionen folgen zu lassen. Zu diesem Zweck soll untersucht werden, inwieweit mittels einer modellunabhängigen Kontextinterpretation auf den rohen Sensordaten bereits Erkenntnisse über die herrschenden Bedingungen gesammelt werden können. Darüber hinaus soll analysiert werden, wie sich die verarbeiteten Daten von weiteren Interpretationsverfahren nutzen und gegebenenfalls mit semantischem Wissen anreichern lassen.

Die weiterführende Interpretation der Daten soll dabei nicht das Ziel dieser Arbeit sein und wird daher in weiteren Projekten an der Hochschule für Angewandte Wissenschaften Hamburg erforscht. Vielmehr sollen hier offene Schnittstellen geschaffen werden, die ohne großen

Aufwand eine Kommunikation von Kontextinformationen zwischen unterschiedlichen Softwarekomponenten ermöglichen.
Außerdem werden die Reaktionen der Aktoren in dieser Arbeit lediglich in theoretischer Form behandelt, da der Fokus auf der Kontexterstellung liegt. Dies hat zur Folge, dass hier primär die Selektion von Informationsquellen im Vordergrund stehen wird. Die konkrete Initiierung einer Änderung der Verhaltensweise des Gesamtsystems wird in der hier vorgestellten Anwendung auf den höher liegenden Interpretationsstufen entwickelt.

Gliederung der Arbeit

Im Kapitel 2 werden einige für das Verständnis dieser Arbeit notwendige Begriffe beleuchtet. Dabei wird u.a. der von Mark Weiser geprägte Begriff des *Ubiquitous Computing* diskutiert und der Forschungsbereich der *Smart Homes* veranschaulicht, indem unterschiedliche Marktsegmente und spezielle Anforderungen an Computersysteme in privaten Wohnbereichen präsentiert werden.

In der Analyse (3) werden drei Szenarien definiert, die verschiedene Aktivitäten in einem Wohnbereich präsentieren und vom hier vorgestellten System erkannt werden sollen. Zu diesem Zweck wird eine Anforderungsanalyse durchgeführt, mit Hilfe derer die grundlegenden Eigenschaften einer Architektur für *Context-Aware Systeme* herausgearbeitet werden. Darüber hinaus werden die Herausforderungen im Umgang mit Kontextinformationen und deren Interpretation untersucht.

Der Abschnitt 4 befasst sich mit der Verarbeitung von rohen Sensordaten und der damit verbundenen Detektion und Auflösung von Rauschen innerhalb der Datensätze. Daraufhin wird mit der Clusteranalyse ein algorithmisches Verfahren vorgestellt, mit dem Strukturen in großen Informationsbeständen und Ähnlichkeiten von Objekten aufgedeckt werden können.

Im Design (5) werden die aus der Analyse hervorgegangen Anforderungen aufgegriffen, mit dem Ziel, zentrale Designentscheidungen im Hinblick auf die zu realisierenden Architektur-Komponenten zu treffen. In diesem Zusammenhang werden Lösungsansätze für eine Kommunikationsplattform und einer Persistenzebene für die Kontextinformationen diskutiert. Des Weiteren werden die Möglichkeiten zur modellunabhängigen Kontextinterpretation und die Kombination mit höheren Level zur Aktivitätserkennung auf der Basis dieser Architektur vorgestellt.

In der Realisierung (6) folgt eine Beschreibung der technischen Umsetzung der aus dem Design stammenden Komponenten. Zudem wird unter Berücksichtigung der in der Analyse identifizierten Aktivitäten evaluiert, welche Tätigkeiten mit dem hier implementierten System erkannt werden können.

Das Kapitel 7 enthält eine Zusammenfassung dieser Arbeit und liefert einen Ausblick über die mögliche Entwicklung ubiquitärer Computersysteme.

2 Ubiquitous Computing and Smart Homes

Die stetig preiswerter werdende Hardware und die technikversierte Generation des Menschen führt eigendynamisch dazu, dass die Menge der Computer die eine Person umgeben, kontinuierlich wächst.
Der Begriff der u.a. dieses Phänomen in Worte fasst lautet *Ubiquitous Computing*. Er beschreibt die Allgegenwärtigkeit von Informationstechnologien und wurde von Mark Weiser in seiner Veröffentlichung „The Computer for the 21st Century“ (Weiser (1991)) geprägt. Die von ihm erklärte Omnipräsenz von Rechenleistung beinhaltet dabei nicht einzig die Möglichkeit persönliche Computer jederzeit durch eine steigende Anzahl verfügbar zu machen, da dieses den Anwender vor die Schwierigkeit stellt, die Verwaltung und den Zugriff auf diese Geräte in Eigenverantwortung steuern zu müssen, vgl. Dearman und Pierce (2008). Vielmehr versteht Weiser hierunter, dass der Personal Computer als Gerät menschlicher Aufmerksamkeit im Hintergrund verschwindet und vom Menschen lediglich unbewusst wahrgenommen wird, weil die Gegenstände des Alltags mit Rechenleistung versehen und untereinander vernetzt sind. Ein signifikanter Aspekt des *Ubiquitous Computing* liegt daher in der nahtlosen Interaktion zwischen Mensch und Computer, was eine Auflösung des engen Kommunikationskanals über die Tastatur und Maus sowie die Erschaffung einer Schnittstelle in das Unterbewusstsein des Menschen zu Computern erfordert.
Mit der Kreation einer solchen Schnittstelle gehen verschiedene Herausforderungen einher, deren Bewältigung sukzessive in das Computerzeitalter des *Ubiquitous Computing* führen. Einige dieser elementaren Aspekte sollen im Folgenden vorgestellt werden.

Damit Computer nicht mehr im Fokus menschlicher Aufmerksamkeit stehen, müssen sie dem Sichtfeld der Person entschwinden. Mit dieser Eigenschaft neuartiger Computersysteme befasst sich das *Disappearing Computing*. In diesem Kontext werden in Streitz u. a. (2007) zwei Arten vorgestellt, inwiefern Computer aus der menschlichen Betrachtung verschwinden können.

1. **Physikalisch:** Hier werden die Recheneinheiten der Wahrnehmung des Menschen entzogen, indem sie miniaturisiert und zumeist in weitere Gegenstände integriert werden. Beispielsweise können RFID-Chips in Karten wie dem Personalausweis einbettet

werden, um darauf weiterführende Informationen zu speichern. Darüber hinaus können Informationstechnologien in Schmuck oder Kleidung integriert werden, was zur Folge hat, dass Funktionen, die bisher ausschließlich mit einem Computer assoziiert wurden, nicht mehr als solche erkannt werden.

2. **Mental:** In diesem Fall werden die Computer vor dem „mentalen“ Auge des Benutzers unsichtbar. Hier liegt das Ziel nicht darin die Computer möglichst klein zu entwickeln, sondern in die Umwelt des Menschen einzugliedern. Hierzu können sie u.a. in das Architektonische Umfeld (z. B. Türen, Wände) oder in Möbel innerhalb des Wohnbereichs (z. B. Tische, Stühle) integriert werden.

Wenn Computer aus dem Augenmerk des Menschen entschwinden, bedeutet dieses, dass im Gegenzug neue Konzepte für die Mensch-Computer-Interaktion konstruiert werden müssen. Mit dieser Forschungsdisziplin befasst sich das *Interaction Design*. Dieser Bereich setzt sich mit den Fragen auseinander wie die Interaktionen des Anwenders mit einem System, der Umgebung oder eines Produktes dahingehend optimiert werden können, dass diese ihn und seine Aktivitäten auf einem effektiven, hilfreichen und einfach benutzbaren Weg unterstützen (Sharp u. a. (2007)). Varianten neuartiger Interaktionsmechanismen, die in den letzten Jahren an Profil gewinnen konnten, sind die Multitouch-Interaktion sowie Tangible User Interfaces (TUIs) (Ishii und Ullmer (1997)). Da der Fokus dieser Arbeit jedoch nicht auf der Erstellung neuer Konzepte für die Kommunikation zwischen Mensch und Computer liegen wird, sei an dieser Stelle auf Rahimi und Vogt (2011) für weitere Informationen über diesen Forschungsbereich an der HAW Hamburg verwiesen.

Des Weiteren muss im *Ubiquitous Computing* eine Interoperabilität zwischen den Geräten herrschen, damit die Fähigkeiten der individuellen Technologien zu einer Struktur von zueinander abgestimmten Funktionalitäten verschmolzen werden, vgl. Edwards und Grinter (2001). Sollte ein solches Zusammenspiel der Informationstechnologien nicht implementiert werden, ergeben sich einzelne Insellösungen, sodass nur gewisse Teilnehmer des Netzwerks miteinander kommunizieren können und der Nutzen für den Anwender schwindet.
Ein weiterer Aspekt der in Edwards und Grinter (2001) aufgegriffen wird und im hier vorgestellten Computerzeitalter von Bedeutung ist, liegt darin, dass der Benutzer den Überblick über die Komplexität des Gesamtsystems nicht verlieren darf, sodass für ihn unvorhersehbare systemseitige Aktionen durchgeführt werden. Die Computer müssen dem Anwender über einen intuitiven Weg erkenntlich machen, welche Fähigkeit sie besitzen, in welchem Zustand sich das Gesamtsystem befindet und wie sich ungewollt hergestellte Fehlerzustände zurücksetzen lassen. Zudem muss das allgegenwärtige Computersystem offene Schnittstellen anbieten, damit der Anwender neue Geräte hinzufügen, entfernen und modifizieren kann.
Diese Gesichtspunkte verdeutlichen, dass die Selbstbestimmung des Benutzers in Computer-ubiquitären Umgebungen stetig im Vordergrund stehen muss, sodass dieser in unangemessenen Situationen die Aktionen sämtlicher Computer unterbinden kann. Diese Tatsache sollte jedoch kein Hindernis für die Realisierung dieser Systeme darstellen,

sondern die Verantwortung der Entwickler unterstreichen. Mark Weiser vergleicht in diesem Zusammenhang das *Ubiquitous Computing* mit den Gefahren, die sich bei jeder Nutzung von Technologien ergeben:

> *„[...] Meiner Meinung nach war die Nutzung von Technologien immer mit Gefahren verbunden. Das Internet kann dazu benutzt werden, Leute übers Ohr zu hauen; Strom wird dazu verwendet, Leute zu exekutieren. Auch Metall ist ein zweischneidiges Schwert: Man kann es dazu verwenden, einen Pflug herzustellen, oder damit Menschen töten. Natürlich liegt auch im Calm Computing eine Gefahr: Wenn etwas in unser Bewusstsein eindringt, ohne dass wir es merken. Das ist auch bei der unterschwelligen Werbung der Fall, wo eine geheime Botschaft durchdringen soll. Und das ist ziemlich übel, keine Frage.“* – Weiser (1999)

Neben den bisher erläuterten Fähigkeiten allgegenwärtiger Computer stellt es einen essentiellen Gesichtspunkt dar, dass das System entsprechend dem aktuellen Kontext handelt. Mit dieser Eigenschaft von Anwendungen befasst sich der Begriff *Context Awareness* und soll im folgenden Abschnitt genauer erläutert werden.

Context Awareness

Damit Computer den Menschen beim Erledigen von alltäglichen Aufgaben angemessen behilflich sein können, sollten sich diese dem aktuellen Bezugsrahmen anpassen und daraufhin situationsbedingt reagieren. Ein System mit diesen Fähigkeiten muss dementsprechend die Umgebung untersuchen und auf Änderungen innerhalb dieser reagieren können (Schilit u. a. (1994)). Vor diesem Hintergrund erstellen die Anwendungen einen aktuellen Kontext, wobei zum einen die Berücksichtigung äußerer Einflüsse und zum anderen die Informationsgewinnung aus der Kommunikation zwischen der Person und seiner Umgebung, einschließlich der ihn umgebenen Computer, eine wesentliche Rolle spielen. Letztere Kontexterfassung sollte dabei von der zwischenmenschlichen Kommunikation inspiriert werden, damit der Mensch seine Ausdrucksform zur Kontextvermittlung gegenüber Computern nicht verändern muss.
Mit dem Ziel, einen solchen Vermittlungsprozess für den Anwender zu ermöglichen, soll im ersten Schritt eine Eigenschaft von Kommunikationen beleuchtet werden, die im Bereich des Zwischenmenschlichen gültig ist. Hierbei handelt es sich um das in Watzlawick u. a. (2007) vorgestellte vierte metakommunikative Axiom:

> *„Menschliche Kommunikation bedient sich digitaler und analoger Modalitäten. Digitale Kommunikationen haben eine komplexe und vielseitige logische Syntax, aber eine auf dem Gebiet der Beziehungen unzulängliche Semantik. Analoge Kommunikationen dagegen besitzen dieses semantische Potential, ermangeln aber die für eindeutige Kommunikationen erforderliche logische Syntax.“*

Dieses Axiom beschreibt, dass bei der zwischenmenschlichen Kommunikation neben dem gesprochenen Wort ebenfalls die Körpersprache, die Sprechweise sowie der gesamte Kontext von Bedeutung sind. Die analoge Kommunikation wird somit zur Verdeutlichung der sprachlichen und inhaltlichen Aussage verwendet und sollte nicht im Widerspruch zur digitalen Kommunikation stehen. Weiterhin wird in Watzlawick u. a. (2007) im Hinblick auf den zwischenmenschlichen Bezugsrahmen geschrieben:

> *„[...] Dies gilt natürlich für jeden zwischenmenschlichen Kontext; jede Mitteilung wird zu einem Bestandteil des Kontextes und bedingt die nachfolgenden Interaktionen.“*

Infolge dieser Auswirkungen der Kommunikation auf den Kontext ist es erstrebenswert kontextsensitive Systeme zu entwickeln, welche die Fähigkeit besitzen, sowohl digitale als auch analoge Modalitäten zu erfassen und zu interpretieren. Zum aktuellen Zeitpunkt kann die Mensch-Computer-Interaktion eine zwischenmenschliche Kommunikation jedoch nicht vollständig abbilden. Ebenso ist es Computern nicht möglich, die Umgebung analog zum Menschen wahrzunehmen, was folglich eine individuelle Definition des Begriffs „Kontext“ im Bereich der *Context Awareness* erfordert.
Hinsichtlich dessen wird „Kontext“ in Abowd u. a. (1999) als jede Information, die dazu verwendet werden kann eine Situation einer Entität zu charakterisieren, beschrieben. Eine Entität kann dabei eine Person, ein Ort oder ein Objekt, welches relevant für die Interaktion zwischen dem Benutzer und der Applikation ist, inklusive derer selbst, darstellen.

Zur Gewinnung dieser beschreibenden Kontextinformationen setzen *Context-Aware Systeme* verschiedenartige Sensoren ein, mit dem Ziel möglichst viele Daten über die Umgebung zu erfassen. Die unbearbeiteten Sensordaten werden daraufhin in einer Kaskade von Prozessen mit weiterem Wissen angereichert, um letztlich als ganzheitlicher Kontext verwendet zu werden. Dieser Vorgang wird in der Regel als *Context-Reasoning* bezeichnet. Für das Reasoning werden die Rohdaten der Sensoren als „Low-Level“ Kontext betrachtet, der in höheren Level mit weiterem Wissen über die Umwelt bestückt wird, mit der Absicht einen „High-Level“ Kontext (konkrete Situationen) zu erstellen, vgl. Bettini u. a. (2010).
Zur Orientierung, welche Informationen zur Bestimmung eines präzisen Bezugsrahmens gesammelt werden sollten, lässt sich das in Jang und Woo (2003) vorgestellte Modell zur Repräsentation von Kontexten verwenden. Es sieht vor, dass die folgenden Fragewörter mit Wissen gefüllt werden: *Who*, *What*, *Where*, *When*, *Why* und *How*.

Ein weiterer Gesichtspunkt, den es bei der computergestützten Kontexterfassung zu Berücksichtigen gilt, sind die spezifischen Charakteristiken, die Informationen dieser Form besitzen. Aus diesem Grund sollen einige dieser Eigenschaften im Folgenden vorgestellt werden, vgl. Henricksen u. a. (2002).
Kontextinformationen besitzen eine Reihe zeitlicher Merkmale. In einem ersten Schritt lassen sich die Daten im Hinblick auf die Zeit zwischen *statisch* und *dynamisch* unterscheiden.

Erstere beschreiben hierbei invariante Aspekte, wie den Geburtstag einer Person, während sich dynamische Kriterien, wie der soziale Status oder die Position eines Menschen, ändern können. Des Weiteren verändern sich Kontexte kontinuierlich, was bedeutet, dass vergangene, aktuelle und zukünftige Kontextinformationen zu betrachten sind und daher Kontexte entlang einer Zeitachse beschrieben werden können. Die Relevanz der Daten kann dabei bezüglich ihrer Entfernung zu einem bestimmten Zeitpunkt variieren (Tarasewich (2003)). Beispielsweise kann der Aufenthaltsort von Personen in einem Areal vor wenigen Minuten aussagekräftiger sein als der Fakt, dass sich bereits vor Tagen Personen in diesem Bereich befanden. In weiteren Situationen könnte wiederum das Gegenteil aufschlussreicher sein.
Ein zusätzliches Attribut, das Kontextdaten zugeschrieben werden kann, ist das häufige Rauschen innerhalb der Datensätze. Dieses kann darin begründet liegen, dass Sensorwerte zu einem bestimmten Zeitpunkt nicht vorliegen oder inkorrekte Informationen in den Bezugsrahmen einfließen, weil die Sensoren einen fehlerhaften Zustand der Welt reflektieren.
Geschuldet der Tatsache, dass die Ableitung von Kontexten in der Regel mit Hilfe von Werten verschiedenartiger Sensoren stattfindet, muss die Heterogenität zwischen den Daten gegebenenfalls durch eine Transformation in ein benötigtes Format aufgelöst werden. *Context-Aware Systeme* erfordern daher entweder eine zentrale Einheit, die in der Lage dazu ist mit sämtlichen Teilnehmern zu kommunizieren oder ein einheitliches Protokoll unter dessen Einhaltung die Sensorwerte im Netzwerk publiziert werden können.

Nachdem die beteiligten Anwendungen den aktuellen Kontext detektiert haben, müssen sie sich diesem adäquat anpassen. Hierzu kennt jeder Aktor sein zu verwendendes Verhalten bezüglich definierter Bedingungen, sodass die Komponenten eigenständig auf den Kontextwechsel reagieren können. Anwendungsbeispiele für Reaktionen eines *Context-Aware Systems* ist die automatische Anpassung der Schriftgröße eines Handys bei stattfindenden Bewegungen oder das ausschließliche Durchstellen wichtiger Benachrichtigungen, sollte sich die Person in einem Meeting befinden (Schmidt u. a. (1999)). Ebenso lassen sich Systeme dieser Art in der Gesundheitspflege finden. Hier passen sich die vom Personal genutzten Computer in der Form dem Kontext an, dass sie an einem Krankenbett ausschließlich die zu dem jeweiligen Patienten bedeutsamen Aufzeichnungen anzeigen (Bardram (2004)). Zudem finden kontextsensitive Applikationen zunehmend Einzug in private Wohnbereiche, indem beispielsweise den Bewohnern das Bild einer Videokonferenz auf dem nächstgelegenen Bildschirm präsentiert wird oder sich das Licht automatisch ausschaltet, falls sich keine Person im entsprechenden Bereich befindet.
Darüber hinaus befasst sich eine weitere Form kontextsensitiver Systeme mit persönlichen *Companion-Systemen* (Biundo und Wendemuth (2010)). Diese Anwendungen beziehen die im Vorfeld über den Menschen vorliegenden Informationen intensiv in den Kontext ein, mit dem Ziel, konsequent und vollständig individuell auf die Person als partnerschaftlicher Dienstleister zu reagieren. Hierzu werden in den Anwendungen interne Modelle des Nutzers gespeichert, damit diese ihr Verhalten anhand seiner Fähigkeiten, aktueller Bedürfnisse und seiner emotionalen Befindlichkeit ausrichten können.

Smart Home

Im Bereich der *Smart Homes* wird das zukünftige Leben der Menschen in ihrem Eigenheim erforscht und versucht die Wohnräume mit Intelligenz auszustatten, indem „[...] die zahlreichen Geräte der Hausautomation (wie Heizung, Beleuchtung, Belüftung), Haushaltstechnik (wie z. B. Kühlschrank, Waschmaschine), Konsumelektronik und Kommunikationseinrichtungen zu intelligenten Gegenständen werden, die sich an den Bedürfnissen der Bewohner orientieren.“ (Strese u. a. (2010)).
In Strese u. a. (2010) werden vier Marktsegmente identifiziert, die sowohl einzeln als auch in Kombination die Motivation zur Entwicklung eines *Smart Home* darstellen können.

- **Energiemanagement:** Zur Wahrung des Klimaschutzes und aus Kostengründen ist es bedeutungsvoll, dass private Wohnräume eine hohe Energieeffizienz dadurch erreichen, dass jeder unnötige Verbrauch von Energie vermieden wird. Hier können intelligente Wohnbereiche einen erheblichen Beitrag leisten, indem u.a. die Heizung oder der Stromverbrauch gesenkt wird, sollte sich keine Person im Haus befinden. Außerdem besitzt ein *Smart Home* durch die verschiedenen Interaktionsmöglichkeiten die Fähigkeit, die Bewohner spielerisch zu einem niedrigen Energieverbrauch zu animieren. Vor diesem Hintergrund lässt sich die „SmartGauge™ with EcoGuide“[1] des Automobilherstellers Ford als Leitbild verwenden, in der ein Baum bei einem ökonomischem Fahrstil wächst und gedeiht.
- **Ambient Assisted Living:** Dieser Teilbereich des intelligenten Wohnens befasst sich mit der Bereitstellung von Konzepten, Produkten und Dienstleistungen, mit Hilfe derer die Lebensqualität und die Sicherheit älterer Menschen in ihren privaten Wohnbereichen computergestützt verbessert werden kann. Das Ziel dieser Disziplin liegt somit darin, den Menschen möglichst lange ein selbstbestimmtes Leben in ihrer gewohnten Umgebung zu ermöglichen. Hier werden beispielsweise die Vital- und Bewegungsdaten überwacht, sodass Notsituationen bemerkt werden und gegebenenfalls eine Alarmierung in Abhängigkeit der Schwere der erkannten Situation erfolgen kann (Hansen und Meissner (2007)).
- **Komfort:** Ein weiterer positiver Aspekt eines *Smart Home* liegt im erhöhten Komfort, der sich durch die Unterstützung der Computer ergibt. In diesem Zusammenhang ist es denkbar, dass die Wohnung selbstständig die Heizung einschaltet, falls die Raumtemperatur sinkt oder die Belüftung aktiviert, sollte sich die Raumluftqualität verschlechtern. Des Weiteren kann das Ambiente automatisch dem Kontext angepasst werden, sobald der Bewohner aufgrund seiner Aktivitäten eine Arbeits- oder Entspannungsatmosphäre schafft.

[1] http://media.ford.com/images/10031/SmartGauge.pdf

- **Sicherheit:** Die vernetzten Computer in einem intelligenten Wohnbereich können in mehrerer Hinsicht dazu beitragen, das Sicherheitsbedürfnis der Bewohner zu befriedigen. Neben der Überwachung des Menschen zu gesundheitlichen Zwecken, lässt sich zudem die Sicherheit des Wohnraumes rechnergestützt verstärken. Hier lassen sich z. B. Rauchmelder einsetzen die automatisch Hilfe anfordern, falls der Bewohner schläft oder nicht im Haus ist. Darüber hinaus lässt sich die Anwesenheit von Menschen simulieren, um einem unbefugten Zugriff auf den Wohnbereich vorzubeugen.

Zur Verwendung der in den Segmenten vorgestellten Technologien ist es erforderlich, dass innerhalb der Wohnbereiche entsprechende Rahmenbedingungen geschaffen werden. Hinsichtlich dessen wird in Glasberg und Feldner (2009) ein Leitfaden für eine Heimvernetzung vorgestellt. Hier werden exemplarisch Anwendungen und Geräte eines intelligenten Haushalts in Anwendungsbereiche eingeteilt und zur Vernetzung dieser auf dem Markt befindlichen Netzwerktechnologien vorgestellt.

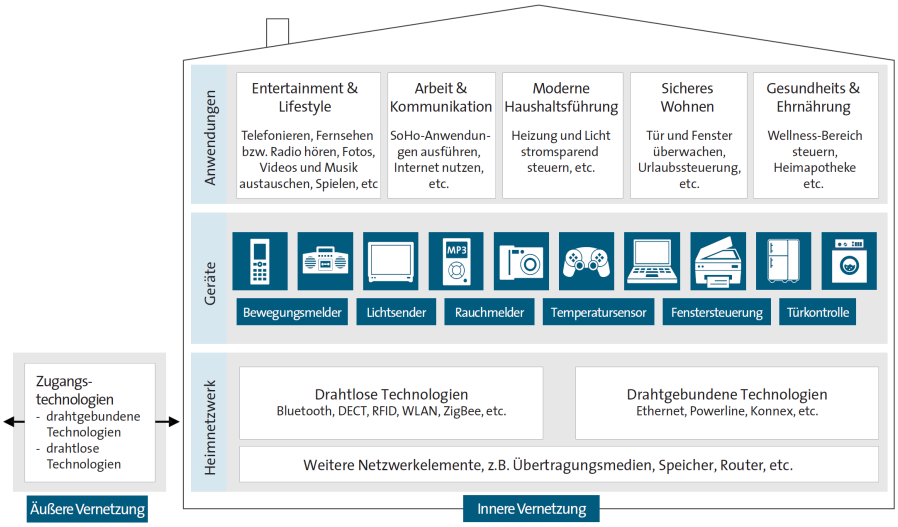


Abbildung 2.1: Übersicht verschiedener Themenfelder bei der Heimvernetzung (Glasberg und Feldner (2009))

Die Entwicklung unterstützender Systeme in *Smart Homes* unterscheidet sich dabei entscheidend von den rechnergesteuerten Mechanismen am Arbeitsplatz, vgl. Meyer und Rakotonirainy (2003). Dieses liegt darin begründet, dass sich u.a. soziale Aktivitäten in privaten Wohnräumen gewöhnlich von denen im Arbeitsumfeld unterscheiden. Beispielsweise findet die Kommunikation unter Mitarbeitern formaler, strukturierter und aufgabenorientierter statt als unter Familienangehörigen. Außerdem entscheiden Menschen daheim eigenständig darüber wie sie die verfügbare Zeit organisieren, welche Aktivitäten sie wann, wo und wie oft ausführen. Des Weiteren besitzen Unternehmen in der Regel Administratoren, die mit

dem technischen Hintergrund der Systeme vertraut und somit in der Lage sind, grundlegende Änderungen an Komponenten vorzunehmen. Da ein solcher Systembetreuer in privaten Wohnräumen häufig nicht existiert, wird hier besonderes Augenmerk darauf gelegt, selbstkonfigurierende und intuitiv zu bedienende Systeme zu integrieren.

Mit dem Ziel, die intelligenten Objekte in einem *Smart Home* in der Form zu konfigurieren, dass sie den Bewohner ohne eine Bevormundung optimal im Hinblick auf seine Präferenzen unterstützen, wird in Streitz u. a. (2005) zwischen zwei möglichen Arbeitsweisen von Anwendungen und Objekten unterschieden. Zum einen handelt es sich hier um die *system-oriented, importunate smartness*, in der die Elemente autark Aktionen auf Basis der gesammelten Informationen ausführen. Das System entscheidet folglich selbständig und proaktiv, welche Reaktionen die Umgebung auf den Kontext folgen lässt. Ein Wohnbereich dieser Form kann z. B. die Heizung als Konsequenz auf einen Abfall der Temperatur einschalten, ohne dass der Bewohner aktiv geworden ist.
Die zweite Funktionsweise von intelligenten Objekten wird als *people-oriented, empowering smartness* bezeichnet. Dieser Ansatz erhebt und aggregiert zwar ebenso Daten über den aktuellen Zustand der Umgebung, bereit diese allerdings so auf, dass der Bewohner diese verstehen kann und die nachfolgenden Reaktionen des Systems in Eigenverantwortung bestimmt. Diese Systeme liefern den Benutzern daher Vorschläge über das mögliche Verhalten auf der Basis des aktuellen Kontextes, beziehen den Anwender jedoch vor jeder Aktion ein. Das Gesamtsystem kann den Nutzer demzufolge beispielsweise darüber informieren, dass ein Fenster geöffnet und die Heizung eingeschaltet ist und aufgrund seiner Informationen über die Raumtemperatur darauf hinweisen, welche Situation geändert werden sollte.

Die oben beschriebenen Aspekte zeigen, welche speziellen Anforderungen an die Geräte und Interaktionskonzepte zur Entwicklung intelligenter Wohnräume gestellt werden. Aus diesem Grund bildet das Gebiet der *Smart Homes* einen aktuellen Forschungsbereich, in dem kontinuierlich potenzielle Lösungen zu den hier vorgestellten Marktsegmenten erarbeitet werden. Der Erfolg dieser Ansätze ist dabei eng an die Berücksichtigung der in diesem Kapitel vorgestellten Aufgaben zur Realisierung kontextsensitiver Anwendungen geknüpft.

3 Analyse

Mark Weiser hatte bereits im Jahre 1991 eine genaue Vorstellung davon, wie ein *Smart Home* seine Bewohner bei alltäglichen Aufgaben behilflich sein könnte. Hierzu beschreibt er in Weiser (1991) innerhalb eines Szenarios einen Vormittag von Sal und zeigt verschiedene Interaktionen zwischen ihr und den sie umgebenden Computern auf, mit der Absicht ein futuristisches Bild eines Bewohners in seiner Wohnung zu präsentieren.
Inspiriert von dieser Vision des zukünftigen Lebens werden bereits seit Jahren diverse Forschungsprojekte durchgeführt. Dabei zeigen Projekte wie das *House_n*[1] des Massachusetts Institute of Technology (MIT), das *iHomeLab*[2] aus Luzern sowie das Frauenhofer *inHaus*[3], dass dieses Forschungsthema ein großes Interesse hervorruft und es dadurch stetig neue Innovationen gibt.

Einen bedeutenden Teil der zur *Context Awareness* benötigten Kontexterstellung nimmt in diesem Zusammenhang das Erkennen der aktuellen Tätigkeit des Menschen ein, das sog. *Activity Recognition*. Dieses kann beispielsweise das Aufstehen aus dem Bett oder der Gang von einem Teil der Wohnung in einen anderen sein. Zur Erkennung dieser Aktivitäten werden in der Regel die Daten verschiedener Sensoren eingesetzt. In Bao und Intille (2004) wurden Testpersonen mit Beschleunigungssensoren versehen, mittels welcher einige alltägliche Aktivitäten, unter bestimmten Annahmen, mit einer über 80%igen Wahrscheinlichkeit erkannt werden konnten. Weitere Sensoren, die zur Erkennung einer Aktivität beitragen können, sind u.a. Kameras, Audiosensoren und Positionssysteme.
In dieser Arbeit wird die Aktivitätserkennung eines Menschen innerhalb eines Wohnraumes weiterhin eine tragende Rolle spielen. Der Fokus wird dabei auf der modellunabhängigen Interpretation von rohen Sensordaten liegen. Außerdem werden die weiteren Schritte einer *Activity Recognition* diskutiert und aufgezeigt.
Zur weiteren Erläuterung des Kontextes dieser Arbeit und zur Betrachtung sich möglicherweise ergebenden Aktivitäten, werden im folgenden Teil drei Szenarien vorgestellt.

[1] http://architecture.mit.edu/house_n/
[2] http://www.ihomelab.ch/index.php?id=14&L=0
[3] http://www.inhaus.fraunhofer.de/

3.1 Szenarien

Im Folgenden werden verschiedene Szenarien betrachtet, die sich mit den Morgenroutinen der Bewohnerin Sal in ihrem *Smart Home* beschäftigen. Im ersten Schritt soll beschrieben werden, wie die Wohnung eigenständig, falls notwendig, die richtige Weckzeit berechnet, bevor zwischen drei verschiedenen Morgenszenarien nach dem Aufstehen unterschieden wird. In der Beschreibung der ersten beiden Anwendungsfälle wird Sal um 7:10 Uhr von ihrem Wecker geweckt, da ihr erstes Meeting um 8:30 Uhr außerhalb ihrer Geschäftsstelle stattfinden wird. Für das dritte Szenario muss keine Weckzeit erstellt werden, da Sal an diesem Tag keine für den Wecker relevanten Termine hat. Die Weckzeit der ersten beiden Szenarien wird dabei automatisch dadurch generiert, dass ihr Wecker beim persönlichen Kalenderagenten den Beginn und Ort des ersten Termins des Tages erfragt. Über eine Befragung weiterer Informationsträger kann ermittelt werden, dass es sich um einen sonnigen Morgen handelt und die äußerlichen Bedingungen somit nicht auf eine längere Anreisezeit hinweisen. Gegenteiliges würde starker Regen oder Schnee bewirken und wäre somit in die Weckzeit einzubeziehen. Außerdem kann durch die Erhebung der aktuellen Verkehrsbedingungen festgestellt werden, dass auf der Strecke zum Meeting stockender Verkehr herrscht. Diese Tatsache führt dazu, dass die Alaramzeit um 10 Minuten vorgezogen wird und initiiert eine Übertragung des Fahrplans mittels öffentlicher Verkehrsmittel, sodass Sal selbst entscheiden kann, ob sie mit dem Auto anreist.
Nach dem Aufstehen kann sich nun eines der drei folgend beschriebenen Szenarien ergeben.

1. **Entspannter Morgen + Arbeitstag:** Sal steht direkt nach dem ersten Weckereignis auf und begibt sich in das Badezimmer, mit der Absicht sich die Zähne zu putzen und sich kurz frisch zu machen. Daraufhin nimmt sie ein Duschbad, bevor sie das Bad verlässt und sich in der Küche ihr Frühstück zubereitet. Nachdem sie dieses im Essbereich der Wohnung zu sich genommen hat, informiert sie sich mit ihrem Tablet-PC über die wichtigsten Nachrichten und Geschehnisse des Tages, woraufhin sie ihre Wohnung verlässt, um mit dem Auto zu ihrem ersten Meeting zu fahren.

2. **Hektischer Morgen + Arbeitstag:** Nachdem das erste Weckereignis aufgetreten ist, hat Sal sich dazu entschieden später als zur der vom Wecker generierten Zeit aufzustehen. Diese Entscheidung führt dazu, dass sie nicht jede der zur normalen Morgenroutine gehörenden Aktionen ausführen kann. An einem hektischen Morgen verzichtet sie deshalb auf die Dusche und wäscht sich lediglich nach dem Zähne putzen am Waschbecken. Das Frühstück fällt in diesem Szenario ebenfalls kürzer aus, da sie hier auf das Essen im Essbereich verzichtet und lediglich einen Kaffee zu sich nimmt. Anschließend verlässt sie das Haus und reist mit öffentlichen Verkehrsmitteln zu ihrem ersten Meeting, wodurch sie etwas Zeit gegenüber der Anreise mit dem Auto einsparen kann, da stockender Verkehr auf der Strecke herrscht.

3. **Morgen am Wochenende:** An einem freien Tag steht Sal erst dann auf, wenn sie ausgeschlafen ist. Sie kann nun in aller Ruhe ihre Morgenroutine inkl. Zähne putzen, Duschbad und ausgiebigem Frühstück durchführen. Danach verlässt sie nicht direkt das Haus, sondern entspannt sich in ihrer Wohnung, indem sie auf der Couch ein Buch liest oder Fernsehen schaut.

3.2 Anforderungsanalyse

Die Realisierung der beschriebenen Szenarien erfordert das Erkennen einer Reihe verschiedener Kontexte und damit verbundener Aktivitäten. Zu diesem Zweck muss im ersten Schritt definiert werden, welche Daten mittels Sensoren zu erheben sind und somit in den Kontext einfließen. Die Zusammenführung der einzelnen Informationen findet daraufhin in einer Kaskade von Prozessen statt, in der sie mit dem Ziel analysiert und interpretiert werden, entweder als Informationsquelle für weitere Komponenten zur Verfügung zu stehen oder direkt eine Reaktion eines Aktors auszulösen.
Der Prozess der Kontexterstellung ist somit von einem Informationsaustausch zwischen den Komponenten im System abhängig. Für diese Kommunikation unter den Teilsystemen ist es deshalb sinnvoll, eine Architektur bereitzustellen, die ein effizientes Versenden von Nachrichten ermöglicht, offene Schnittstellen zur Integration neuer Komponenten in das Gesamtsystem zur Verfügung stellt und entsprechende Mechanismen zur Anreicherung bestehender Kontextinformationen anbietet. Diese Aspekte werden in Abschnitt 3.2.2 aufgegriffen, um anhand dieser die Services zu identifizieren, die von einer Architektur bereitgestellt werden sollten. Zur Ermittlung dieser Dienste soll zunächst in 3.2.1 diskutiert werden, welche Herausforderungen sich bei der Entwicklung von *Context-Aware Systemen* aus der Sicht eines Entwicklers ergeben. Außerdem wird in diesem Kapitel diskutiert, welche Sensoren als „Context Provider“ zur Realisierung der in 3.1 vorgestellten Szenarien dienen könnten und wie eine Kontextinterpretation strukturiert werden sollte.

3.2.1 Herausforderungen bei der Kontexterstellung

Bei der Implementierung von kontextsensitiven Anwendungen und der damit verbundenen Kontextmodellierung ergeben sich für den Entwickler einige Herausforderungen, die es zu bewältigen gilt, falls er neue Applikationen in ein bestehendes System einbinden muss. Einige dieser Aspekte werden in Chen (2004) sowie Satyanarayanan (2001) beschrieben und sollen nun präsentiert werden.

Kontext Repräsentation

Zur computergestützten Verarbeitung der erlangten Sensordaten muss der Entwickler die Daten in die entsprechenden Datenstrukturen transformieren. Außerdem muss er in Erfahrung bringen, welche Informationen der Kommunikationspartner zur Weiterverarbeitung benötigt. In diesem Zusammenhang sind neben den erfassten Daten über die Umgebung, beispielsweise additiv zu nennende zeitliche, räumliche und benutzerbezogene Fakten, bezüglich der erhobenen Werte vorstellbar.

Wissensaustausch

Context-Aware Systeme besitzen häufig eine offene und dynamische Infrastruktur, in der einzelne Komponenten ihr Wissen mit anderen Teilnehmern im Netz austauschen. Falls eine Anwendung ihre Informationen für weitere Applikationen zur Verfügung stellen möchte, muss sowohl der Speicherort von Kontexten als auch der zur Veröffentlichung der Kontextdaten zu verwendende Mechanismus bekannt sein. Gleiches gilt für das Abfragen von Daten.

Service Identifikation

In *Smart Homes*, in denen verschiedene kontextsensitive Systeme zum Einsatz kommen, werden diverse Services innerhalb des Sensornetzwerkes angeboten. Hier besteht die Herausforderung darin, nützliche Dienste im Netzwerk zu identifizieren und ihren Anbieter zu kontaktieren. Sollte ein Entwickler die Existenz eines bereits implementierten Service nicht zur Kenntnis nehmen, kann dieses zu einer unnötigen Mehrarbeit führen. Darüber hinaus kann es Inkonsistenzen im System hervorrufen für den Fall, dass zwei Dienste aktiv sind die zwar eine identische Aufgabe erfüllen sollen, allerdings aufgrund verschiedener Operationen unterschiedliche Ergebnisse liefern.

Frequenz der Publikation von Kontextinformationen

Eine Schwierigkeit im Umgang mit Kontextdaten liegt darin, zu entscheiden, zu welchem Zeitpunkt ein Versand der vorliegenden Informationen an weitere Komponenten durchzuführen ist. Zum einen soll das Netzwerk nicht mit unbedeutendem Datentransfer belastet werden, da dieses einen wertlosen Overhead zur Folge haben würde. Zum anderen kann sich eine langanhaltende Zurückhaltung bedeutender Informationen negativ auf die Reaktionszeit des Systems auswirken. Aus diesem Grund stellt es eine Herausforderung dar, ein Gleichgewicht zwischen diesen Aspekten zu schaffen.

Datenschutz

In vielen Situationen spielen die persönlichen Daten eines Anwenders eine tragende Rolle bei der Kontextmodellierung. Diese Daten beinhalten u.a. Informationen über soziale Verhaltensmuster, Aktivitäten und Präferenzen über die Person. Der Umgang mit sensiblen Daten, wie sie hier vorliegen, erfordert vom Entwickler entsprechende Designentscheidungen und die Implementierung geeigneter Sicherheitsmechanismen, sodass der Anwender die Kontrolle über das System behält und eine Möglichkeit zur

Einsicht der erhobenen Daten erhält. Für genauere Informationen über Designempfehlungen bei der Entwicklung von Applikationen dieser Form sei an dieser Stelle an die „Technikfolgen-Abschätzung Ubiquitäres Computing und Informationelle Selbstbestimmung“ (TAUCIS) von Bizer u. a. (2006) verwiesen.

3.2.2 Anforderungen an eine Architektur für Context-Aware Systeme

Aufgrund der verteilten Natur der Anwendungen in einem *Smart Home* sowie den in 3.2.1 beschriebenen Herausforderungen bei der Erstellung von Kontexten, findet in diesem Abschnitt eine Erläuterung der Anforderungen an eine Architektur für Applikationen dieser Art statt. Hierbei sollen zunächst die Aufgabenstellungen beim Entwurf verteilter Systeme im Vordergrund stehen, bevor unter Berücksichtigung dieser näher darauf eingegangen wird, inwiefern eine solche Architektur dem Entwickler von kontextsensitiven Systemen Hilfestellungen in der Handhabung von Kontextinformationen liefern kann.
Coulouris u. a. (2005) präsentieren einige Gesichtspunkte, die es beim Aufbau eines verteilten Systems, wie sie *Smart Homes* darstellen, zu berücksichtigen gilt. In Systemen dieser Form kommen verschiedenartige Netzwerke, Hardware, Betriebssysteme und Programmiersprachen zum Einsatz, was zu einer großen *Heterogenität* innerhalb des Systems führt. Des Weiteren besitzen sie in der Regel eine dynamische Struktur, was bedeutet, dass das System kontinuierlich erweitert wird, indem neue Applikationen integriert oder bestehende Anwendungen erneut implementiert werden. Deshalb müssen die Schlüsselschnittstellen für den Zugriff auf gemeinsam genutzte Ressourcen veröffentlicht und dokumentiert werden. Es wird in diesem Fall von der *Offenheit* des Systems gesprochen.
Einen weiteren bedeutenden Aspekt stellt die Zuverlässigkeit der Anwendung beim Auftreten großer Datenmengen und somit die *Skalierbarkeit* dar. Eine Anwendung wird als skalierbar bezeichnet, wenn es bei einem wesentlichen Anstieg der Ressourcen und deren Kommunikation weiterhin effektiv arbeitet. Sollte es dennoch zu Engpässen in der Performance kommen, sollten entsprechende Prozesse angestoßen werden, damit elementare Daten weiterhin ihre Empfänger erreichen. Entsprechende Themen, die sich mit diesen Mechanismen befassen, werden unter dem Begriff „Quality of Service“ (QoS) zusammengefasst. Verschiedene QoS-Verfahren werden in Welzl (2005) vorgestellt.
Eine weitere Eigenschaft verteilter Systeme liegt in der *Nebenläufigkeit*. In verteilten Anwendungen werden Ressourcen bereitgestellt, die von weiteren Teilnehmern gemeinsam genutzt werden können, wodurch es zu einem gleichzeitigen Zugriff kommen kann. Deshalb muss für gemeinsam genutzte Objekte sichergestellt werden, dass sie in einer nebenläufigen Umgebung korrekt arbeiten, indem ihre Operationen synchronisiert werden und die Daten konsistent bleiben.
Ergänzend zu den bisher genannten Merkmalen kann es insbesondere in verteilten Systemen zu Ausfällen von Komponenten kommen. Aus diesem Grund spielt die Strategie zur

Fehlerverarbeitung in diesen Systemen eine tragende Rolle. In diesem Zusammenhang liegt die grundsätzliche Schwierigkeit darin, die Fehler zu erkennen und daraufhin eine geeignete Verarbeitung dieser vorzunehmen, sodass lediglich die von der Komponente beeinflusste Arbeit betroffen ist, während die Verfügbarkeit des Gesamtsystems weiter gewährleistet wird.

Die Herausforderungen bei der Entwicklung von *Context-Aware Systemen* und die Eigenschaften sowie die Aufgaben beim Entwurf eines verteilten Systems sollen nun dazu verwendet werden, Anforderungen an eine Architektur für Systeme, wie sie intelligente Wohnräume darstellen und die bisher vorgestellten Merkmale vereint, vorzustellen. Hierzu sollen einige aus Dey u. a. (2001) beschriebene Gesichtspunkte näher betrachtet und um einige Aspekte erweitert werden.

- **Trennung der Zuständigkeiten:** Die Interpretation der Sensorwerte sollte von den einzelnen Anwendungen getrennt werden. Andernfalls würde es dazu führen, dass sich jeder Entwickler mit den technischen Details der Sensoren auseinander setzen müsste, dessen Werte innerhalb der Applikation Verwendung finden sollen. Außerdem wäre es ohne eine Trennung der Low-Level Sensordaten von den Semantiken der Anwendungen möglich, dass gleiche Sensorwerte in verschiedenen Applikationen unterschiedlich gedeutet werden, was zu einem inkonsistenten Verhalten des Gesamtsystems führen könnte. Eine Trennung zwischen der Erfassung und der Verwendung des Kontextes führt somit dazu, dass Kontextinformationen ohne Wissen über den technischen Hintergrund verwendet werden können und als konsistente Information im System zur Verfügung stehen.
  Die zu entwickelnde Architektur sollte demnach eine Komponente besitzen, die in der Lage dazu ist, rohe Sensordaten aufzubereiten und ggf. eine Vereinigung von Sensordaten vornehmen, damit sie von weiteren Anwendungen in eigene Operationen eingebunden werden können.

- **Kontextinterpretation und Kommunikation:** Für die Anwendungen in einem *Smart Home* ist es unentbehrlich, dass sie über Veränderungen des Kontextes informiert werden. Aus diesem Grund sollten sowohl Mechanismen zum Abfragen als auch zur Auslieferung von Daten offeriert werden. Da sämtliche Anwendungen zu jeder Zeit identische Kontexte verwenden sollen, ist es sinnvoll, eine zentrale Einheit einzusetzen, welche die aktuellen Kontextinformationen zusammenführt und interpretiert. Dabei ist es nützlich, den Interpreter in Teilkomponenten zu gliedern, sodass eine Information auf verschiedenen Abstraktionslevel mit semantischem Wissen über die Umgebung angereichert werden kann, bevor sie zur entsprechenden Applikation gelangt. Eine solche Aufteilung in mehrere Interpretationslevel hat den Vorteil, dass Anwendungen erst dann eine Benachrichtigung erhalten können, falls ein Kontext wie „Sal frühstückt“ erkannt wurde und entsprechend kein Interesse an Daten einzelner Sensoren besteht.

- **Interpretationsmöglichkeiten über die Zeit:** Zur Erstellung der dem System bekannten Kontexte ist ein Zugriff auf vergangene Daten erforderlich. Hier ist es denkbar, dass der Bewohner eines *Smart Homes* aufgrund gespeicherter Daten eigenständig Situationen in das System einträgt. In einem weiteren Schritt sollte die Architektur dazu beitragen, dass der Wohnbereich mittels vergangener Daten selbstständig lernt, welche Kontexte auftreten können.
  Gespeicherte Informationen können ebenfalls mit dem Ziel verwendet werden, Vorhersagen über folgende Bezugsrahmen, wie beispielsweise die Schätzung des zukünftigen Aufenthaltsortes des Bewohners, zu treffen.
- **Flexible Service Anbieter/Konsument-Struktur:** Gesetzt den Fall, dass eine Applikation mit weiteren Komponenten kommunizieren möchte, muss bekannt sein, wer die benötigten Informationen liefern kann, wo derjenige sich im Netzwerk befindet und über welche Protokolle und Mechanismen die jeweilige Komponente anzusprechen ist. Eine Architektur muss aus diesem Grund einen Service bereitstellen, der eine Möglichkeit bietet, Teilnehmer innerhalb des Netzwerkes zu suchen und mit ihnen zu kommunizieren. Ziel einer solchen Service-Discovery-Komponente ist demzufolge die Auflösung der Heterogenität der diversen Geräte und Dienste in ubiquitären Computersystemen.
- **Interaktionsmöglichkeiten mit dem Benutzer:** Der Anwender muss zu jeder Zeit eine Eingriffsmöglichkeit in das System erhalten, damit unerwünschte Reaktionen abgelehnt und aktuell gewünschte Verhaltensmuster des Systems hervorgerufen werden können. Da die Architektur die Möglichkeit besitzt mit allen weiteren Komponenten zu kommunizieren, ist es sinnvoll eine Schnittstelle anzubieten, über welche der Anwender mit dem System interagieren kann. Eine denkbare Variante wäre in diesem Zusammenhang das System über ein mobiles Gerät zu steuern oder eine Gesten- und Sprachsteuerung zu implementieren.

Aus den beschriebenen Anforderungen lassen sich bereits einige Komponenten identifizieren, die eine Architektur für *Context-Aware Systeme* in einem *Smart Home* enthalten sollte. Diese werden in der Abbildung 3.1 demonstriert.

3.2.3 Context Provider

Der zur Realisierung der Szenarien erforderliche Kontext soll ein möglichst exaktes Abbild der aktuell herrschenden Bedingungen in der Umgebung liefern. Die Präzision dieses Abbildes hängt dabei von der Anzahl und der Korrektheit der erfassten Daten ab. Aus diesem Grund müssen verschiedene „Context Provider“ zum Einsatz kommen, die mit ihren Low-Level Sensordaten den High-Level Kontext beeinflussen können. Welche Informationsquellen für den Aufbau eines *Smart Home* von Interesse sind, wird im folgenden Abschnitt

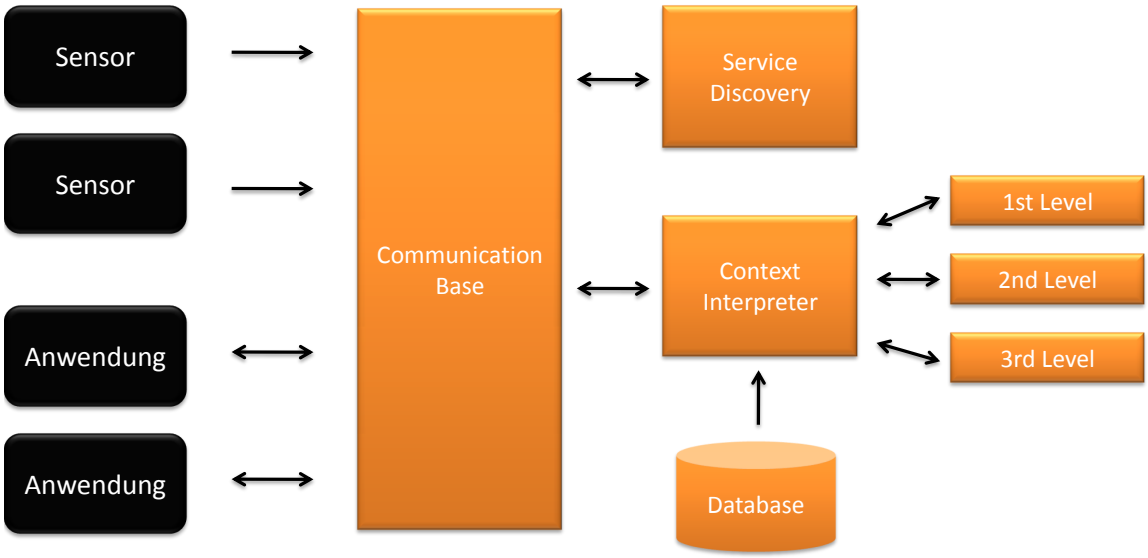


Abbildung 3.1: Überblick der aus den Anforderungen abgeleiteten Komponenten der Architektur

diskutiert.
In Gu u. a. (2004) werden „Context Provider“ in die Kategorien *externe* und *interne* Informationsquellen aufgeteilt. Dabei werden beispielsweise aktuelle Wetterinformationen als externe Quelle bezeichnet, während interne Informationsträger Daten über den Zustand innerhalb der Wohnumgebung, wie z.B. die aktuelle Raumtemperatur, liefern. Letztere Informationen stammen folglich aus dem Sensornetzwerk des *Smart Home*. Die Daten interner Quellen können außerdem Aufschluss darüber geben, welche Aktivität der Bewohner durchführt und wie sich sein soziales Umfeld zum aktuellen Zeitpunkt darstellt.
Diese Aufteilung wird im nächsten Schritt mit der Absicht verwendet, Sensoren zu ermitteln, deren Daten im Hinblick auf die Szenarien von Nutzen sein könnten.

Interne Context Provider

In Schmidt u. a. (1998) wird eine Reihe von Sensortypen vorgestellt, die sich als interne „Context Provider“ eignen. Einige dieser Technologien sollen in diesem Abschnitt kurz vorgestellt und um einige Informationen ergänzt werden.

- **Optisch:** Ein einzelner optischer Sensor kann dazu verwendet werden, die Lichtintensität oder den Typ des Lichts (z.B. Sonnenlicht, Typ des künstlichen Lichts) zu messen.

Mit Hilfe von Kameras, die als eine Ansammlung von optischen Sensoren verstanden werden können, ist es zusätzlich möglich, Objekte in einem Raum zu erkennen und zu verfolgen. Hierzu werden auf die gesammelten Bilder Algorithmen, wie der „Scale Invariant Feature Transform (SIFT)“ (Lowe (1999)) mit dem Ziel angewandt, Objekte unabhängig von der Beleuchtung und ihrer Lage in Bildern zu finden.

- **Audio:** Audiosensoren können auf der einen Seite dazu beitragen, Informationen über die herrschende Lautstärke und die Art der Hintergrundgeräusche zu gewinnen. Auf der anderen Seite werden Sensoren dieser Form mit der Zielsetzung eingesetzt, eine Sprachsteuerung zu implementieren. Mit der hierzu erforderlichen Spracherkennung in einem *Smart Home* befasst sich Witt (2011).

- **Biologische:** Durch den Einsatz von Biosensoren lassen sich u.a. Körperwerte wie Hautwiderstand, Puls, Blutdruck und Körpertemperatur messen. Diese Sensorwerte dienen u.a. zur Überwachung der Vitalfunktionen und zur Prognose des emotionalen Zustandes des Bewohners. Letzterer Aspekt könnte das Gesamtsystem dahingehend beeinflussen, dass die Reaktionen bei gestresster Stimmung ungleich derer bei entspannter Gemütslage ausfallen. Damit die Person sich nicht eigenverantwortlich die einzelnen Sensoren anlegen muss, ist es sinnvoll, diese in die Kleidung zu integrieren, sodass ein Vorhandensein dieser Technik nicht bemerkt wird. Mit den Herausforderungen bei der Gestaltung von intelligenter Kleidung befasst sich Müller (2010).

- **Bewegung:** In einer intelligenten Wohnumgebung können Bewegungssensoren in mehrerer Hinsicht verwendet werden. Zum einen spielen diese Sensoren eine tragende Rolle bei der gestenbasierten Interaktion mit Computersystemen. Mit der hierzu erforderlichen Interpretation von Bewegungen beschäftigt sich u.a. Voskuhl (2009). Zum anderen lässt sich mittels dieser Sensoren herausfinden, ob in einem Areal des Hauses Bewegungen stattfinden. In einem weiteren Schritt kann daraufhin analysiert werden, um welche Aktivität es sich bei den erkannten Bewegungen handelt.

- **Kapazitiv:** Kapazitive Sensoren besitzen einen flexiblen Anwendungsbereich in intelligenten Wohnräumen. Mittels kapazitiver Näherungssensoren lassen sich Annäherungen metallischer und nicht metallischer Stoffe erkennen. Dementsprechend kann mit Hilfe dieser Technologie erkannt werden, ob sich der Bewohner einem Gegenstand nähert und frühzeitige Maßnahmen, wie das Einschalten eines Anzeigegerätes, initiiert werden. Darüber hinaus lassen sich durch den Einsatz kapazitiver Beschleunigungssensoren Bewegungen von Objekten, wie sie beim Öffnen einer Schublade oder einer Tür auftreten, detektieren. Außerdem können kapazitive Berührungssensoren genutzt werden, um den Standort einer Person oder die Nutzung eines Gegenstandes, wie das Sitzen auf dem Sofa, zu ermitteln. Für eine detaillierte Beschreibung zur Nutzung von kapazitiven Sensoren in Mobiliar und Böden sei an dieser Stelle auf Dreschke (2011) verwiesen.

- **Position:** Positionsdaten bieten eine große Hilfestellung bei der Kontexterstellung, da sich anhand dieser Informationen der aktuelle Aufenthaltsort von Personen und Geräten bestimmen lässt. Die erhobenen Daten geben somit Aufschluss darüber, in welchem Teilbereich eine Reaktion des Systems zu erfolgen hat. Des Weiteren ist eine Verwendung von Positionsdaten für die Schätzung über den zukünftigen Aufenthalt einer Person denkbar. Gängige Sensortypen zur Erhebung von Daten dieser Art sind zum einen Kombinationen aus den bereits vorgestellten Sensoren, wie Lichtschranken, RFIDs und zum anderen eigenständige spezielle Indoor-Positioning Systeme.

Externe Context Provider

Unter externen „Context Providern“ versteht man Informationsquellen, deren Werte nicht aus dem Sensornetzwerk, sondern von außenstehenden Bezugsquellen erfragt werden müssen. Einige dieser Daten und ihre potenzielle Wirkung auf den Gesamtkontext werden in diesem Textabschnitt präsentiert.

- **Zeitinformationen:** Bei der Ausführung einer Aktivität des Bewohners spielen die aktuellen Zeitinformationen eine wesentliche Rolle. Beispielsweise sollte das System beim Aufstehen überprüfen, welche Uhrzeit vorliegt und festlegen, ob der Bewohner bereits in den Tag startet oder zurück ins Bett kommen wird. Diese Entscheidung würde daraufhin die Beleuchtungssituation des Wohnbereichs beeinflussen. Des Weiteren ist es von Interesse, ob der Sonnenaufgang bereits stattgefunden hat. In diesem Fall wäre z. B. eine Beleuchtung der betretenden Räume unnötig.
- **Tagesdaten:** Die über den Tag vorliegenden Fakten sind ein entscheidender Bestandteil des Kontextes. Wie bereits in den Szenarien (3.1) beschrieben wurde, ist zum Beispiel an einem Werktag mit anderen Aktivitäten zu rechnen als am Wochenende. In diesem Zusammenhang gilt es jedoch zu beachten, dass sich trotz eines Werktages die traditionellen Aktivitäten des Wochenendes ergeben können für den Fall, dass es sich um einen Urlaubs- oder Feiertag handelt. Weiterhin ist es denkbar, dass sich an bestimmten Werktagen wiederkehrende Aktivitäten des Bewohners ergeben und eine entsprechende Reaktion des Systems erfordern. Zudem stellt es einen relevanten Gesichtspunkt dar, in welcher Jahreszeit der betrachtete Tag liegt. Ein Anwendungsfall in diesem Zusammenhang ist das Öffnen der Fenster beim Aufstehen. Dieses sollte im Winter so lange zurückgehalten werden bis die Person das Zimmer verlassen hat, während diese Aktion im Sommer direkt beim Aufstehen ausgeführt wird.
- **Außenbedingungen:** Die äußerlichen Bedingungen sind insbesondere für die hier vorgestellten Morgenszenarien von Bedeutung. Aus diesem Grund muss das System dazu in der Lage sein, möglichst viele Informationen über Medien wie das Internet zu

beziehen. Für die Weckzeit eines Bewohners soll mitunter der erste Termin des Tages einbezogen werden. Mit der Absicht eine angemessene Zeit für den Weckruf zu generieren, müssen deshalb Stauinformationen, Wetterbedingungen und Routeninformationen erfragt werden. Zudem wäre es eine mögliche Reaktion des Systems, einen Sonnenaufgang für den Bewohner zu simulieren, falls die Abfrage der Wetterdaten ergibt, dass es draußen regnet.

3.2.4 Kontextinterpretation

Wie bereits in Abschnitt 3.2.2 diskutiert wurde, müssen die Daten der „Context Provider“ so zusammengeführt werden, dass ein Gesamtkontext entsteht. Dabei liegt die Herausforderung darin, die rohen Sensorwerte schrittweise mit semantischem Wissen anzureichern. Dieser Vorgang macht eine Interpretation der Daten erforderlich, weshalb der folgende Teil sich mit dem Begriff „Interpretation“ befassen wird.
Das Wort selbst stammt aus dem Lateinischen und bedeutet „Auslegung“, „Erklärung“. Zur Beschreibung des eigentlichen Prozesses der Interpretation soll ein Zitat der Brockhaus Online-Enzyklopädie[4] verwendet werden:

> *“[...] Gegenüber dem naiven Verstehen, das Voraussetzung der Interpretation ist, zeichnet sich diese durch stetige Reflexion ihrer Bedingungen, ihres Gegenstandes und ihres Vorgehens aus.“*

Eine Interpretation von Sensorwerten beinhaltet folglich die stetige Deutung der vorliegenden Daten unter Einbeziehung der äußeren Einflüsse. Dabei werden die miteinander kombinierten Informationen dahingehend untersucht, ob sie mit einer Kontextaussage aus der vordefinierten Menge aller Kontexte übereinstimmen. Falls keine zutreffende Situation erkannt werden sollte, kann dieses dadurch begründet sein, dass entweder elementare Daten fehlen oder die Kombination der Beobachtungen zu keinem bekannten Kontext passt. Darüber hinaus kann es bei der Interpretation ebenso auftreten, dass mehrere definierte Situationen mit den existierenden Werten übereinstimmen, da eine Ambiguität der Daten vorliegt. Sollte die Kontexterstellung in einen dieser Zustände gelangen, muss eine Strategie zur Auflösung dieser vorliegen. Eine Herangehensweise hierzu wäre, sämtliche aktuelle Kontextinformationen mit der Absicht zu verwerfen, die Erstellung des Bezugsrahmens von neuem zu beginnen. Des Weiteren könnten nicht kategorisierte Daten als „Unbekannt/Sonstiges“ gekennzeichnet werden, mit dem Ziel, diese Situationen bei der Erhebung neuer Informationen erneut zu untersuchen. Alternativ lassen sich Verfahren einsetzen, die trotz der Unsicherheit in den Daten Schlussfolgerungen daraus ziehen, indem sie Hypothesen über die Daten aufstellen. Zur Beschreibung dieser Verfahren sollen zunächst drei der vier Gruppen von Wissen vorgestellt werden, in denen Unsicherheiten auftreten, vgl. Spies (1993).

[4] http://www.brockhaus-enzyklopaedie.de/be2_article.php

Der Bereich des *unscharfen Wissens* befasst sich mit Mengen, in denen Objekte mehr oder weniger stark enthalten sind. Ein Beispiel hierzu wäre die Kategorisierung von „Personen die groß sind“. Eine unscharfe Menge beinhaltet somit Menschen die mehr oder weniger groß sind, weshalb hier Zugehörigkeitsfunktionen eingesetzt werden, welche die graduelle Zugehörigkeit eines Elementes zu einer Gruppe beschreibt. Ein gewichtiger Vertreter zur Modellierung und Verarbeitung von unscharfem Wissen ist die Fuzzy-Set-Theorie. Sie wurde in Zadeh (1965) vorgestellt und befasst sich mit Mengen, deren Elemente mit einer gewissen Wahrscheinlichkeit zu diesen gehören. Hierzu werden Zugehörigkeitsfunktionen, die einen Wert zwischen 0 und 1 besitzen, eingesetzt und den Objekten entsprechend des Grades ihrer Zugehörigkeit zur Menge zugeteilt. Sollte ein Element den Wert 0 erhalten steht dieses für „nicht zugehörig“ und der Wert 1 für „zugehörig“. Zwischen diesen Werten lassen sich zudem beliebig feine Abstufungen umsetzen.

Eine weitere Gruppe von Wissen in denen Unsicherheiten auftreten, sind die *Wahrscheinlichkeitsschlüsse*. In diesem Zusammenhang soll die bedingte Wahrscheinlichkeit genauer betrachtet werden. Hierunter versteht man die Wahrscheinlichkeit, dass ein Ereignis A eintrifft, unter der Voraussetzung, dass bereits ein Ereignis B gilt und wird als $P(A \mid B)$ geschrieben. Ein Verfahren, dass die Intensität von Abhängigkeiten von Variablen auf der Basis von bedingten Wahrscheinlichkeiten angibt, sind Bayes'sche Netze. Hierbei handelt es sich um gerichtete azyklische Graphen, in denen die Knoten Zufallsvariablen darstellen, während die Kanten die Abhängigkeiten zwischen den Variablen abbilden. Die Menge der Zufallsvariablen ist in diesem Fall als die unsichere Wissensbasis zu verstehen. Die Topologie des Netzes spezifiziert die in der Domäne geltenden bedingten Unabhängigkeitsbeziehungen, vgl. Russell und Norvig (2004). Zu diesem Zweck wird jedem Knoten eine bedingte Wahrscheinlichkeitstabelle beigefügt, in der jede Zeile die bedingte Wahrscheinlichkeit jedes Knotens für den bedingten Fall darstellt. Ein bedingender Fall stellt hier eine Wertekombination für die Elternknoten dar. Das Fehlen einer Verknüpfung zwischen zwei Knoten spezifiziert dabei die bedingte Unabhängigkeit zwischen den Zufallsvariablen.
Für die Kontextinterpretation können die Informationen mit Hilfe eines Bayes'schen Netz als bedingte Wahrscheinlichkeiten beschrieben werden. Daraufhin kann die Relevanz der Kontextinformation aus den einzelnen Wahrscheinlichkeiten errechnet werden. Eine Schwierigkeit, die mit diesem Ansatz einhergeht, ist das erforderliche Wissen über jede mögliche Wahrscheinlichkeitsverteilung, welche nicht in jedem Fall bestimmt werden kann. Mit der Verwendung von Bayes'schen Netzen zur Kontextmodellierung befassen sich u.a. Ranganathan u. a. (2004) und Castro und Munz (2000).

Der dritte Typ von Unsicherheit befasst sich mit der *Plausibilität* von Annahmen. Ein Verfahren, welches Informationen zusammensetzt und die Glaubwürdigkeit der Quellen einbezieht, ist die Evidenztheorie von Dampster und Shafter. Diese wurde in Shafer (1976) auf der Basis einer grundlegenden Arbeit von Arthur Pentland Dempster aus dem Jahre 1967 (Dempster (2008)) vorgestellt. Eine Evidenz setzt sich dabei aus dem Vertrauen darauf, dass die Aus-

sage der Quelle wahr ist und der Plausibilität des Auftretens dieses Ereignisses zusammen. Der entscheidende Unterschied dieser Theorie zur Wahrscheinlichkeitsrechnung liegt darin, dass hier weder die Behauptung noch das Gegenteil plausibel sein muss. Aus diesem Grund wird hier nicht mit einem einzelnen Wahrscheinlichkeitswert gearbeitet, sondern einem Intervall, mit Hilfe dessen die Wahrscheinlichkeit definiert wird, dass die Hypothese korrekt oder nicht korrekt ist. Eine Besonderheit dieser Theorie liegt darin, dass bei einer Wahrscheinlichkeit von 60% für das Eintreten eines Ereignisses *A* die restlichen 40% nicht dem Komplement, sondern der Menge aller möglichen Ereignisse zugeschrieben wird. Diese Eigenschaft liegt darin begründet, dass die 40% für das Nichteintreten des Ereignisses *A* entweder falsch oder richtig sein können.

Die vierte Gruppe von Wissen mit Unsicherheiten ist das *Erkennen und Verstehen unvollständiger und verzerrter Muster* und soll in dieser Arbeit nur kurz erwähnt werden, da in diesem Fall die Musterverarbeitung im Vordergrund steht und es nicht direkt um die Wissensverarbeitung geht. In dieser Disziplin werden u.a. neuronale Netze eingesetzt, um Merkmalsmuster zu erkennen, zuzuordnen und zu verknüpfen.

Bisher wurde in diesem Abschnitt der Begriff der Interpretation diskutiert und aufgezeigt, wie mit unsicheren Kontextinformationen umgegangen werden kann. Vor der Untersuchung einer Information mit den vorgestellten Verfahren findet in der Regel eine Sensorfusion (4.3) statt, sodass möglichst ausschließlich relevante und fundierte Daten vorliegen und damit die Unsicherheit innerhalb der Informationen minimal ist. Aus diesem Grund soll im Folgenden ein potenzieller Prozess zur Kontexterstellung betrachtet werden. Vor diesem Hintergrund muss im ersten Schritt eine Systematik für den Umgang mit rohen Sensordaten gefunden werden. Eine grundlegende Maßnahme in diesem Zusammenhang liegt in der Verifizierung, ob es sich bei den erhobenen Informationen um valide Daten handelt, da es bei dem Einsatz vieler Sensoren in der Regel zu inkorrekten Messungen kommt. Wie solche fehlerbehafteten Werte identifiziert werden können und welche Möglichkeiten für die Handhabung dieser Daten bestehen, wird im Abschnitt 4.1 präsentiert. Erst nachdem ein Datensatz „bereinigt“ wurde, sollte eine Interpretation der Werte erfolgen. Diese kann unter Anwendung verschiedener Methoden durchgeführt werden. In dieser Arbeit steht die modellunabhängige, die modellabhängige und die ontologiebasierte Kontextinterpretation im Fokus.

Die modellunabhängige Interpretation von Kontextdaten zeichnet sich dadurch aus, dass sie keine räumlichen und semantischen Informationen über die Umwelt benötigt. Hier werden die vorliegenden Daten miteinander verknüpft und untersucht, ob sich verwertbare Kontextinformationen ergeben. Die hier verwendeten Verfahren arbeiten häufig direkt mit den Rohdaten und können demnach ohne weitere Vorverarbeitung auf die Werte angewandt werden. Sollte die modellunabhängige Interpretation in Kombination mit weiteren Ansätzen zum Einsatz kommen, ist es denkbar, die Identifizierung inkorrekter Werte in diesem Schritt der Datenanalyse durchzuführen. Der Vorteil der mit dieser Form der Interpretation einhergeht, liegt in der Unabhängigkeit von der Umgebung. Dieses ermöglicht es, die genutzten Verfahren

für gleiche Datentypen in weitere *Smart Homes* zu übertragen, ohne Anpassungen vornehmen zu müssen. Ein Nachteil dieses Ansatzes ergibt sich durch die Unbekanntheit der sich im Raum befindlichen Objekte. Zwar lassen sich beispielsweise anhand von Kameradaten generelle Bewegungen im Raum detektieren, allerdings ist aufgrund der vorliegenden Wissensbasis eine Objekterkennung unmöglich. Daher ist es sinnvoll, den modellunabhängigen Ansatz mit weiteren Ansätzen zu kombinieren, damit die Erstellung des Kontextes möglichst exakt durchgeführt wird. Mit der Thematik, inwiefern mittels Low-Level Sensordaten Wissen über Aktivitäten und Verhalten von Menschen generiert werden kann, befassen sich Yang (2009) und Patterson u. a. (2003).

Die modellabhängige Kontextinterpretation nutzt ein Modell der Umgebung, um Schlüsse über den aktuellen Kontext zu ziehen. Hierzu werden zweidimensionale bzw. dreidimensionale Modelle mit dem Ziel verwendet, die Räumlichkeiten und Objekte möglichst exakt aus der Realität abzubilden. Diese Abbildungen stellen zudem Services bereit, die Aufschluss über den Zustand der einzelnen Entitäten liefern und vorverarbeitete Daten mit semantischen Informationen anreichern. Mit der Realisierung einer entsprechenden Abfragesprache für 3D-Gebäudemodelle zur Gewinnung räumlicher Kontextinformationen befasst sich Borrmann u. a. (2006). Somit entsteht u.a. die Möglichkeit, Objekte und ihre räumlichen Relationen zu erkennen. Außerdem lassen sich mit Hilfe einer modellbasierten Interpretation z.B. nicht nur Bereiche erkennen, in denen sich der Bewohner häufig aufhält, sondern es kann exakt definiert werden, ob er sich im Sichtbereich eines Anzeigegerätes befindet, am Herd kocht oder im Essbereich am Tisch sitzt. Eine Schwierigkeit dieser Interpretationsform liegt in der Erstellung des Modells. Insbesondere die Entwicklung von dreidimensionalen Modellen erfordert einen aufwändigen Prozess. Dabei gilt es zu beachten, dass ein Modell eine auf das Wesentliche reduzierte Abbildung der Realität darstellt und es unmöglich ist, jedes Detail der Umgebung zu modellieren. Eine weitere Herausforderung liegt in der Konsistenzwahrung zwischen dem Modell und der realen Welt. Sollte sich die räumliche Anordnung von Mobiliar verändern, ohne dass dieses im Modell vermerkt wird, würden die angebotenen Services falsche Informationen an weitere Komponenten liefern, was zu einem inkorrekten Kontext führen würde.

Ein ontologiebasierter Ansatz kann sowohl zur Interpretation als auch zur Modellierung des Kontextes verwendet werden, vgl. Wang u. a. (2004). Die Kontextmodellierung durch Ontologien hat den Vorteil, dass eine allgemeingültige Wissensbasis erstellt wird, was bedeutet, dass sämtliche Geräte und Services über einen identischen Satz von Kontexten verfügen, während sie miteinander kommunizieren. Zur Kontextinterpretation lassen sich Regeln innerhalb der Ontologie definieren, die eine Aktivität oder Situation auszeichnen. Hierzu werden häufig unter Zuhilfenahme der „Semantic Web Rule Language“ (SWRL)[5] Regeln erstellt, die Prädikate erster Ordnung verwenden und diese mittels boolescher Algebra verknüpfen. Ein Beispiel für eine Regel einer Aktivität „Bewohner kocht“ könnte

[5] http://www.w3.org/Submission/SWRL/

wie folgt lauten: $(?u\ befindetSichIn\ Kueche) \wedge (Herd\ befindetSichIn\ Kueche) \wedge (Herd\ sensorStatus\ ON) \Rightarrow (?u\ situation\ Kochen)$. Diese erstellten Regeln können daraufhin bei einer Änderung der Umgebung mit einem regelbasierten System untersucht und demzufolge eine Aktivität erkannt werden. Der Nachteil beim Einsatz von Ontologien liegt darin, dass für die Untersuchung einer Regel weit fortgeschrittene Kontextinformationen vorliegen müssen. Es ist daher eine Vorverarbeitung der Daten erforderlich. Außerdem benötigt man für die Entwicklung einer Ontologie Experten und verifizierte Quellen, damit korrekte Begriffe eingetragen werden und folglich von den Benutzern gefunden werden können.

Aufgrund der Vor- und Nachteile der einzelnen Ansätze ist es sinnvoll zu analysieren, ob es eine Möglichkeit gibt, diese verschiedenen Herangehensweisen miteinander zu kombinieren, sodass in erster Linie ihre Vorteile zum Tragen kommen und eine zuverlässige Aktivitätserkennung gewährleistet werden kann.

3.2.5 Zeitliche und kausale Abhängigkeiten von Aktivitäten

In den in Abschnitt 3.1 vorgestellten Szenarien führt die Bewohnerin eine Reihe von Aktivitäten aus, die es von der Wohnumgebung zu erkennen gilt. Eine Aktivität wie z.B. das „Frühstücken“ beinhaltet dabei mehrere Aktionen seitens des Bewohners, die dieses Teilszenario charakterisieren. An dieser Stelle sei „Frühstücken“ durch folgende Tätigkeiten definiert: „Küche betreten“, „Kaffee kochen“, „Nahrung verzehren“, „Zeitung lesen“, „Küche verlassen“. Das Erkennen des „Frühstücks“ erfordert dabei die Beschreibung einiger Vorbedingungen, mit Hilfe derer die Aktionen in eine zeitliche Reihenfolge gebracht werden, da u.a. die Bedingung gelten muss, dass „Küche betreten“ vor dem Ereignis „Küche verlassen“ erkannt wurde. Die weiteren Aktivitäten zwischen diesen beiden Ereignissen können hingegen in beliebiger Reihenfolge auftreten und sich darüber hinaus partiell zeitlich überschneiden, da es möglich ist die Zeitung zu lesen und gleichzeitig einen Kaffee zu trinken.
In weiteren Szenarien wäre es jedoch denkbar, dass entweder jede Aktivität vom erfolgreichen Abschluss der vorherigen abhängt und sich so eine starre Sequenz von Aktionen ergibt oder die Ereignisse vollständig unabhängig voneinander sind, sodass die zeitliche Reihenfolge ihrer Ausführung keine Rolle spielt.
Speziell bei voneinander abhängigen Aktivitäten muss klar definiert sein, in welcher zeitlichen Abfolge diese auftreten müssen. Da jedes Ereignis eine bestimmte Zeit lang andauert und somit Zeitintervalle betrachtet werden müssen, können sich verschiedene Relationen zwischen den Aktionen ergeben. Aus diesem Grund wurde in Allen (1983) eine „interval-based temporal logic“ vorgestellt, deren mögliche zeitliche Beziehungen zwischen zwei oder mehreren Intervallen in Abbildung 3.2 präsentiert werden.

Eine weitere elementare Beziehung zwischen aufeinander folgenden Ereignissen ist die Kausalität. Diese beschreibt die Relation zwischen Ursache und Wirkung der beobachteten Zu-

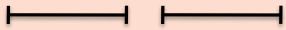
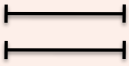
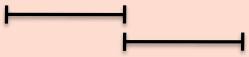
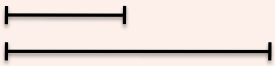
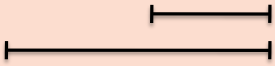
Relation	Symbol	Inverses Symbol	Piktografische Darstellung
X before Y	<	>	
X equal Y	=	=	
X meets Y	m	mi	
X overlaps Y	o	oi	
X during Y	d	di	
X starts Y	s	si	
X finishes Y	f	fi	

Abbildung 3.2: Überblick der Basisrelationen der „interval-based temporal logic“

stände. In diesem Zusammenhang soll die Aktivität „Kaffee kochen“ im Hinblick auf ihre kausalen Beziehungen untersucht werden. Die Ursache liegt in diesem Fall darin, dass der Bewohner die Kaffeemaschine präpariert und einschaltet, während die Wirkung beinhaltet, dass Kaffee aus der Maschine läuft. Dabei stellt die Trennbarkeit von Ursache und Wirkung einen signifikanten Aspekt dar. Außerdem ist es denkbar, dass eine beobachtete Wirkung mehrere Ursachen besitzt und nicht jede als solche erkannt wird oder eine inkorrekte Ursache einer Wirkung zugeschrieben wird. Begriffe die sich mit dieser fehlerhaften Zuordnung befassen lauten u.a. *false positive* und *false negative*. Ein Beispiel zur Erläuterung dieser Begriffe soll an dieser Stelle so gewählt sein, dass ein Klassifikator verschiedene Tage in „sonnig“ (positiv) und „nicht sonnig“ (negativ) einteilen soll. Bei einer *false positive* Klassifikation würde ein „sonniger“ Tag fälschlicherweise einem „nicht sonnigen“ Tag zugeordnet werden. Im Gegensatz hierzu würde eine *false negative* Einteilung einen „nicht sonnigen“ Tag als „sonnigen“ Tag einstufen. Diese Schwierigkeiten bei der korrekten Zuordnung vom Stattfinden eines Ereignisses und dessen Wahrnehmung begründen, dass jede kausale Beziehung exakt untersucht und verifiziert werden sollte.
In der Kausalität spielt die Zeit eine entscheidende Rolle, da die Ereignisse zum einen in eine zeitliche Reihenfolge gebracht werden und zum anderen eine Kette von Ursache und

Wirkung erst im Nachhinein bestimmt werden kann. Es ist somit vorstellbar, dass kausale Abhängigkeiten im ersten Schritt als Zufälle erkannt werden, wobei sich im Nachhinein die Beziehungen untereinander herauskristallisieren. Darüber hinaus ist es möglich, dass keine Ursache zu einer Wirkung beobachtbar ist und somit kein Auslöser für eine Kausalkette identifiziert werden kann.
Während der Kommunikation in Sensornetzwerken stellt die Kausalität eine besondere Herausforderung dar, weil sich Nachrichten im Netzwerk überholen können und daraufhin in eine geordnete Reihenfolge zu bringen sind. Hierzu werden in der Regel logische Uhren eingesetzt, mit Hilfe derer versendete Daten einen Zeitstempel erhalten, um die kausale Ordnung herzustellen. Bekannte Verfahren, eine solche Uhr zu realisieren, sind die Lamport-Uhr (Lamport (1978)) und Vektoruhren (Mattern (1989) und Fidge (1988)).

Die in Allen (1983) vorgestellten Relationen sowie die kausalen Beziehungen von Ereignissen sollen im weiteren Verlauf dieser Arbeit dazu verwendet werden, die für die Szenarien identifizierten Aktivitäten und ihre inbegriffenen Aktionen, in die erforderliche zeitliche Reihenfolge zu bringen. Mit dieser Aufgabe befasst sich der Abschnitt 5.4.

3.3 Vergleichbare Arbeiten

Der Schwerpunkt dieser Arbeit liegt in der Kontextinterpretation in einer *Smart Home* Umgebung. Da in diesem Bereich eine Vielzahl von Forschungsprojekten existiert, soll dieser Abschnitt dazu dienen, die Konzepte zur Kontextmodellierung einiger Projekte mit ähnlicher Zielsetzung vorzustellen.

SOCAM

Die „Service-Oriented Context-Aware Middleware“ (SOCAM) wurde mit dem Ziel entwickelt, die Erfassung von Kontexten durch den Einsatz verschiedener „Context Provider“ zu unterstützen, Kontexte zu interpretieren und die Kommunikation von Kontextinformationen als Push- und Pull-Mechanismus anzubieten, vgl. Gu u. a. (2005). In der Vorstellung dieser Middleware soll der Fokus auf die hier verwendete Kontextrepräsentation und die Verfahren zur Schlussfolgerung aus den Informationen gelegt werden.
Zur Speicherung möglicher Kontexte werden hier in der „Web Ontology Language“ (OWL)[6] geschriebene Ontologien verwendet. Die Kontexte selbst werden dabei in Prädikaten erster Ordnung, in der Form *Predicate(subject, value)*, repräsentiert.
In der SOCAM-Architektur werden Kontexte in die zwei Hauptkategorien „Direkter Kontext“ und „Indirekter Kontext“ eingeteilt. Direkte Kontexte können ohne weitere Verarbeitung von

[6] http://www.w3.org/TR/owl-semantics/

den „Context Providern“ bezogen werden. Hierunter sind beispielsweise die Raumtemperatur, die Uhrzeit oder von außen bezogene Wetterbedingungen zu verstehen. In einem weiteren Schritt lassen sich direkte Kontexte in die Kategorien „Erfasste Kontexte“ und „Definierte Kontexte“ einstufen. Erstere geben beispielsweise Auskunft darüber, wie der Status einer Tür ist, indem ein entsprechender Sensor abgefragt wird, während definierte Kontexte typischerweise durch einen Benutzer eingegeben werden. Beispiele für Kontexte dieser Art sind das Festlegen von Präferenzen beim Essen und bei der Musik eines Bewohners.
Indirekte Kontexte werden durch eine Analyse und eine Interpretation aus den direkten Kontexten mit dem Ziel abgeleitet, Schlussfolgerungen aus ihnen zu ziehen.

Die Weiterverarbeitung von Low-Level Sensorwerten bis hin zu High-Level Kontextinformationen wird in dieser Architektur mit einem „Context Interpreter“ durchgeführt. Zu diesem Zweck enthält diese Komponente einen „Context Reasoner“ (CR) und eine „Knowledge Database“ (KB). Der CR ist dafür zuständig, die von ihm abgeleiteten Kontexte zu kommunizieren sowie Konflikte und Inkonsistenzen innerhalb der KB aufzudecken. Die KB selbst bietet hingegen Mechanismen an, über welche die Komponenten das aktuelle Wissen über Kontexte erfragen, hinzufügen, löschen oder modifizieren können und beinhaltet die Ontologie.
Mit der Absicht Schlussfolgerungen auf der Basis der vorliegenden Daten zu ziehen, arbeitet der „Context Interpreter“ mit ontologiebasiertem und benutzerdefinierten regelbasiertem Ableiten. Ersteres wird u.a. dazu verwendet, um transitive Eigenschaften zwischen Regeln zu identifizieren und in die Schlussfolgerung einfließen zu lassen. Dazu gehört beispielsweise das Schließen darauf, dass sich der Bewohner in seiner Wohnung befinden muss, falls eine Detektion im Bad stattfindet. Das benutzerdefinierte regelbasierte Reasoning versucht über die Verkettung verschiedener Bedingungen, Kontexte zu erkennen. Für die Schlussfolgerungen werden hier neben den aktuellen Informationen auch vergangene Kontexte, die in einer „Context Database“ gespeichert werden, einbezogen.

Ähnliche Architekturen werden in „A Middleware for Context-Aware Agents in Ubiquitous Computing Environments“ (Ranganathan und Campbell (2003)) und „An Ontology for Context-Aware Pervasive Computing Environments“ (Chen u. a. (2003)) vorgestellt.

Bewertung

Die „Service-Oriented Context-Aware Middleware“ bietet aufgrund der Ontologie die Möglichkeit neue Kontexte in das System zu integrieren, eine allgemeingültige Begriffssammlung einzuführen und transitive Relationen zwischen Regeln aufzudecken. Durch den Einsatz einer Rule-Based Engine lassen sich Kontexte erkennen und benutzerdefiniert modifizieren, falls das Verhalten des Systems verändert werden soll.

Der Einsatz einer Ontologie erfordert allerdings, wie bereits in 3.2.4 beschrieben, einen aufwändigen Entwicklungsprozess, in dem geeignete Begrifflichkeiten und Regeln zwischen diesen definiert werden. Des Weiteren ergibt sich in dieser Architektur ein „Single Point of Failure“, was beinhaltet, dass bei einem Ausfall des „Context Interpreters“ keine Kontextinformationen verarbeitet werden können.

Hydrogen

Das Hydrogen Context-Framework wurde in Hofer u. a. (2003) beschrieben und stellt eine Drei-Schichten-Architektur vor. Für die Handhabung der Kontexte werden diese in „physikalische“ und „logische“ Informationen unterteilt. Physikalische Kontexte befinden sich auf einem sehr niedrigen Abstraktionslevel und tragen kontinuierlich dazu bei, den Zustand der Umgebung zu beschreiben. Logische Kontextinformationen werden hingegen mit dem Ziel verwendet, die vorliegenden physikalischen Kontexte mit semantischem Wissen anzureichern. Beispielsweise könnte aus den einzelnen Koordinaten über den Standort einer Person, unter Verwendung logischer Informationen, der Raum in dem er sich befindet gefolgert werden.
Die Zuständigkeit der drei Schichten wird in diesem Framework in die Interaktion mit den physikalischen Sensoren (Adaptor Layer), die Speicherung und Veröffentlichung von Kontexten (Management Layer) und der Arbeitsweise der einzelnen Applikationen (Application Layer), welche die Kontexte nutzen, klassifiziert. Eine genauere Betrachtung dieser Schichten findet im Folgenden statt.

1. **Adopter Layer:** Diese Schicht ist verantwortlich für die Erfassung der physikalischen Kontexte und reichert diese gegebenenfalls mit logischen Kontextdaten an. Außerdem steuert sie den synchronisierten Zugriff auf gemeinsam genutzte Ressourcen. In einem weiteren Schritt werden die Informationen an die Management Schicht weitergeleitet.
2. **Management Layer:** Auf dieser Schicht arbeitet ein „Context Server“, der alle aktuell vorliegenden Informationen über die Umgebung speichert und die Möglichkeit besitzt, diese mit anderen Komponenten über eine Peer-to-Peer Kommunikation zu teilen. Dieser Server bietet hierzu zum einen Mechanismus für die Abfrage von Daten an und besitzt zum anderen die Fähigkeit, weitere Komponenten bei Veränderungen des Bezugsrahmens zu benachrichtigen.
3. **Application Layer:** Zu dieser Schicht gehören diejenigen Anwendungen, welche die veröffentlichten Informationen der unterliegenden Schichten verwenden.

Im Gegensatz zur SOCAM wird hier keine Ontologie und keine Rule-Base Engine für das Reasoning über Daten eingesetzt. Dieses wird in diesem Framework im „Adopter Layer“ über die Anreicherung von Kontextobjekten mit semantischem Wissen realisiert. Außerdem

bleibt im Hydrogen-Framework die Persistierung von Kontextdaten aus, da zu jeder Zeit lediglich die aktuell vorliegenden Informationen im „Context Server“ verwendet werden.

Eine ähnliche zentrale Architektur zur Kontextinterpretation wird in „CASS - Middleware for Mobile Context-Aware Applications“ (Fahy und Clarke (2004)) vorgestellt.

Bewertung

Dieses Framework bietet durch seine offene Struktur die Gelegenheit, neue Kontextobjekte zu entwickeln und in die Architektur zu integrieren. Außerdem ist es in Hydrogen nicht erforderlich, eine spezielle Persistenzschicht zu pflegen, da hier keine Historie auftretender Daten angelegt wird und nur die aktuell vorliegenden Daten relevant sind. Durch das Fehlen vergangener Informationen ergibt sich jedoch der Nachteil, dass Anwendungen, die mit Werteveränderungen über die Zeit arbeiten, die vorangegangenen Daten eigenständig speichern müssen. Außerdem können keine Lernverfahren auf bisherige Kontextinformationen angewandt werden, welche beispielsweise beim Erkennen von Verhaltensmustern essentiell sind.
Aufgrund der zentralen Architektur ergibt sich wie bereits bei SOCAM ein „Single Point of Failure“. Sollte der „Context Server“ ausfallen unterbindet dieses eine Veröffentlichung und Abfrage von Kontextinformationen durch die Anwendungen.

Verteiltes Context-Reasoning

Die bisher vorgestellten Architekturen haben einen zentralen Ansatz zur Kontextinterpretation gewählt. Die damit einhergehenden negativen Aspekte, wie die in vielen Fällen suboptimale Ausnutzung von Netzwerkressourcen und die häufig geringe Performance des Systems beim Auftreten sehr großer Datenmengen, wurden in Rizou u. a. (2010) aufgegriffen und versucht mittels einer verteilten Kontextinterpretation aufzulösen.
Die zu identifizierenden Situationen erhalten in diesem Ansatz Erkennungsmuster, welche die herrschenden Bedingungen beschreiben. Diese „Situation Templates“ sind als Graphen zu verstehen, deren Knoten zum einen die Kontexte und zum anderen Operatoren darstellen. Unter Operatoren versteht man hier Verarbeitungsentitäten, die Sensorwerte mit Wissen anreichern, mit der Absicht High-Level Kontexte zu generieren.
Die Architektur dieses Systems gliedert sich in drei Schichten. Die erste Schicht, die „World Model“-Schicht, veröffentlicht die Kontextdaten an die „Context Reasoning“-Schicht, in der die beobachteten Daten zu höherwertigen Kontextinformationen verarbeitet werden. Auf der „Application“-Schicht arbeiten diejenigen Anwendungen, welche die Informationen in ihr Verhalten einbeziehen. Während diese Schichten-Architektur sehr derjenigen des Hydrogen-

Frameworks ähnelt, wird in dieser Architektur ein anderer Mechanismus zur Kontextinterpretation gewählt.
Die Interpretation des Kontextes teilt sich mit der „Initialization Phase“ und der „Execution Phase“ in zwei Abschnitte auf. Erstere generiert einen Ausführungsplan, in dem beschrieben wird, welcher Operator welche Aufgabe zur Kontexterstellung übernimmt. Für die Erstellung des Plans erhält die entsprechende Komponente eine Spezifikation der zu erkennenden Situation vom Anwender. Daraufhin ermittelt sie das geeignete „Situation Template“ und die erforderlichen Kontextquellen, Applikationen und Operatoren. In einem weiteren Schritt wird nun jedem Operator zugeteilt, welche Quellen bezogen und welche Applikationen von ihm bedient werden sollen.
Nachdem der Ausführungsplan an die Komponenten verteilt wurde, beginnt die „Execution Phase“. In diesem Abschnitt führen die Operatoren ihre zugewiesene Reasoning-Aufgabe aus, indem sie aktuelle Werte aus dem „World Model“ interpretieren und die „Application“-Schicht bei Veränderungen der Umgebung benachrichtigen. Die erzielten Ergebnisse werden von den Operatoren ebenfalls an das „World Model“ zurückgeschrieben, da die gegenwärtige Situation einen Teil des Modells beschreibt und zudem eine Persistierung vergangener Kontexte ermöglicht.
Für die Realisierung des verteilten Schlussfolgerns wird in dieser Architektur der verteilte Message-Passing Algorithmus von Pearl (Pearl (1988)) verwendet, der eine Verbreitung von Annahmen über die Umwelt in Bayes'schen Netzen ermöglicht. Außerdem wird hier ein Algorithmus zur Korrektur der Wahrscheinlichkeitstabellen eingesetzt, damit das Feedback des Benutzers verwertet werden kann.

Mit Architekturen für ein verteiltes Context-Reasoning befasst sich außerdem „Peer-to-Peer Context Reasoning in Pervasive Computing Environments“ (Gu u. a. (2008)).

Bewertung

Der hier vorgestellte Ansatz zum verteilten Context-Reasoning zeichnet sich durch eine effiziente Ausnutzung der Netzwerkressourcen aus, da keine zentrale Einheit erforderlich ist, welche die gesamten Daten zusammenführt. Hiermit einhergehend ist ebenfalls, dass sich der „Single Point of Failure“ aus der SOCAM sowie dem Hydrogen-Framework zu einem großen Teil auflöst und beim Ausfall einer Teilkomponente die weiteren Teilnehmer weiterhin miteinander kommunizieren und die Kontexterstellung vorantreiben können.
Jedoch wird zur Planerstellung in der „Initialization Phase“ in dieser Architektur ebenso eine zentrale Entität benötigt, da diese jedem Operator seine Reasoning-Aufgabe mitteilen muss. Dies bedeutet, dass die Auflösung des „Single Point of Failure“ lediglich für die Ausführungsphase gilt. Des Weiteren wird in diesem Ansatz davon ausgegangen, dass der Anwender die zu erkennende Situation festlegt. Für die automatische Aktivitätserkennung beinhaltet dieser

Aspekt, dass bereits eine Vorverarbeitung diverser Daten stattfinden muss, damit dem System mitgeteilt werden kann, welche Situation tendenziell eintritt und ein entsprechender Plan generiert wird.
Ein Vorteil, der sich bei einem zentralen Ansatz gegenüber dem verteilten Context-Reasoning ergibt, liegt darin, dass ein zentraler Kontextinterpreter Inkonsistenzen zwischen Informationen auflösen kann, bevor die Daten an die Applikationen gelangen. Bei der dezentralen Vorgehensweise müssen widersprüchliche Daten innerhalb der Anwendung erkannt und behandelt werden, was eine weitere Herausforderung für den Entwickler bedeutet.

3.4 Fazit

In diesem Kapitel wurden drei Morgenszenarien einer Bewohnerin in einem *Smart Home* vorgestellt, um diverse Aktivitäten zu identifizieren, die von einer intelligenten Wohnumgebung erkannt werden sollten. Diese sollen dem Menschen eine möglichst exakte Hilfestellung beim Erledigen von Aufgaben unter Einbeziehung des Kontextes gewährleisten. Im Hinblick auf die Aktivitätserkennung wurden daraufhin im ersten Schritt die Herausforderungen bei der Entwicklung kontextsensitiver Applikationen und Anforderungen an eine Architektur für Anwendungen dieser Art definiert. Dabei wurde erkennbar, dass der Wissensaustausch verschiedener „Context Provider“ und die Kontextinterpretation eine essentielle Rolle bei der Erstellung von *Context-Aware Systemen* spielen. Vor diesem Hintergrund wurde der Vorgang zur Interpretation des Bezugsrahmens genauer betrachtet und aufgezeigt, dass das hier vorgestellte System in der Lage sein muss, mit unsicherem Wissen umzugehen und eine Schnittstelle zu allen im Netzwerk vertretenden Teilnehmern bereitzustellen. Darüber hinaus konnten mit der modellunabhängigen, der modellbasierten und der ontologiebasierten Interpretation drei verschiedene Ansätze ermittelt werden, die im Hinblick auf eine Kontexterstellung und einer darauf folgenden Aktivitätserkennung miteinander zu kombinieren sind.

Aufgrund der Erkenntnisse aus diesem Kapitel soll der Fokus dieser Arbeit im weiteren Verlauf auf die Erstellung einer grundlegenden Architektur für *Smart Homes* und die modellunabhängige Kontextinterpretation gelegt werden. Die modellbasierte sowie ontologiebasierte Interpretation wird in verwandten Arbeiten an der HAW Hamburg untersucht. Aus diesem Grund findet hier zwar weiterhin eine Betrachtung im Hinblick auf die Kombination der Ansätze statt, allerdings sei an dieser Stelle auf Karstaedt (2012) für die modellabhängige Interpretation und Ellenberg (2011) für die ontologiebasierte Interpretation verwiesen, in denen diese Vorgehensweisen detaillierter beschrieben werden.
Bevor im Kapitel 5 das *Smart Home*, in dem diese Arbeit stattfindet, und ein Entwurf des zu realisierenden Systems vorgestellt wird, sollen im Folgenden einige grundlegende Verfahren und Herausforderungen, die speziell bei der modellunabhängigen Interpretation auf-

treten, beschrieben werden. Die Kontextinterpretation ohne vorliegendes Modell eignet sich insbesondere zur Verarbeitung von rohen Sensordaten, weshalb die Mechanismen zur Wissensgenerierung mit Rauschen behaftete Informationen umgehen können müssen. Dieses liegt darin begründet, dass bei dem Einsatz vieler Sensoren häufig inkorrekte Daten generiert werden, da die Informationsquellen in der Regel nicht fehlerfrei arbeiten. Nachdem ein Datensatz von möglichen Messfehlern bereinigt wurde, lassen sich verschiedene Methoden mit dem Ziel anwenden, Wissen aus den Daten zu ziehen. Eine bekannte und bewährte Vorgehensweise ist es, Cluster aus den Informationen zu bilden. Deshalb wird sich das folgende Kapitel 4 mit der Behandlung von Messfehlern beschäftigen und einen Einstieg in die Clusteranalyse liefern.

4 Algorithmische Verfahren

In Smart Home Umgebungen werden mittels diverser Sensoren sehr hohe Anzahlen an Daten über die Zeit generiert, wie beispielhaft in der Abbildung 4.1 demonstriert wird.

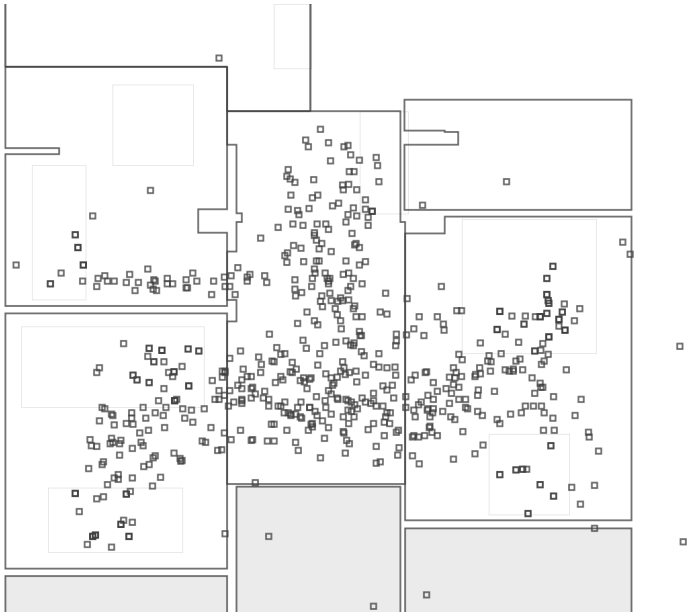


Abbildung 4.1: Beispielhafte Übersicht von Positionsdaten in einem Smart Home

Bevor diese als Kontextinformation in das System einfließen, sollte eine Bewertung und ggf. eine Vorverarbeitung der Daten stattfinden, damit sichergestellt wird, dass lediglich relevante Informationen das System beeinflussen. Zur Erfüllung dieser Anforderung ist es u.a. notwendig „Ausreißer“ während der Erhebung zu identifizieren. Hierunter werden Beobachtungen von Daten verstanden, die so weit von anderen Beobachtungen abweichen, dass sie den Verdacht erwecken, mittels eines anderen Mechanismus generiert worden zu sein, vgl. Hawkins (1980). Diese Abweichungen können innerhalb eines Datensatzes in sehr variablen Verteilungen auftreten und sind entweder auf Messfehler oder stark schwankende Wertegruppen zurückzuführen. Sollte es sich bei Ausreißern um unbemerkte Messfehler handeln, können diese fehlerhafte Reaktionen des Systems hervorrufen, weshalb ihre Auswirkungen möglichst zu minimieren sind. Mit der Detektion und einem möglichen Umgang mit diesen

Daten befasst sich der Abschnitt 4.1.
Des Weiteren können gespeicherte Sensordaten mit dem Ziel verwendet werden, neues Wissen über die Umgebung zu erlangen. Beispielsweise können sie dazu beitragen ein Verhaltensprofil des Bewohners einer Wohnung zu erstellen, mit der Absicht möglichst frühzeitig eine systemseitige Reaktion auf einen sich ergebenden Kontext zu initiieren. Eine Methode Strukturen in Datenbeständen zu erkennen ist die Clusteranalyse, welche im Teil 4.2 genauer diskutiert wird.
Außerdem wird in diesem Kapitel betrachtet, wie verschiedenartige Sensordaten miteinander kombiniert werden können, sodass sie auf einem höheren Interpretationslevel als eine einzelne Information zur Verfügung stehen. Die Möglichkeiten einer solchen „Sensorfusion“ werden in Abschnitt 4.3 vorgestellt.

4.1 Ausreißer

Da keine allgemeingültige mathematische Definition zu Ausreißern existiert, müssen Extremwerte häufig auf der Basis subjektiver Eindrücke klassifiziert werden, wodurch sich eine besondere Herausforderung bei der Identifizierung dieser Daten ergibt. Mit dem Ziel, dennoch möglichst fundierte Aussagen darüber zu treffen, welche Werte als Ausreißer zu betrachten sind, bietet das mathematische Fachgebiet der Statistik verschiedene Verfahren an. Eine Strategie liegt hier in der Einführung von Lagemaße und Streuungsmaße, mit der Absicht für eine Beobachtungsreihe eine Maßzahl zu bestimmen, anhand derer die Werte klassifiziert werden. Einige solcher Maße werden in Hartung u. a. (1999) erläutert und sollen im Folgenden vorgestellt werden.

$$\textbf{Arithmetisches Mittel: } \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.1}$$

$$\textbf{Median: } \bar{x}_{med} = \begin{cases} x_{((n+1)/2)} & \text{, falls n ungerade} \\ \frac{1}{2}\left(x_{(n/2)} + x_{((n+2)/2)}\right) & \text{, falls n gerade} \end{cases} \tag{4.2}$$

$$\textbf{Harmonisches Mittel: } \bar{x}_h = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}} \tag{4.3}$$

Für die Formel 4.2 ist zu beachten, dass die Beobachtungswerte in der Urliste $x_1, \ldots, x_n$ als geordnete Beobachtungsreihe $x_{(1)} \leq \ldots \leq x_{(n)}$ vorliegen müssen.

Diese Lagemaße lassen allerdings keine Rückschlüsse darüber zu, wie weit ein konkreter Messwert vom berechneten Zentrum einer Beobachtungsmenge entfernt ist. Mit dem Ziel, die Abweichung der Werte im Hinblick auf das oben berechnete Zentrum einer Datenmenge zu beschreiben, werden Streuungsmaße eingesetzt. Dieser Streubereich umfasst somit den Wertebereich in dem alle Beobachtungen liegen und wird durch den kleinsten sowie den größten gemessenen Wert bestimmt. In einem weiteren Schritt können Streuungsmaße dazu verwendet werden, eine durchschnittliche Abweichung der Daten hinsichtlich einer errechneten Lagemaße zu bestimmen. Eine verbreitete Vorgehensweise diese zu berechnen ist die mittlere absolute Abweichung vom Median (4.4).

$$d = \frac{1}{n} \sum_{i=1}^{n} |x_i - \bar{x}_{med}| \tag{4.4}$$

Eine weitere Herangehensweise zur Berechnung der durchschnittlichen Abweichung bietet die Varianz (4.5). Diese bezieht sich auf das arithmetische Mittel und liefert als Abstandsmaß keine betragliche Differenz, sondern einen quadratischen Abstand. Die Wurzel der Varianz ergibt daraufhin die Standardabweichung der Datenmenge und birgt den Vorteil, dass sie die gleiche Dimension der erhobenen Werte besitzt.

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \tag{4.5}$$

Unter Anwendung eines Lagemaßes und einem Streuungsmaß entsteht nun die Möglichkeit, Ausreißer innerhalb eines Datenbestandes zu identifizieren. Beispielsweise könnten alle Werte, die eine höhere Distanz als die mittlere absolute Abweichung vom Median oder als die Standardabweichung zum errechneten Zentrum der Beobachtungsreihe aufweisen, als Ausreißer deklariert werden.
Ein Verfahren, welches eine Form dieser Vorgehensweise verwendet um stark abweichende Werte zu detektieren, ist der „Ausreißertest nach Grubbs“ (Grubbs, 1969). Die Voraussetzung für die Durchführung dieses Tests ist das Vorliegen normalverteilter Werte. Sollte diese Bedingung erfüllt sein, wird geprüft, ob es sich beim kleinsten bzw. größten gemessenen Wert um einen Ausreißer handelt. Hierzu wird die Distanz der Werte zum arithmetischen Mittel berechnet und ins Verhältnis zur Standardabweichung gesetzt, um daraufhin mit dem Verhältnis in der Normalverteilung verglichen zu werden. Sollte der Wert dabei als Ausreißer identifiziert werden, wird er aus der Beobachtungsreihe gelöscht und mit dem Test von neuem begonnen.

Neben den Lage- und Streuungsmaßen lassen sich Extremwerte mittels einer Clusteranalyse ermitteln. Hier stellen diejenigen Werte Ausreißer dar, welche entweder keinem Cluster zugeordnet werden können, da sie eine zu hohe Distanz zu den übrigen Werten aufweisen oder einem Cluster mit zu geringer Gesamtgröße angehören.

Bewertung

Die hier vorgestellten Lagemaße werden mit dem Ziel verwendet, ein geeignetes Zentrum innerhalb eines Datenbestandes zu generieren. Im Hinblick auf ihre jeweilige Berechnung und die anschließende Verwendung für die Detektion von Ausreißern ist es bedeutend, ihre unterschiedlichen Eigenschaften zu betrachten.
Während der Median in seiner Berechnung nicht von einzelnen Ausreißern beeinflusst wird, bezieht das arithmetische Mittel Extremwerte vollständig mit ein. Der Median liefert somit keine Anhaltspunkte darüber, ob innerhalb der Beobachtungsreihe Abweichungen existieren. Diese werden jedoch nachträglich in die Berechnung der mittleren absoluten Abweichung vom Median einbezogen. Daher lässt sich feststellen, dass ein umso größerer Extremwert das Abstandsmaß vom Zentrum vergrößert und die Chancen für andere Werte erhöht, nicht als Ausreißer identifiziert zu werden. Aus diesem Grund ist es sinnvoll, wie der „Ausreißertest nach Grubbs“ vorgibt, entdeckte Abweichungen zu entfernen und einen erneuten Test mit den übrig geblieben Daten durchzuführen.
Bei der Ausreißer Ermittlung mit Hilfe einer Clusteranalyse (4.2) ist zu beachten, dass die Algorithmen darauf spezialisiert sind, die Werte in Clustern zusammenzufassen. Dies bedeutet, dass ähnliche Extremwerte zu einem Cluster verbunden und nicht als Ausreißer erkannt werden könnten.

Behandlung von Ausreißern

Wenn in einer Beobachtungsreihe Ausreißer detektiert werden, muss definiert sein, wie diese Werte zu behandeln sind. Eine Möglichkeit für die Glättung des Datensatzes besteht darin, diese Daten auszuschließen und zu eliminieren. Bedeutet das Entfernen dieser Daten allerdings einen schweren Datenverlust, sollten die Daten erhalten bleiben und in weitere Analysen einbezogen werden. Eine weitere Option ist die Kombination von Löschung und Erhaltung, dadurch dass die Ausreißer als fehlerhafte Werte gekennzeichnet werden. Somit bleibt zum einen die Größe des original Datensatzes erhalten, zum anderen werden folgende Analyseergebnisse nicht durch die Werte der Ausreißer verzerrt. Ein weiterer Vorteil der mit diesem Vorgehen einhergeht, ist die Möglichkeit Verfahren einzusetzen, mittels derer sich die gekennzeichneten Daten schätzen lassen. Dies lässt sich u.a. mit dem Mittelwertersatz durchführen, in welchem für den fehlenden Wert z.B. der Mittelwert oder der Median seiner Nachbarn verwendet wird. In diesem Fall muss jedoch zuvor definiert werden, wie viele

Nachbarwerte in die Berechnung einzubeziehen sind.
Eine weitere Eigenschaft des Mittelwertersatzes liegt darin, dass sämtliche einbezogene Werte, unabhängig von ihrer Distanz zum betrachteten Datum, mit gleicher Gewichtung in die Berechnung einfließen. In einigen Messreihen sollen hingegen die Nachbarn in der unmittelbaren Umgebung stärker gewichtet werden. In diesem Fall bietet sich ein Gauß-Filter an, der die Gewichtungen anhand einer Gauß-Kurve erstellt und daraufhin den neuen Wert berechnet.

Ein weiteres Verfahren für die Behandlung eines mit Rauschen behafteten Datensatzes ist der Einsatz des Kalman-Filters, welches erstmals 1960 von Rudolf E. Kalman in Kalman (1960) beschrieben wurde. Darauf folgend wurden verschiedene Erklärungen des Filters verfasst, wobei sich die hier folgende Beschreibung auf Welch und Bishop (1995) und Maybeck (1979) bezieht.
Das Kalman-Filter arbeitet mit allen verfügbaren Messwerten, um den aktuellen Systemzustand zu schätzen. Hierzu verwendet es u.a. Wissen über das System, die statistische Beschreibung über Systemrauschen und Messfehler sowie die initial verfügbaren Informationen über die Zustandsvariablen. Der Algorithmus dieses Filters ist rekursiv implementiert, was bedeutet, dass hier die vorangegangen Ergebnisse nicht im Speicher gehalten werden müssen. Die generelle Arbeitsweise teilt sich dabei in die abwechselnden Schritte „Predict“ und „Correct“ auf. Ersterer ist dafür verantwortlich eine a priori Schätzung des Systemzustandes und der Fehlerkovarianzmatrix zu berechnen. Im zweiten Schritt werden die Schätzungen unter Verwendung der Informationen des neuen Messwertes korrigiert. Man erhält somit einen a posteriori Zustand. Außerdem wird in diesem Schritt die a priori Fehlerkovarianz korrigiert. Eine Übersicht einer solchen Berechnungsoperation des Kalman Filters liefert Abbildung 4.2. Nach jeder Durchführung der beiden Schritte wird der Prozess wiederholt und die a posteriori Schätzungen dazu verwendet, die neuen a priori Schätzungen zu errechnen.

4.2 Clusteranalyse

Unter Clusteranalyseverfahren versteht man Algorithmen, mit Hilfe derer die Aufdeckung von Strukturen in Datensätzen ermöglicht wird. Das Ziel dieser Analyse ist es Cluster zu bilden, in denen die zugeordneten Objekte eine große Ähnlichkeit zueinander aufweisen, während zwischen den einzelnen Gruppen eine große Heterogenität herrscht.
Die Anwendung einer typischen Clusteranalyse besteht aus verschiedenen Schritten, in denen elementare Entscheidungen im Hinblick auf das Resultat gefällt werden müssen. Im Folgenden werden diese Aspekte kurz erläutert, vgl. Jain u. a. (1999).

1. **Repräsentation des Modells:** An dieser Stelle wird definiert, in welcher Form sich die zu untersuchenden Objekte beschreiben lassen und die Menge der einzubeziehenden

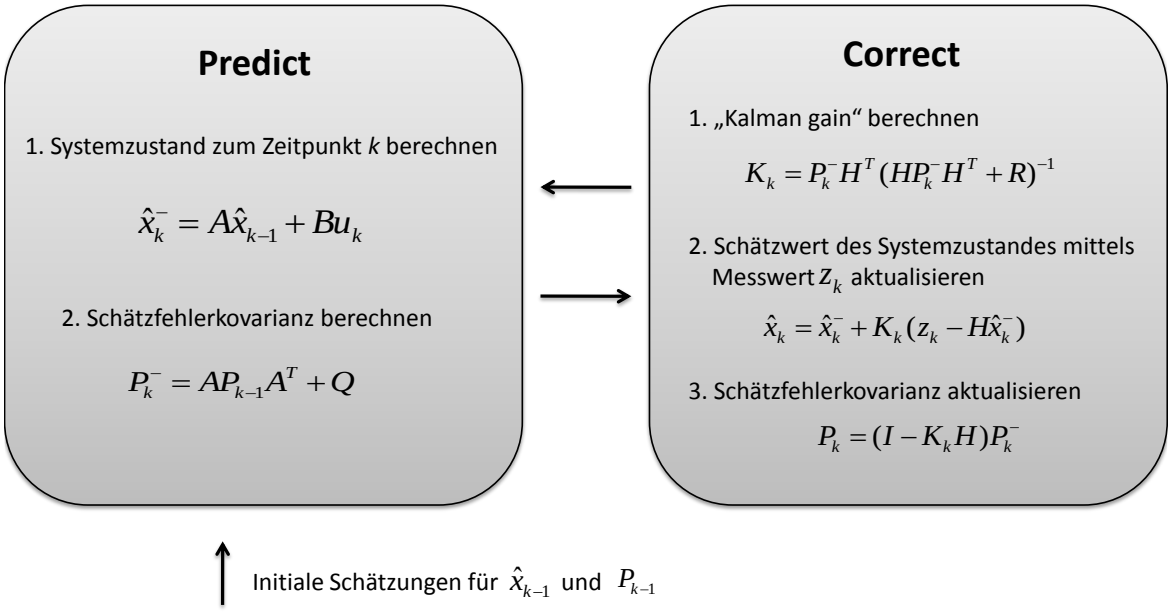


Abbildung 4.2: Übersicht einer Kalman-Filter Berechnungsoperation

Elemente definiert. Außerdem wird gegebenenfalls die Anzahl und Größe der Klassen bestimmt denen die Daten zugeordnet werden sollen. Zudem wird in diesem Schritt ein geeignetes Clusteringverfahren ausgewählt.

2. **Festlegung der Ähnlichkeits- bzw. Distanzmaße:** Hier werden die Ähnlichkeiten und Distanzen zwischen Objekten spezifiziert. Einige Formen dieser Maße werden im Abschnitt 4.2.1 genauer betrachtet.

3. **Clustering:** Im Anschluss an die Durchführung der ersten Schritte folgt die Ausführung der Clusteranalyse. Diese führt dazu, dass die betrachteten Objekte Clustern zugeordnet werden, bis eine entsprechende Abbruchbedingung erreicht wird. Einige dieser Bedingungen sowie elementare Unterschiede zwischen den Analyseverfahren werden in 4.2.2 vorgestellt.

4. **Datenabstraktion und Interpretation der Ergebnisse:** Auf das Clustering folgt die Interpretation des Resultats, die beispielsweise in der Untersuchung besteht, inwiefern sich die ermittelten Cluster voneinander unterscheiden. Zu diesem Zweck kann es hilfreich sein Daten zu abstrahieren, indem den Klassen kurze Beschreibungen hinzugefügt oder bei großen Clustern einige repräsentative Objekte wie Clusterzentren ausgewählt werden.

4.2.1 Proximitätsmaße

Wie bereits in 4.2 beschrieben wurde, muss für eine Clusteranalyse ein geeignetes Maß für die Ähnlichkeit (Ähnlichkeitsmaß) bzw. Unähnlichkeit (Distanzmaß) von Objekten gefunden werden, damit sie entsprechend dieser klassifiziert werden können. In Cha (2007) wird eine Übersicht dieser Maße geliefert, von denen einige nachfolgend vorgestellt werden.
Für die Distanzberechnung metrischer Daten ergeben sich u.a. die folgenden Berechnungsmöglichkeiten:

$$\textbf{Euklidische Distanz: } d_{Euc} = \sqrt{\sum_{i=1}^{d} |P_i - Q_i|^2} \tag{4.6}$$

$$\textbf{Manhattan Distanz: } d_{CB} = \sum_{i=1}^{d} |P_i - Q_i| \tag{4.7}$$

$$\textbf{Tschebyscheff Distanz: } d_{Cheb} = \max_i |P_i - Q_i| \tag{4.8}$$

Während die *euklidische Distanz* den Abstand liefert, der sich für den Fall ergibt, dass eine Gerade durch die beiden Punkte gelegt wird, gibt die *Manhattan Distanz* die Entfernung zwischen zwei Punkten an, sollte man sich entlang der Koordinatenachsen bewegen. Die Tschebyscheff-Distanz beschreibt hingegen die größte absolute Distanz eines Wertepaares.
Für nicht metrische Daten gehört der *Jaccard-Koeffizient* zu den bekanntesten Proximitätsmaßen. In diesem Fall wird die Ähnlichkeit von Objekten berechnet.

$$\textbf{Jaccard-Koeffizient: } J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{4.9}$$

Der *Jaccard-Koeffizient* berechnet sich aus der Anzahl der Elemente in beiden Mengen dividiert durch die Größe der Vereinigungsmenge.
Ähnlichkeitswerte sind in der Regel auf ein Intervall [0,1] normiert, wobei mit einem größeren Zahlenwert eine größere Ähnlichkeit einhergeht. Distanzmaße beschreiben dagegen die Unähnlichkeit von Objekten, was bedeutet, dass ein kleinerer Zahlenwert eine größere Ähnlichkeit kennzeichnet.

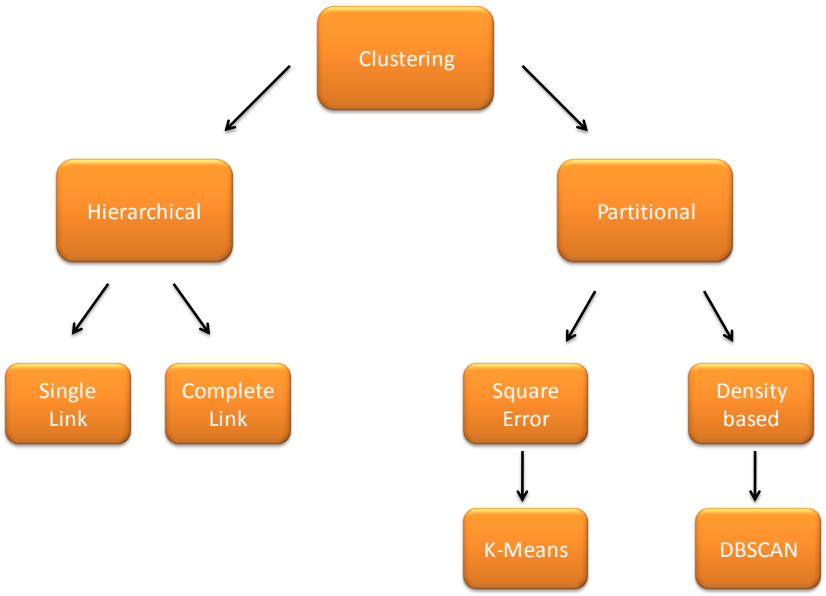


Abbildung 4.3: Übersicht der in dieser Arbeit erwähnten Clusteringverfahren

4.2.2 Clusteringverfahren

In der Literatur finden sich diverse Einteilungen von Clusteringverfahren. Die Aufteilung in der hier gezeigten Grafik 4.3 bezieht sich dabei ausschließlich auf die hier erwähnten Verfahren.

Clustering Algorithmen können generell entweder agglomerativ oder divisiv arbeiten. Bei agglomerativen Algorithmen stellt zunächst jedes Element ein einzelnes Cluster dar. Im weiteren Verlauf wird jedes Untersuchungsobjekt dem Nachbarn hinzugefügt, welches die größte Ähnlichkeit zu ihm aufweist. Die Cluster in diesem Verfahren wachsen somit stetig. Im Gegensatz hierzu beginnen divisive Algorithmen mit einem großen Cluster, welches alle zu untersuchenden Elemente enthält und sich in den folgenden Schritten in kleinere Gruppen teilt. Zu beachten gilt, dass beide dieser Ansätze ein entsprechendes Abbruchkriterium benötigen, welches beispielsweise eine gefundene Anzahl von Clustern entspricht.
In der Abbildung 4.3 findet in einem ersten Schritt eine Aufteilung der Algorithmen in hierarchische und partitionierende Ansätze statt, dessen Eigenschaften im folgenden Textabschnitt erläutert werden sollen.
Ersterer Ansatz findet seine Cluster, indem schrittweise zuvor gefundene Cluster miteinander verbunden werden. Es entsteht somit eine Struktur mit sich überschneidenden Gruppen in einem hierarchischen Baum. Zwei bedeutende hierarchisch agglomerative Clustering Varianten stellen zum einen das Single-Link und zum anderen das Complete-Link dar. Der Unterschied dieser Algorithmen liegt in der Beurteilung der Ähnlichkeit zwischen zwei Clus-

tern. Während im Single-Link die Distanz zweier Gruppen anhand der beiden am nächsten liegenden Einzelobjekte bestimmt wird, arbeitet das Complete-Link mit den beiden entferntesten Elementen aus beiden Clustern. Die Vereinigung von Clustern geschieht dabei jeweils anhand der minimalen Distanz zwischen den Gruppen. Der Single-Link Algorithmus tendiert zu Clustern mit einer langgezogenen Struktur, da die Gruppen hier durch wenige, kettenförmig verteilte Elemente verbunden werden. Im Gegensatz hierzu führt Complete-Link i.d.R. zu kompakten und fest gebundenen Clustern.

Partitionierende Algorithmen verwenden eine einzelne Partitionierung der Daten und ordnen die Elemente exakt einer Gruppe zu. Ein bedeutender Vertreter dieser Verfahren ist der k-Means Algorithmus (MacQueen (1967)), dessen Vorgehensweise im Folgenden genauer betrachtet wird.

1. Im ersten Schritt werden k Clusterzentren gewählt und zufällig im Datenraum verteilt.
2. An dieser Stelle folgt eine Zuordnung der Elemente in das Cluster, zu dessen Zentrum sie die minimale Distanz aufweisen.
3. Neuberechnung der Clusterzentren, indem diese so positioniert werden, dass die Summe der quadrierten Distanzen minimal ist.
4. Schritt zwei und drei werden so lange wiederholt, bis ein Konvergenzkriterium erreicht wird. Typische Kriterien sind das Erreichen einer minimalen Verkleinerung der aufsummierten Abstandsquadrate oder das Ausbleiben von Neuzuordnungen der Objekte in neue Cluster.

Ein Vorteil des k-Means Algorithmus liegt darin, dass er aufgrund seiner linearen Laufzeit selbst bei großen Datenmengen schnell Resultate hervorbringt. Ein Nachteil hingegen ist, dass die Anzahl der zu findenden Cluster bereits vor der Durchführung bekannt sein muss.

Eine weitere Form der partitionierenden Clusteranalyse findet sich in den dichtebasierten Algorithmen. Das bekannteste Verfahren innerhalb dieser Kategorie stellt das „Density-Based Spatial Clustering of Applications with Noise“ (DBSCAN) dar, welches im Jahr 1996 vorgestellt wurde (Ester u. a. (1996)). Die Grundidee hier liegt darin, Objekte anhand einer Dichteverbundenheit zu einem Cluster zu vereinen. Dieser Algorithmus liefert neben den gefundenen Clustern auch sog. Rauschpunkte, welche diejenigen Elemente darstellen, die keiner Gruppe zugeordnet werden konnten. Neben diesen Punkten unterscheidet der Algorithmus außerdem zwischen Kernobjekten und dichte-erreichbaren Objekten, welche mit Hilfe der Initialisierungsparameter ϵ und *minPts* gefunden werden. ϵ definiert dabei einen Bereich um ein Objekt, innerhalb dessen andere Objekte erreichbare Elemente sind. *MinPts* spezifiziert hingegen die minimale Anzahl an Elementen, die ein Cluster beinhalten muss. Ein Kernobjekt ist somit ein Objekt, welches mindestens *minPts* Elemente in seiner ϵ-Umgebung besitzt. Ein dichte-erreichbares Objekt ist demgegenüber kein Kernobjekt, sondern liegt in

der ϵ-Umgebung eines solchen. Ein beispielhafter Cluster aus einzelnen Positionsdaten (Abbildung 4.1) wird in der Grafik 4.4 dargestellt.
Die Vorteile des DBSCAN's liegen darin, dass die Anzahl der zu findenden Cluster nicht vor der Analyse definiert werden muss, sondern sich während der Durchführung des Algorithmus ergibt. Außerdem werden Ausreißer erkannt und separat als Rauschen innerhalb des Datensatzes deklariert. Bei sehr stark schwankenden Werten und ungünstig gewähltem ϵ und *minPts* kann sich allerdings der Nachteil ergeben, dass keine Cluster gefunden werden, da kein Objekt *minPts* Elemente in seiner ϵ-Umgebung aufweist.

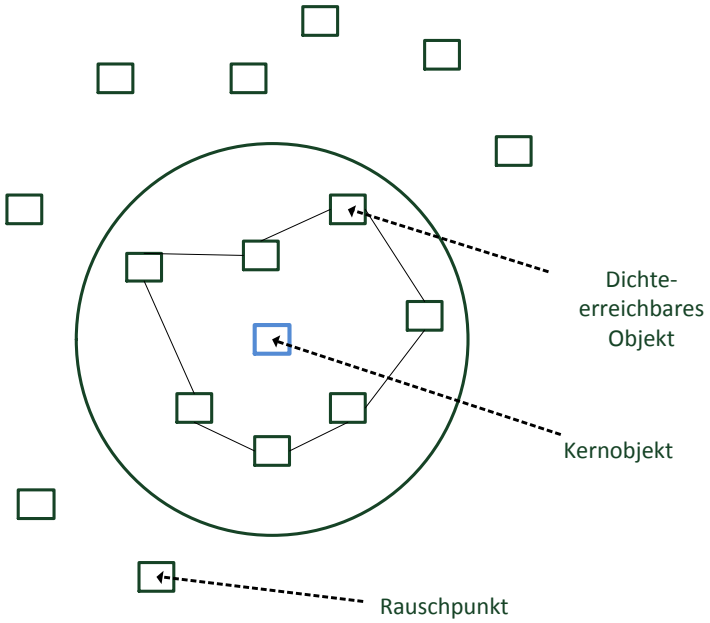


Abbildung 4.4: Ein ermitteltes DBSCAN-Cluster mit ϵ = 0,5 und MinPts = 7

4.3 Sensorfusion

Unter einer Sensorfusion versteht man die Verknüpfung verschiedener Sensordaten, mit der Absicht neues oder exakteres Wissen aus den Messwerten zu gewinnen. Je nach Anwendungsfall eignen sich dabei verschiedene Konzepte zur Integration von einzelnen Sensoren in ein Multisensorsystem. In diesem Zusammenhang werden in Ruser und León (2007) drei verschiedene Integrationskonzepte vorgestellt, die im Folgenden kurz erläutert werden.

- **Konkurrierende Integration:** Hier werden mehrere gleichartige Sensoren für eine identische Aufgabe eingesetzt, um Mehrdeutigkeiten und Unsicherheiten innerhalb der

Werte auszulöschen. Sollten beispielsweise von fünf Bewegungssensoren, vier eine registrierte Aktivität melden, während ein Sensor das Gegenteil berichtet, ist davon auszugehen, dass die These der Mehrzahl die korrekte darstellt. Bei einer Temperaturmessung mehrerer Informationsträger wäre hingegen eine Mittelung der erhobenen Werte denkbar.

- **Komplementäre Integration:** In diesem Fall werden ebenso gleichartige Sensoren, jedoch vor einem anderen Hintergrund, miteinander verknüpft. Eine solche Integration wird mit dem Ziel verwendet, den Detektionsbereich zu erweitern, indem Sensoren mit unterschiedlichen Messbereichen eingesetzt werden.
- **Kooperative Integration:** Dieser Ansatz verfolgt die Kombination von Beobachtungen unterschiedlicher Informationsquellen, mit der Absicht die Daten so zusammenzuführen, dass sie in einem weiteren Interpretationsschritt als Gesamtinformation verwendet werden können. Ein solches Vorgehen hat den Vorteil, dass die Komponenten, welche im Verlauf einer Kontexterstellung zum Einsatz kommen, keine Kenntnisse über das Zustandekommen der erlangten Informationen benötigen, sondern diese direkt in Ihre Berechnungen einbinden können.

Zur Durchführung einer Sensorfusion werden in Ruser und León (2007) drei Abstraktionsebenen vorgestellt, anhand derer eine Zusammenführung durchgeführt werden kann. Auf der ersten Ebene, der *Signalebene*, werden die Rohdaten der einzelnen Sensoren, unter der Voraussetzung dass diese vergleichbar sind, kombiniert. Sollte auf dieser Ebene kein zeitlicher oder räumlicher Zusammenhang zwischen den Sensorwerten herstellbar sein, bietet sich die *Merkmalsebene* zur Fusionierung an. Hier werden Signaldeskriptoren mit dem Ziel kombiniert, verbesserte Schätzwerte über die Eigenschaften der Sensoren zu erlangen. Auf der *Symbolebene* werden symbolische Signaldeskriptoren, wie beispielsweise Klassifikationsresultate zusammengeführt, um anhand jeweils bewerteter Wahrscheinlichkeiten Entscheidungen zu treffen.

Während einer Vereinigung von Sensorwerten ist es möglich, dass benötigte Daten zum aktuellen Zeitpunkt nicht vorliegen. In diesem Fall lassen sich die in 4.1 erläuterten Schätzverfahren einsetzen, mit dem Ziel, die erforderten Messwerte zu schätzen.

5 Design

In der Analyse wurden Anforderungen und Erwartungen an eine Applikation definiert, die im Folgenden in einen Systementwurf eingebettet werden sollen. Das verfolgte Ziel liegt darin, eine für einen intelligenten Wohnbereich geeignete Architektur zur Kontextinterpretation zu implementieren, auf deren Basis verschiedene Aktivitäten der Bewohner erkannt werden können. Im ersten Schritt soll jedoch das *Smart Home* vorgestellt werden, in dessen Rahmen diese Arbeit entstanden ist, um die Struktur, in die das zu implementierende System integriert wird, näher zu beleuchten.

5.1 Umgebung

Seit Beginn des Jahres 2009 wird auf dem Campus der Hochschule für Angewandte Wissenschaften Hamburg (HAW) an der Realisierung eines *Smart Home* gearbeitet. Hierbei handelt es sich um eine 130m^2 große, als Loft ausgeprägte und real bewohnbare Wohnung (siehe Abbildung 5.1), in der Applikationen vor dem Hintergrund entwickelt werden, eine nahtlose Interaktion zwischen Mensch und ubiquitärer Computertechnik in alltäglichen Situationen zu erzielen. Die Ziele und Grundsätze des *Living Place Hamburg* werden in einer offiziellen Kurzbeschreibung (von Luck u. a. (2010)) zusammengefasst und sollen nun kurz vorgestellt werden.

> *„Cornerstone of our installations will be the seamless interaction between ubiquitous computing elements, including the touching of tangible objects in order to achieve smart living situations. In times of digitalisation based on social software the concept of neighbourhood is currently in redefinition. Integration of social software into the Living Place Hamburg will be part of our efforts for the purpose of creating a new kind of community building.*
> *Ubiquitous computing let disappear the borderline between labour and leisure time. Therefore one part of the research in the Living Place Hamburg is the integration of labour in leisure time in a sense of extending the enterprise 2.0 metaphor into the home office metaphor.“*

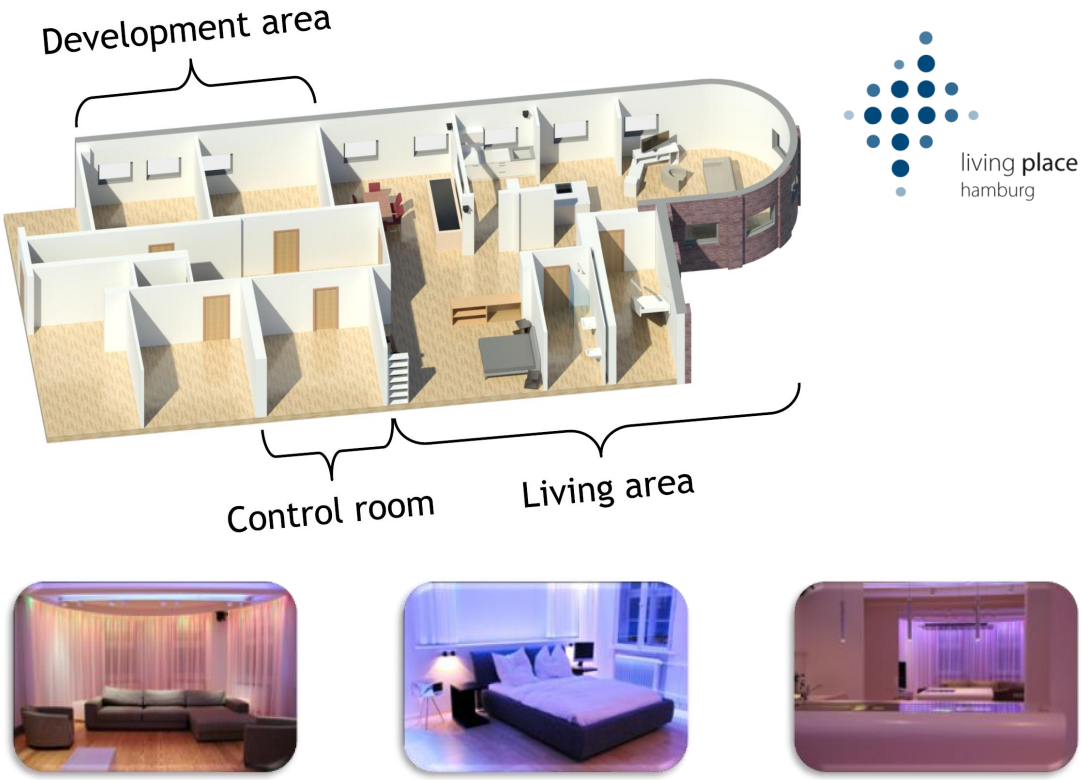


Abbildung 5.1: 3D-Modell des Living Place Hamburg (erstellt von Bastian Karstaedt) sowie reale Bilder der Wohnung

Ein wesentlicher Vorteil, der mit den Eigenschaften des *Living Place Hamburg* einhergeht, ist die Möglichkeit, Experimente unter realen Wohnbedingungen durchzuführen. Hierdurch steigt die Aussagekraft von Usability-Tests signifikant, da den Probanden die Systeme in einer natürlichen Umgebung präsentiert werden. Die ausgeführten Tests können über den an die Wohnung angeschlossenen Kontrollraum vollständig überwacht werden und die Reaktionen des Menschen mit Hilfe von verschiedenen Kameras, Mikrophonen und weiteren Monitoring-Verfahren direkt aufgezeichnet werden. Die entwickelten Interaktionskonzepte können somit in einem iterativen Prozess evaluiert und verbessert werden, da die Rückmeldungen unterschiedlicher Personen kontinuierlich in die Systeme einfließen.

Der *Living Place Hamburg* besitzt eine offene Struktur und dient einer Reihe von interdisziplinären Projekten als Entwicklungsgrundlage. Deren parallele Bearbeitung führt zu einer starken Lebendigkeit des Gesamtsystems, da stetig neue Anwendungen in die Architektur integriert und bereits bestehende Systeme erweitert oder ausgetauscht werden. Aus diesem Grund ist es unentbehrlich, dass in diesem Kapitel vorgestellte Design im Hinblick auf diese Eigenschaft zu entwickeln und die Zuständigkeiten der Teilsysteme in einzelne gekapselte

Module zu gliedern, um ein lose gekoppeltes System zu erhalten.
Ein leitendes Prinzip, welches sich für die Implementierung eines Systems mit verschiedenen isolierten Elementen eignet, stammt aus der Softwareentwicklung und lautet „separation of concerns“ (SoC). Es beschreibt den Prozess der Partitionierung von Computerprogrammen in verschiedene Teile, sodass sich die entstehenden Segmente möglichst wenig in ihrer Funktionalität überschneiden. In Hürsch u. a. (1995) wird diese Vorgehensweise in zwei verschiedene Level aufgeteilt.

1. **Konzeptuelles Level:** Dieses Level beinhaltet zwei elementare Aufgaben. Im ersten Schritt müssen klare Definitionen und konzeptuelle Identifikationen der einzelnen zu erledigenden Aufgaben im Hinblick auf ihre Unterscheidbarkeit zu anderen gefunden werden. Daraufhin muss sichergestellt werden, dass die individuellen Konzepte *primitiv* sind, was bedeutet, dass sie keine Kompositionen mehrerer Konzepte darstellen.
2. **Implementationslevel:** Hier muss untersucht werden, in welcher Form die Aufgaben voneinander isolierbar sind. Das Ziel dieses Levels liegt somit in der Trennung von Arbeitsschritten, die unterschiedliche Aufgabenteile bearbeiten, damit eine lose Kopplung der Komponenten untereinander erreicht wird.

In dem Artikel „The Art of Separation of Concerns“ von David Greer werden einige Vorteile dieses Ansatzes gegenüber einem monolithischen System vorgestellt (Greer (2008)). Diese liegen u.a. in der einfacheren Wartbarkeit des Systems, da in der SoC geringe Redundanzen und eine Singularität der einzelnen Komponenten angestrebt wird, wodurch das Gesamtsystem übersichtlicher gestaltet wird. Mit der guten Wartbarkeit wird außerdem eine Stabilität des Systems erreicht.
Die Eigenschaft dieser Strategie, dass jede Komponente lediglich einen einzelnen Satz von kohäsiven Aufgaben besitzt und geeignete Schnittstellen nach außen anbietet, resultiert häufig in einer guten Erweiterbarkeit des Systems. Außerdem hat die lose Kopplung unter den Elementen zur Folge, dass Bestandteile des Gesamtsystems in weiteren Projekten eingesetzt werden können, ohne neu implementiert oder angepasst werden zu müssen.

Aufgrund der positiven Effekte die sich durch die Herangehensweise des „separation of concerns“ ergeben, hat der folgende Abschnitt zum Ziel, im Hinblick auf diese Strategie die einzelnen Komponenten näher zu beschreiben und ihnen Aufgaben auf der Basis der Analyseergebnisse zuzuteilen.

5.2 Zentrale Designentscheidungen

Bevor ein Entwurf des Gesamtsystems vorgestellt wird, sollen zunächst einige elementare Designentscheidungen für die in Abbildung 3.1 identifizierten Komponenten diskutiert wer-

den. Diese Entscheidungen sollen den Entwicklungsprozess des Systems dahingehend beeinflussen, dass sowohl die interne Arbeitsweise der einzelnen Komponenten als auch die Kommunikation mit weiteren Teilnehmern fixiert wird.

5.2.1 Communication Base

Die Communication Base soll in dieser Architektur die Aufgabe der Kommunikationsschnittstelle übernehmen. Das bedeutet, dass jeder Teilnehmer des Netzwerkes mittels dieser Komponente die Möglichkeit besitzen muss, Services zu erfragen und Kontextinformationen zu verteilen. Aus diesem Grund werden nun einige grundlegende Kommunikationsformen vorgestellt und analysiert, die für das zu implementierende System geeignet sind.

Peer-to-Peer

In einem Peer-to-Peer (P2P) Netzwerk sind alle Teilnehmer gleichgestellt und können Dienste anderer Peers in Anspruch nehmen sowie selbst Services zur Verfügung stellen. Diese Eigenschaft der Netzwerke stellt den elementaren Unterschied zur Client-Server-Infrastruktur dar, in welcher der Server als einziger Dienstanbieter fungiert.
P2P-Netzwerke lassen sich in strukturierte und unstrukturierte Systeme kategorisieren, vgl. Steinmetz und Wehrle (2005). Letztere unterteilen sich dabei in drei Bereiche:

1. **Zentralisierte P2P-Systeme:** In diesen Netzwerken kommt ein zentraler Server zum Einsatz. Im Gegensatz zur Client-Server-Struktur werden hier nicht die gemeinsam verwendeten Ressourcen und Dienste auf dem Server bereitgestellt, sondern es wird ausschließlich die IP der Peers gespeichert, unter der entsprechende Daten zu finden sind. In dieser Sterntopologie verbindet sich jeder Peer mit dem Server, um ihm mitzuteilen, welche Informationen und Services über ihn bezogen werden können und zur Anfrage benötigter Daten. Bei einer solchen Anfrage antwortet der Server daraufhin in der Regel mit den IP's und Ports der entsprechenden Quellen, woraufhin sich ein Datenaustausch ohne weitere Belastung des Servers initiieren lässt.
2. **Reine P2P-Systeme:** Hier werden keine zentralen Einheiten verwendet. Wenn ein Peer an einem solchen Netzwerk teilnehmen möchte, muss er die IP von mindestens einem aktiven Teilnehmer besitzen. Diese kann er beispielsweise aus einem eigenen Host-Cache beziehen. Nachdem sie dem P2P-Netzwerk beigetreten sind, können Peers sich über Ping-Nachrichten im Netz bekannt machen. Als Antwort auf diese Nachricht erhalten die neuen Knoten weiterführende Informationen, wie z. B. die IP des Empfängers.

3. **Hybride P2P-Systeme:** Eines der Ziele dieser Systeme ist die Auflösung des hohen Nachrichtenaufkommens in den reinen P2P-Systemen. Zu diesem Zweck beinhalten diese Systeme eine Hierarchie in der zwischen Superpeers und einfachen Knoten unterschieden wird. Zur Informationsgewinnung kommunizieren die einfachen Knoten mit dem ihnen zugeteilten Superpeer, der daraufhin mit weiteren Superpeers interagiert, mit dem Ziel, den benötigten Service ausfindig zu machen. Nach einer erfolgreichen Mitteilung des Superpeers bauen zwei einfache Knoten eine Verbindung zueinander auf und können folgend ohne die Superpeers miteinander kommunizieren.

In unstrukturierten Peer-to-Peer Systemen verwalten die Peers keine Informationen über die angebotenen Ressourcen anderer Netzteilnehmer, was eine Weiterleitung von Anfragen unmöglich macht. Aus diesem Grund werden auf der einen Seite häufig Server eingesetzt, über welche Peers die Möglichkeit erhalten Komponenten aufzuspüren die erforderliche Daten besitzen. In diesem Ansatz ergibt sich durch die zentrale Einheit allerdings ein Flaschenhals, da die Peers von der Performance und der Erreichbarkeit des Servers abhängig sind. Auf der anderen Seite können die einzelnen Peers so lange Anfragen als Broadcast-Nachrichten in das Netzwerk senden, bis ein Peer mit der geforderten Information antwortet. Diese Herangehensweise erzeugt in der Regel ein unnötiges und sehr hohes Datenaufkommen innerhalb des Netzwerks und kann somit die Performance des gesamten Systems negativ beeinflussen.
Zur Auflösung dieser negativen Effekte werden strukturierte P2P-Systeme eingesetzt, die „Distributed Hash Tables“ (DHT) verwenden. Im Gegensatz zu den unstrukturierten Systemen ist in einer DHT jeder Knoten für eine bestimmte Menge von Daten zuständig. Dabei werden Knoten und Daten auf denselben Schlüsselbereich abgebildet, weshalb eine Route zu einem Knoten ebenso zu den Daten führt, für welche der jeweilige Teilnehmer verantwortlich ist. In einer DHT besitzt jede Entität einen eindeutigen Identifier, die ihm über eine Hashfunktion zugewiesen wird. Ein Beispiel für die Aufteilung einer DHT präsentiert die Abbildung 5.2. Damit alle Schlüssel innerhalb einer Tabelle erreichbar sind, pflegen die einzelnen Knoten Routing-Tabellen, in denen sie die Schlüssel eines Teils des Gesamtsystems abspeichern. Sollte ein Knoten eine Anfrage für einen nicht von ihnen verwalteten Schlüssel erhalten, leiten sie die Anfrage an einen Knoten aus ihrer Routing-Tabelle weiter. Zur Handhabung für den Beitritt, Austritt und Ausfall eines Knotens liefert die Datenstruktur ebenso entsprechende Mechanismen.
Bekannte „Distributed Hash Table“-Algortithmen sind „Chord“ (Stoica u. a. (2001)) und „Pastry“ (Rowstron und Druschel (2001)).

Reine, hybride und strukturierte Peer-to-Peer Netzwerke skalieren aufgrund ihrer beschriebenen Eigenschaften sehr gut, da es keine zentrale Einheit gibt, die den Informationsaustausch zwischen den Kommunikationsteilnehmern organisieren muss. Außerdem kann durch den Einsatz von strukturierten P2P-Netzen der „Single Point of Failure“ vollständig aufgelöst werden. Dieses ist in reinen und hybriden P2P-Netzen zwar ebenso möglich, jedoch nur un-

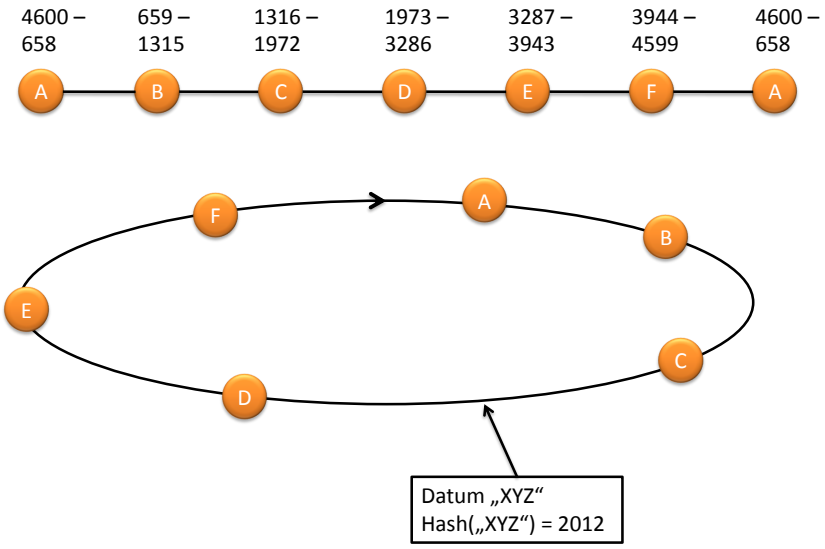


Abbildung 5.2: Beispiel einer „Distributed Hash Table“ mit sieben Peers, nachempfunden aus Steinmetz und Wehrle (2005)

ter den hohen Kosten einer Überflutung des Netzes oder einer Neuwahl eines Superpeers. In DHT's führt die Auflösung der zentralen potenziellen Fehlerquelle zu einer hohen Fehlertoleranz, da bei einem Ausfall die zu verwaltenden Daten unter den weiterhin verfügbaren Knoten verteilt werden. In reinen und hybriden Netzen sind bei dem Ausfall eines Knotens die Daten nur dann weiter verfügbar, falls sie ein weiterer Peer dupliziert hat.
Unter Betrachtung der positiven Effekte die sich bei einem Einsatz einer DHT ergeben, würde diese Datenstruktur eine ideale Kommunikationsbasis für den *Living Place Hamburg* darstellen. Allerdings stellt die Aktualität der kommunizierten Daten einen signifikanten Aspekt dar, was bedeutet, dass Clients, die sich für entsprechende Kontextdaten interessieren, kontinuierlich im Netzwerk nach neuen Informationen fragen müssten. Dieser Aspekt würde somit, wie bereits in reinen P2P-Netzen beschrieben, zu einem unnötigen Nachrichtenaufkommen führen. Darüber hinaus muss der Entwickler seine Anwendungen mit Wissen versehen, welches nicht ausschließlich zur Erfüllung der Aufgabe des Systems verwendet wird. Dieser Aspekt widerspricht folglich der Anforderung, dass Entwickler möglichst einfach neue Anwendungen in das System einfließen lassen können sollen. Ebenfalls stellt es eine Herausforderung dar, eine Fehlerbehandlung zu realisieren, da hier keine globale Sicht auf das Gesamtsystem vorliegt.

Pipes and Filters

Das Pipes-and-Filters Muster bietet einen Ansatz für den Umgang mit Datenströmen, vgl. Buschmann u. a. (1998). Hierzu teilt das System eine zu bewältigende Aufgabe in mehrere Verarbeitungsschritte, während sich jeder Schritt durch eine Datenverarbeitung mittels einer Filterkomponente auszeichnet. Die Folge der einzelnen Schritte wird dabei als Verarbeitungs-Pipeline bezeichnet. Innerhalb dieser Pipeline werden die Filter mit dem Ziel eingesetzt, die vorliegenden Daten zu ergänzen, dadurch dass Operationen auf den Daten ausgeführt oder eigene Informationen hinzugefügt werden. Die Ausführung dieser Komponente lässt sich durch drei Ansätze realisieren. Entweder können Filter ihre Aufgabe durchführen, sobald die vorhergehende Komponente neue Eingabedaten übergibt, das nachfolgende Element Informationen entnimmt oder indem sie als schleifenbasierte Instanz dauerhaft aktiv sind.
Für den Datenfluss zwischen Filtern werden Kanäle (engl. Pipes) implementiert, über welche das Versenden von Nachrichten und eine Synchronisation aktiver Pipes stattfindet. Des Weiteren existiert in diesem Architekturmuster eine Datensenke, an welche die Ergebnisse der Verarbeitungs-Pipeline zu übergeben sind. Hier wird zwischen einer aktiven, die Daten eigenständig aus dem Filter entnimmt und einer passiven Datensenke, in welche der Filter seine Informationen schreibt, unterschieden.

Das hier vorgestellte Muster weist eine offene Struktur auf, mit der sich einige Vorteile ergeben. Hierzu zählt u.a. die Austauschbarkeit von Filtern. Sollte ein Filter neu implementiert werden, kann dieser aufgrund der losen Kopplung innerhalb des Systems ohne weiteren Aufwand ersetzt werden, falls sie die notwendigen Schnittstellen zur Umwelt besitzen. Außerdem kann eine Verarbeitungs-Pipeline durch eine neue Anordnung von Filtern oder dem Hinzufügen weiterer Filter neu definiert werden. Mit der Absicht, diese positiven Effekte in die Kommunikationsbasis zur Kontextinterpretation zu übertragen, wird nun betrachtet, welche Aufgaben von den Komponenten aus diesem Architekturmuster übernommen werden können.
In der Architektur des *Living Place Hamburg* treten die Context-Provider als Filter auf und ergänzen den aktuellen Zustand um ihre Informationen. Der hierdurch neu entstandene Kontext wird daraufhin an die nachfolgende Filterkomponente gesendet und weiterhin bearbeitet. Sobald die Daten jedes Filters einbezogen wurden, wird das Resultat an die Datensenke, welche eine zentrale Einheit darstellt und in der Lage dazu ist, mit sämtlichen Aktoren zu kommunizieren, geschickt. Diese kann nun aufgrund der erhaltenen Informationen eine entsprechende Reaktion des Gesamtsystems einleiten.
Die Pipes-and-Filters Komponenten besitzen ebenso wie P2P-Systeme die Herausforderung der Fehlerbehandlung, da es keinen gemeinsamen globalen Zustand gibt. Sollte ein Filter seine Operation unerwartet abbrechen, existiert keine Instanz die diesen Abbruch bemerkt, was zu einem endlosen Warten der nachfolgenden Komponenten führen könnte. Selbst bei der Einführung einer maximalen Bearbeitungszeit von Informationen müsste die Kontexter-

stellung von neuem beginnen, falls ein Filter seine Aufgabe nicht beendet. Des Weiteren ergibt sich durch den Einsatz dieses Musters eine Herausforderung in der Positionierung der Filter innerhalb der Pipeline, da die einzelnen Teilnehmer ihre Informationen nur dann beisteuern können, wenn sie an der Reihe sind. Sollte eine Komponente neue Kontextinformationen erhalten, nachdem sie ihre Daten weitergegeben hat, bedeutet dieses, dass die Informationen nicht in die Reaktion des Systems einfließen können.

Blackboard Architektur

Das erste Blackboard-System war das HEARSAY-II-Spracherkennungssystem und wird in Erman u. a. (1980) beschrieben. Dieses Architekturmuster lässt sich sehr treffend mit folgender Metapher beschreiben:

> *„Imagine a group of human specialists seated next to a large blackboard. The specialists are working cooperatively to solve a problem, using the blackboard as the workplace for developing the solution.*
> *Problem solving begins when the problem and initial data are written onto the blackboard. The specialists watch the blackboard, looking for an opportunity to apply their expertise to the developing solution. When a specialist finds sufficient information to make a contribution, she records the contribution on the blackboard, hopefully enabling other specialists to apply their expertise. This process of adding contributions to the blackboard continues until the problem has been solved.“* – Corkill (1991)

Wenn man dieses Sinnbild im Hinblick auf die Kommunikationsbasis für Sensoren zur Kontextinterpretation betrachtet, lassen sich die Komponenten wie folgt übertragen: Die Sensoren sind die Spezialisten, welche auf das Blackboard, also die Kommunikationsbasis schauen. Diese versuchen das Problem der Kontexterstellung zu lösen, indem jeder Sensor seine Werte auf das Blackboard schreibt, sobald er davon ausgeht, einen Beitrag zur Lösungsfindung leisten zu können.
In Buschmann u. a. (1998) werden einige Begriffe und Komponenten definiert, die für die Entwicklung einer Blackboard-Architektur nützlich sind. Hier wird die Menge aller Elemente, die auf dem Blackboard erscheinen können, als *Vokabular* definiert. Diejenigen Objekte, die während eines Lösungsvorgangs auf dem Blackboard erscheinen, werden als *Hypothese* bezeichnet und können gegebenenfalls zu einem späteren Zeitpunkt wieder verworfen werden. Die *Wissensquellen* stellen unabhängige Subsysteme dar und liefern jeweils eine Teillösung an das Blackboard. Die Gesamtlösung des Problems ist dabei nur unter Einbeziehung mehrerer Wissensquellen möglich. Die Quellen kommunizieren in dieser Architektur

zu keiner Zeit auf direktem Wege miteinander, sondern schreiben ihre Informationen ausschließlich auf das Blackboard und lesen von diesem. Grundvoraussetzung für die Teilnahme an der Problemlösung ist somit, dass das Vokabular des Blackboards verstanden wird. Außerdem muss jede Komponente eigenverantwortlich bestimmen, unter welchen Bedingungen sie einen Teil zur Problemlösung beisteuern kann.
Des Weiteren besitzt das Blackboard eine *Steuerungskomponente*, welche die Veränderungen beobachtet und daraufhin entscheidet, welche Reaktion das System folgen lässt. Hierzu versucht sie unter einer festgelegten Strategie die aktuell auf dem Blackboard befindlichen Daten miteinander zu kombinieren, indem sie Operationen auf diesen ausführt und die Hypothesen bewertet. Eine beispielhafte Kommunikation von Aktoren und Sensoren mithilfe eines Blackboards zeigt die Abbildung 5.3.

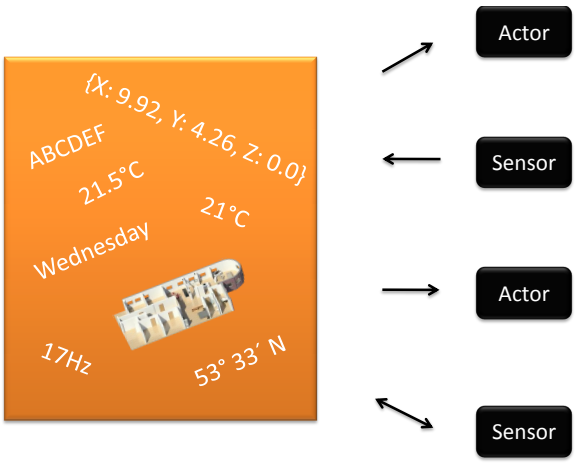


Abbildung 5.3: Beispielhafte Kommunikation über das Blackboard von Aktoren und Sensoren

Durch den Einsatz einer Blackboard-Architektur verliert man die positiven Effekte vollständig dezentraler Ansätze, da das Blackboard einen zentralen Gesamtspeicher darstellt der einen „Single Point of Failure“ hervorruft und eine Abhängigkeit aller Netzteilnehmer bezüglich dieser Einheit schafft. Außerdem können sich unnötige Rechenoperationen ergeben, falls Wissensquellen ihre Daten zu häufig senden und somit nur selten zur Problemlösung beitragen.
Ein großer Vorteil dieses Architekturmusters liegt darin, dass die Teilnehmer sich nicht eigenverantwortlich um die Gewinnung neuer Informationen bemühen müssen. Durch die Steuerungskomponente lassen sich Mechanismen implementieren, über welche interessierte Teilnehmer automatisch benachrichtigt werden, sobald ein entsprechendes Datum auf das Blackboard geschrieben wird. Ein weiterer Nutzen dieser Architektur ergibt sich aufgrund der offenen Struktur des Systems. Hier lassen sich, wie bereits in P2P-Netzwerken vorgestellt, neue Komponenten in das System integrieren, ohne weitere Teilnehmer kennen zu müssen.

Im Umkehrschluss bedeutet dieses, dass durch den Einsatz eines Architekturmusters dieser Form ein fehlertolerantes System entsteht, da bei einem Ausfall einer Wissensquelle alle weiteren Teilnehmer an der Problemlösung weiterarbeiten können. Jedoch ist es denkbar, dass der Ausfall einer Quelle bezweckt, dass notwendige Informationen nicht in die Kontextmodellierung integriert werden. In einem solchen Fall wäre es dem System unmöglich ein Ergebnis zu liefern. Deshalb sollte die Steuerungskomponente bei einem Ausfall einer Quelle benachrichtigt werden, sodass nicht auf die Information gewartet wird, sondern entweder ein unvollständiges Resultat geliefert oder der Lösungsprozess abgebrochen wird.

Fazit

In diesem Abschnitt wurden drei mögliche Ansätze für die Realisierung der Kommunikationsbasis vorgestellt und bewertet. Während strukturierte P2P-Netzwerke den „Single Point of Failure“ einer zentralen Einheit vollständig auflösen können, ergibt sich in Netzwerken dieser Form die Schwierigkeit, dass interessierte Anwendungen nicht automatisch benachrichtigt werden, sollten relevante Kontextinformationen erscheinen.
Das Pipes and Filters Muster bietet eine flexible Struktur, da Filter verschieden kombiniert und ausgetauscht werden können. Allerdings birgt es den Nachteil, dass Filter nur dann Informationen zur Kontexterstellung beitragen können, wenn sie an der Reihe sind, was zu inkorrekten Bezugsrahmen führen kann.
Eine Blackboard Architektur beinhaltet einen „Single Point of Failure“, da hier eine zentrale Einheit für die Kommunikation eingesetzt wird. Allerdings bringt ein Blackboard den Vorteil, dass Netzteilnehmer automatisch benachrichtigt werden, sobald bedeutende Informationen auftreten, ohne dass sich der Sender und der Empfänger der Nachrichten kennen müssen. Diese Systemeigenschaft stellt für das *Living Place Hamburg* einen maßgebenden Nutzen dar, weil hier stetig neue Sensoren und Applikationen Informationen abrufen oder zur Verfügung stellen. Außerdem haben die weiteren positiven Effekte, wie die hohe Fehlertoleranz dieses Architekturmusters, dazu geführt, dass als Kommunikationsbasis des hier vorgestellten Systems eine Blackboard Architektur implementiert werden soll.

5.2.2 Persistenz

Im bisherigen Verlauf dieser Arbeit wurde bereits erwähnt, dass bei der Kontextinterpretation der Faktor Zeit eine gewichtige Rolle spielt. Wenn der aktuelle Kontext beinhaltet, dass die Raumtemperatur steigt, muss der Interpreter vergangene Temperaturwerte einbeziehen, damit er diesen Zustand bemerkt. Dabei lassen sich mit der Betrachtung aufgezeichneter Sensorwerte Kenntnisse darüber erwerben, ob ein Wert derzeit steigt oder fällt sowie die Geschwindigkeit in der er dieses tut.

Des Weiteren lassen sich Historien von Sensorwerten mit dem Ziel untersuchen, Regeln aus ihnen abzuleiten. Vor diesem Hintergrund ist es denkbar, dass sich aus verschiedenen Sensorwerten erkennen lässt, dass der Bewohner des *Living Place Hamburg* bei einer Raumtemperatur von unter 20°C in der Regel die Heizung einschaltet. Mittels geeigneter Lernverfahren kann das System solche Strukturen wahrnehmen und entsprechend in das Regelwerk aufnehmen, sodass das Einschalten der Heizung in Zukunft vom Wohnbereich übernommen wird.

Zur Kontextinterpretation von Daten über die Zeit müssen die relevanten Daten im ersten Schritt in einem zentralen Speicher persistiert werden. Zu diesem Zweck lässt sich eine relationale Datenbank (Codd (1970)) verwenden, in der die Werte in Tabellen als Tupel organisiert werden. Diese Tabellen unterliegen jedoch festen Schemata, was besonders bei unstrukturierten Daten, die sich nur schwer in ein relationales Modell überführen lassen, zu sehr komplexen und schwer zu handhabenden Datenbanken führen könnte.
Besonders die hohe Anzahl an unterschiedlichen Sensoren im *Living Place Hamburg* würde zu einer individuellen Tabelle für die einzelnen Context Provider führen, was die Verwaltung der Daten immens erschwert. Außerdem wäre es möglich, dass ein Sensor nur vereinzelt ein bestimmtes Datum in seinen Informationen mitsendet und demnach kein konsistentes Datenformat der Nachrichten vorliegt. In diesem Fall könnte dieses zu Einfüge-Anomalien führen, sodass entsprechende Datensätze nicht eingefügt werden.

Abhilfe bei der Persistierung von unstrukturierten Daten soll eine Art von Datenbanken schaffen, die einen nicht-relationalen Ansatz verfolgen, indem sie keine festgelegten Tabellenschemata verwenden. Diese Form der Datenbankmanagementsysteme wird unter dem Begriff NoSQL-Datenbanken zusammengefasst. Die drei bekanntesten Typen dieser Datenbanken sollen im Folgenden genauer betrachtet werden, vgl. Leavitt (2010).

- **Key-Value Stores:** In diesem Ansatz werden die zu speichernden Werte mit Hilfe eines eindeutigen Schlüssels indiziert, damit sie anhand dieses Schlüssel abgefragt werden können. In diese Form der Datenbank lassen sich sowohl strukturierte als auch unstrukturierte Daten speichern. Ein populärer Vertreter von Key-Value Stores ist Amazon Dynamo (DeCandia u. a. (2007)). Hier werden komplizierte Datenbankzugriffe vermieden, indem direkt über den Schlüssel auf die Daten zugegriffen wird. Für die Skalierbarkeit des Gesamtsystems wird hier ein Netzwerk von gleichberechtigten Computern erstellt, in dem jeder Teilnehmer jede Aufgabe übernehmen kann.
- **Column-Oriented Databases:** Im Gegensatz zu relationalen Datenbanken werden Datensätze hier nicht zeilenweise, sondern in Spalten gespeichert, was erhebliche Auswirkungen auf die Lesereihenfolge hat. Während bei traditionellen Datenbanken die Reihenfolge ein weiteres Attribut des gleichen Tupels liefert, wird in diesem Ansatz das gleiche Attribut des nächsten Tupels gelesen. Dieser Ansatz bietet einen Vorteil, falls häufig gleiche Attribute und selten ganze Tupel gelesen werden sollen. Der Ansatz

der spaltenorientierten Datenbanken wird u.a. in BigTable von Google (Chang u. a. (2006)) und von Apache Cassandra (Lakshman und Malik (2010)) verfolgt. Letztere orientiert sich dabei an BigTable, verwendet jedoch ebenso Konzepte der Key-Value Stores.

- **Document-Based Stores:** Diese Datenbanken speichern und organisieren Datensätze als sog. Collections von Dokumenten. Die einzelnen Dokumente erlauben dabei das Hinzufügen beliebig vieler Felder mit jeglicher Länge. Innerhalb der Dokumente wird in der Regel ein Key-Value Ansatz verfolgt, wobei dokumentenbasierte Datenbanken von unstrukturierten Daten ausgehen. Zwischen den Dokumenten bestehen keine Relationen, wodurch sich gekapselte Elemente ergeben, die jeweils einen individuellen Aufbau und Inhalt besitzen. Für das Abfragen der Dokumente wird hier jedes Dokument mit einem eindeutigen Bezeichner versehen. Der Leitgedanke dieser Form von Datenbanken liegt darin, zusammengehörige Daten gebündelt zu speichern, mit dem Ziel komplexe Abfragen über mehrere Tabellen, wie es häufig in relationalen Datenbanken der Fall ist, zu verhindern. Zwei verbreitete Vertreter dieser Datenbankmanagementsysteme sind CouchDB[1] und MongoDB[2]. Für weiterführende Informationen zu CouchDB sei an dieser Stelle auf Anderson u. a. (2010) und zu MongoDB auf Plugge u. a. (2010) verwiesen.

Das Ziel dieser schemafreien Ansätze ist nicht die vollständige Ablösung der relationalen Datenbanken, sondern sie sollen Alternativen für die Verwaltung von Daten, die nur schwer in relationale Modelle zu überführen sind, liefern. Dabei steht die unkomplizierte Nutzung, die Skalierbarkeit und die Geschwindigkeit im Vordergrund von NoSQL-Datenbanken. Da diese Eigenschaften für die Architektur des *Living Place Hamburg* eine entscheidende Rolle spielen, fiel die Entscheidung in dieser Arbeit auf einen nicht-relationalen Ansatz zur Persistierung der Kontextinformationen.
Aufgrund der Datenvielfalt, welche durch die verschiedenen Sensoren generiert wird, soll im hier vorgestellten System eine dokumentenbasierte Datenbank zum Einsatz kommen, da hier die Möglichkeit besteht, die Informationen eines Context Providers gegebenenfalls mit weiterem Wissen anzureichern und in individuellen Dokumenten zu speichern. Für die unterschiedlichen Sensortypen wäre es darüber hinaus denkbar, dass jeweils Collections erstellt werden, in die beispielsweise sämtliche Temperaturwerte eingetragen werden.

Neben der Form der Archivierung der Datensätze stellt es einen weiteren elementaren Aspekt dar, wie die Daten aus der Datenbank zu lesen sind. Dieses liegt darin begründet, dass während der Kontextinterpretation häufig gespeicherte Daten abzufragen sind, damit sie mit neuen Informationen in Verbindung gesetzt werden können. In diesem Zusammenhang ist die Geschwindigkeit und die Struktur in der die Daten geliefert werden von beson-

[1] http://couchdb.apache.org/
[2] http://www.mongodb.org/

derer Bedeutung. Deshalb wird im folgenden Abschnitt der Map/Reduce Ansatz vorgestellt, über welchen dokumentenorientierte Datenbanken für gewöhnlich ihre Daten zur Verfügung stellen.

Ihren Ursprung besitzen Map/Reduce-Funktionen in der funktionalen Programmierung und dienten Google Inc. im Jahre 2004 als Inspiration das MapReduce Framework zu entwickeln (Dean und Ghemawat (2004)). Map/Reduce Algorithmen bestehen in ihrer ersten Phase aus der Anwendung des Mappings, bevor im zweiten Schritt die Reduce-Funktionen ausgeführt werden. Die einzelnen Funktionen werden vom Entwickler anwendungsspezifisch implementiert. Zur Bearbeitung der Daten werden den Map-Funktionen Datensätze als Schlüssel/Werte-Paare übergeben, die sie entsprechend ihrer Implementierung bearbeiten und als Zwischenergebnisse abspeichern. Die Reduce-Funktionen sind im darauffolgenden Schritt dafür zuständig, ein Endergebnis zu erarbeiten. Hierzu wird von den Funktionen eine Ergebnisliste erstellt, die zu jedem Schlüssel/Werte-Paar lediglich ein Ergebnis bereithält.
In Dean und Ghemawat (2004) wird ein einfaches praktisches Beispiel präsentiert, welches die Aufgaben der jeweiligen Funktionen sehr anschaulich beschreibt. In diesem Beispiel sollen die Vorkommnisse der einzelnen Wörter in einem Textdokument gezählt werden. Hierzu werden im Pseudo Code die folgenden Funktionen erstellt.

Listing 5.1: Beispielhafte Map-Funktion

```
0  map(String key, String value):

   // key: document name
   // value: document contents

5  for each word w in value:
     EmitIntermediate(w, "1");
```

Listing 5.2: Beispielhafte Reduce-Funktion

```
0  reduce(String key, Iterator values):

   // key: a word
   // value: a list of counts

5  int result = 0;
   for each v in values:
     result += ParseInt(v);
   Emit(AsString(result));
```

Die Map-Funktion 5.1 erstellt zu jedem Wort ein Schlüssel/Werte-Paar mit dem Wert 1. Dies bedeutet, dass das Zwischenergebnis letztlich so viele 1-Einträge besitzt wie es Wörter im Dokument gibt. Die Reduce-Funktion 5.2 besitzt nun die Aufgabe die Vorkommen der einzelnen Wörter aufzusummieren. Dieses Beispiel ist sehr einfach gewählt und gewinnt an Komplexität, sobald das Dokument auf mehrere Map-Prozesse aufgeteilt wird und somit die Zwischenergebnisse zusammengeführt werden müssen.

5.3 Systemüberblick

In Abschnitt 3.2.4 wurden bereits drei Ansätze vorgestellt, anhand derer eine Kontextinterpretation, im Hinblick auf eine Aktivitätserkennung, durchführbar ist. Des Weiteren wurde beschrieben, dass es sinnvoll ist, eine Kombination verschiedener Vorgehensweisen anzuwenden, da sich die Interpretation des Bezugsrahmens üblicherweise in mehrere Ebenen gliedert und es keinen allgemeingültigen Ansatz für sämtliche Stufen gibt.
Dieser Abschnitt präsentiert ein geeignetes Zusammenspiel der modellunabhängigen, der modellbasierten und der ontologiebasierten Kontextinterpretation sowie deren Kommunikation mit den weiteren Systemkomponenten. Die Grundlage hierzu soll die in einer Vorarbeit entstandene Ausarbeitung „An Environment for Context-Aware Applications in Smart Homes“ (Ellenberg u. a. (2011)) liefern.

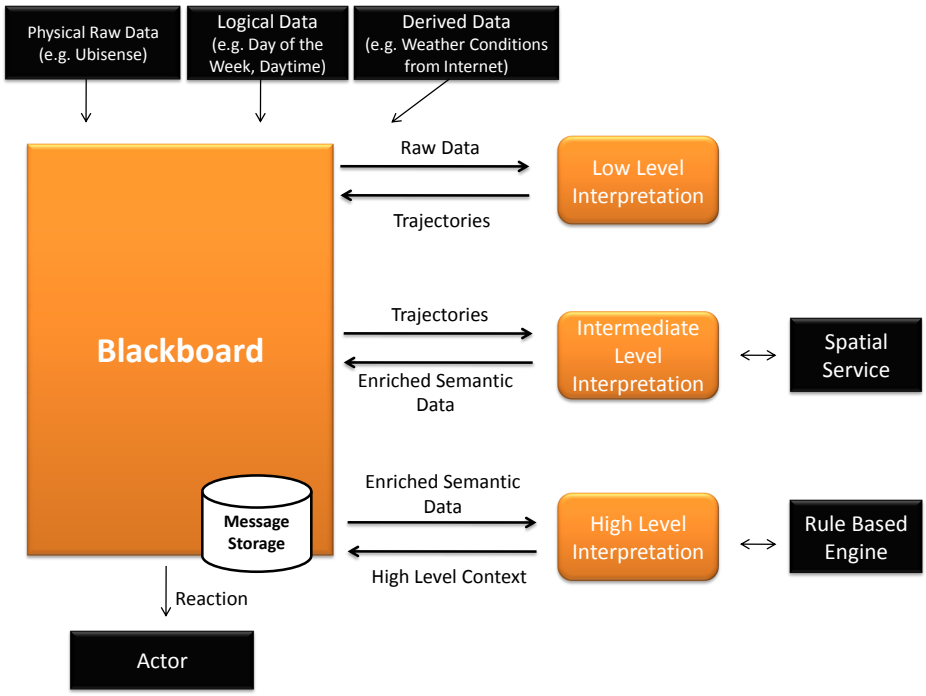


Abbildung 5.4: Beispielhafte Kommunikation verschiedener Level zur Erstellung eines High-Level Kontextes

5.3.1 Systemkomponenten

Das Gesamtsystem setzt sich auf der einen Seite aus dem Blackboard und auf der anderen Seite aus den sog. Interpretationslevel zusammen. Die Interaktionen der einzelnen Kompo-

nenten wird beispielhaft auf der Abbildung 5.4 dargestellt. Dabei kommen in dieser Architektur drei verschiedene Sensorarten zum Einsatz. Die physikalischen Rohdaten sind dabei analog zu den in 3.2.3 vorgestellten internen Context Providern zu betrachten. Die externen Informationsquellen werden hingegen in logische und abgeleitete Daten unterteilt, mit dem Ziel, die Sensordaten präziser voneinander unterscheiden zu können. Die logischen Informationen stellen in dieser Arbeit Werte dar, die zum aktuellen Zeitpunkt invariant und somit nicht von äußeren Einflüssen veränderbar sind. Außerdem lassen sich zukünftige logische Daten bei einem gegebenen Startzeitpunkt durch eine Berechnung vorhersagen. Beispiele für diese Art der Informationen sind der Wochentag, die Uhrzeit und die Jahreszeit. Die abgeleiteten Daten sind hingegen äußere Bedingungen, die unter Umständen ebenso nicht von außen änderbar sind, sich jedoch nicht aus der natürlichen Umgebung ergeben und somit nicht antizipiert werden können. Sie müssen daher zu jedem Zeitpunkt erneut von einer externen Einheit erfragt werden.

Nachdem die Verantwortlichkeiten des Blackboards sowie des Message Storages bereits in den Abschnitten 5.2.1 und 5.2.2 erläutert wurden, soll nun die Aufgabe der einzelnen Interpretationslevel beleuchtet werden.
Die einzelnen Level sollen in dieser Architektur in der Form zusammenarbeiten, dass jede Ebene die bisherigen Informationen mit Wissen anreichert, sodass letztlich ein konsistenter Gesamtkontext zur Verfügung steht. Dabei erwartet jede Stufe eine bestimmte Form der Nachrichten und geht in den höheren Schichten davon aus, dass eine entsprechende Vorverarbeitung der Daten stattgefunden hat. In dieser Arbeit sollen sowohl physikalische, logische und abgeleitete Informationen in die Kontexterstellung einfließen. Bei der Verarbeitung von physikalischen Rohdaten soll der Fokus hier insbesondere auf Positionsdaten, welche im *Living Place* aufgezeichnet wurden, liegen. Diese unverarbeiteten Koordinaten sollen dazu genutzt werden, Schritt für Schritt mit weiterem Wissen angereichert zu werden, mit der Absicht letztlich genaue Bewegungsprofile des Bewohners zu erstellen. Diese Profile spielen bei der Aktivitätserkennung eine wesentliche Rolle, da sie u.a. Aufschluss darüber geben können, wohin der Bewohner aktuell geht bzw. Schätzungen darüber zulassen. Außerdem lassen sich mögliche Aktivitäten eingrenzen, sofern bekannt ist, in welchem Bereich der Wohnung sich der Bewohner befindet. Beispielsweise kann die Aktivität „Duschbad nehmen“ ausgeschlossen werden, sollte sich die Person im Wohnzimmer aufhalten. Mit der Aktivitätserkennung aufgrund von Positionsdaten befassen sich u.a. Tapia u. a. (2004) und Liao u. a. (2005).
Mit dem folgenden Abschnitt soll dargestellt werden, inwiefern die drei Level zusammenarbeiten, mit dem Ziel, aus den rohen Positionsdaten einen High-Level Kontext zu erstellen.

Low Level Interpretation

Die „Low Level Interpretation“ arbeitet mit den unverarbeiteten Positionsdaten und soll mittels einer modellunabhängigen Interpretation realisiert werden. Dieser Ansatz wird für die Inter-

pretation des untersten Level gewählt, da es hier häufig nicht erforderlich ist, ein aufwändiges Modell zu erstellen. An dieser Stelle sollen die Daten unabhängig von einem expliziten Modell untersucht und mit möglichst viel Wissen bestückt werden, ohne dass spezielle Kenntnisse über die Umgebung vorliegen. Der in diesem System gewählte Ansatz wurde durch die Disziplin der Bildverarbeitung inspiriert. Hier wird versucht, Objekte oder menschliche Bewegungen zu erkennen, indem diese als einfache Low-Level zweidimensionale Merkmale, innerhalb einer „Region of Interest“, beschrieben werden, vgl. Gavrila (1999).
Die Wissensgenerierung auf diesem Level bietet verschiedene Möglichkeiten, wovon die „Low Level Interpretation“ in diesem Fall die folgenden Aufgaben besitzt:

- **Trajektorien bilden:** Sollte sich der Bewohner sich in seiner Wohnung von einem Bereich in einen anderen bewegen, liegt im ersten Schritt lediglich ein Satz von Koordinaten vor, der in eine sinnvolle Reihenfolge zu bringen ist. Hierzu müssen Ausreißer unter den einzelnen Positionsdaten identifiziert und verarbeitet werden, sodass die korrekt gemessenen Werte zu einer Trajektorie zusammengefasst werden können und abschließend eine realitätsnahe Verfolgung des genommenen Weges möglich ist. Diese bereinigten Trajektorien werden daraufhin zur weiteren Interpretation vergangener Pfade an das höhere Level übermittelt.

- **Trajektorien clustern:** Die erstellten Trajektorien sollen mit einem geeigneten Clustering-Algorithmus untersucht werden, um ähnliche Pfade in Gruppen zusammenzufassen. Diese generierten Cluster sollen Hinweise darauf liefern, welche Wege sich innerhalb des Wohnbereichs wiederholend ergeben und somit relevante Trajektorien für die Aktivitätserkennung darstellen. Des Weiteren lässt sich anhand der Cluster erkennen, welche Wege häufig und welche nur sehr selten gelaufen werden.

- **Cluster bereitstellen:** Damit die höheren Level die Cluster in die Kontexterstellung einbeziehen können, reicht es nicht aus, sämtliche Trajektorien eines Clusters zu übergeben. Hieraus lässt sich beispielsweise nicht erkennen, ob sich eine Person auf ihrem aktuellen Pfad innerhalb eines Clusters bewegt. Aus diesem Grund muss eine Vorgehensweise gefunden werden, einen Rahmen um die einzelnen Cluster zu legen. Eine Möglichkeit zur Umrandung der Cluster ist die Berechnung einer *konvexen Hülle*. Diese sowie konvexe Mengen, werden in Klein (1997) wie folgt definiert:

  konvex „Eine Teilmenge K vom $\mathbb{R}^d$ heißt *konvex*, wenn sie zu je zwei Punkten p, q auch das Liniensegment pq enthält.“

  konvexe Hülle „ Für eine beliebige Teilmenge A des $\mathbb{R}^d$ ist

  $$ch(A) = \bigcap_{\substack{K \supseteq A \\ K \text{ konvex}}} K \tag{5.1}$$

  die kleinste konvexe Menge, die A enthält.“

Diese Hülle besitzt allerdings die Schwierigkeit, dass sie in einigen Fällen nicht die ideale Umrandung zu einer gegebenen Punktemenge findet. Beispielsweise würde die konvexe Hülle, die den Buchstaben „C“ umrandet, die gleiche Hülle wie für den Buchstaben „S“ ergeben, vgl. Galton (2008). Wenn das Ziel, wie in diesem Fall, die Umrandung genau derjenigen Regionen ist die von Punkten abgedeckt wird, lässt sich alternativ die konkave Hülle einsetzen. Diese kann als eine Erweiterung der konvexen Hülle betrachtet werden, die es zusätzlich erlaubt, nicht-konvexe Polygone zu generieren. Ein Algorithmus zur Berechnung dieser Hülle wird in Moreira und Santos (2007) vorgestellt. Zur Veranschaulichung des Unterschiedes zwischen einer konvexen und einer konkaven Hülle wird in der Abbildung 5.5 eine jeweilige Hülle zu einer Punktemenge dargestellt.

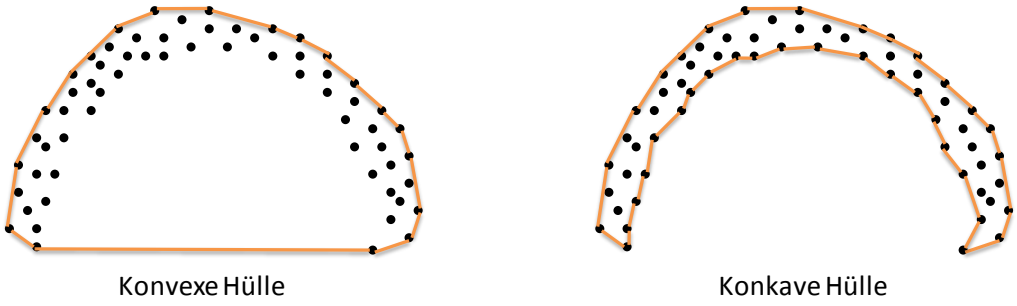


Abbildung 5.5: Konvexe und konkave Hülle einer Punktemenge

- **Zeitliche Informationen liefern:** Beim Clustering ist zu berücksichtigen, dass ausschließlich Trajektorien zusammengefasst werden, die insbesondere in ähnlichen Zeitabschnitten (z.B. Tageszeit), jedoch auch in weiteren vergleichbaren äußeren Einflüssen, aufgetreten sind. Dies liegt darin begründet, dass in den verschiedenen Tageszeiten unterschiedliche Aktivitäten stattfinden und folglich die Pfade anderen Tätigkeiten zuzuordnen sind. Für die Morgen-Szenarien sind die zeitlichen Informationen weiterhin von Bedeutung, weil eine aus mehreren Aktionen zusammengesetzte Aktivität gegebenenfalls eine zeitliche Reihenfolge der Ausführung aufweisen muss, damit sie erkannt wird.
- **Logische und abgeleitete Werte bereitstellen:** Die „Low Level Interpretation“ hat in dieser Architektur weiterhin die Aufgabe, die logischen Daten zu berechnen und zur Verfügung zu stellen. Darüber hinaus soll diese Komponente in der Lage dazu sein, abgeleitete Daten wie das Wetter oder den ersten Termin des Bewohners zu erfragen und in die Kontexterstellung einzubeziehen. In diesem Zusammenhang wäre es denkbar, dass logische Werte, wie die Jahreszeit oder der Wochentag, die Trajektorien und

daher den Kontext beeinflussen. Aus diesem Grund sollten diese Daten ebenso in das Clustering einfließen. Die abgeleiteten Daten können auf den Bezugsrahmen dahingehend Auswirkungen haben, dass die Weckzeit abhängig vom Wetter und der Entfernung zum ersten Termin variieren kann, was u.a. zum Szenario „hektischer Morgen“ (siehe 3.1) führen kann. Des Weiteren lassen sich mit Hilfe logischer und abgeleiteter Daten auf den höheren Level Schätzungen darüber anstellen, welche Aktivitäten und Trajektorien sich aufgrund der Werte ergeben.

Intermediate und High-Level Interpretation

Die „Intermediate Level Interpretation“ stellt in dieser Architektur den modellbasierten Ansatz zur Kontextinterpretation dar. Hier soll ein Service mit dem Ziel bereitgestellt werden, die erhaltenen Informationen mit semantischem Wissen anzureichern, indem räumliche Entitäten und ihre Relationen zueinander verwendet werden. Zu diesem Zweck bietet sich ein dreidimensionales Gebäudemodell an, mit Hilfe dessen die Services u.a. die übermittelten Cluster und Trajektorien mit semantischem Wissen versehen. Vor diesem Hintergrund soll zu einer gegebenen Trajektorie erkannt werden, in welchen Bereichen sich der Bewohner bewegt. Hierzu werden die Koordinaten mit den im Modell hinterlegten Arealen, *Functional Spaces* und *Range Spaces*, verknüpft. Ein *Functional Space* beschreibt in diesem System einen Bereich um ein Objekt herum, innerhalb dessen ein Mensch detektiert werden muss, damit er mit diesem interagieren kann. Diese Umgebungen lassen bereits erste Schlussfolgerungen über die Benutzung von Geräten zu. Sollte eine Person im *Functional Space* des Herds erkannt werden, besteht eine hohe Wahrscheinlichkeit, dass dieser in Betrieb genommen wird. *Range Spaces* bezeichnen hingegen das Areal, das in dem Sichtbereich eines Sensors liegt, wie beispielsweise einer Kamera (Bhatt u. a. (2009)). Anhand dieser Bereiche kann das System erkennen, ob z.B. bei einer Videokonferenz die richtige Kamera ausgewählt ist bzw. es Sensoren gibt, die derzeit keine Informationen zur Kontexterfassung beitragen können. Eine Überschneidung der Spaces ist in diesem System durchaus möglich.
Für die Schätzung zukünftiger Aufenthaltsorte soll die „Intermediate Level Interpretation“ die übergebenen konkaven Hüllen verwenden. Anhand dieser lässt sich überprüfen, ob der Bewohner sich aktuell in einem Cluster bewegt. Sollte dieses der Fall sein, können entsprechende Schlussfolgerungen daraus gezogen werden, in welchen Bereich der Wohnung sie gehen wird.

Die „High Level Interpretation“ soll in diesem System von einem ontologiebasierten Ansatz übernommen werden. Innerhalb der Ontologie findet hierzu eine Modellierung des gesamten Wissens über die Umwelt statt. An dieser Stelle werden demnach neben den zur Wohnumgebung gehörenden Objekten, ebenfalls die Aktivitätsmuster gespeichert. Hiermit geht einher, dass an dieser Stelle die einzelnen Aktivitäten sowie die zugehörigen Aktionen definiert werden müssen. Durch diese Vorgehensweise gewährleistet die Ontologie daher eine

allgemeingültige Wissens- und Begriffsgrundlage, auf deren Basis jegliche Teilnehmer im Netzwerk kommunizieren.
Darüber hinaus soll die Ontologie mit einem regelbasierten System zusammenarbeiten, um die übermittelten Kontextinformationen zu interpretieren. Da die Rohdaten der Sensoren bei der „High Level Interpretation“ bereits durch zwei Interpretationsebenen mit Wissen angereichert wurden, sollen auf dieser Stufe die in den Szenarien vorgestellten Regeln zu den Aktivitäten implementiert werden. Folglich wird auf diesem Level die Erkennung von Aktionen durchgeführt und eine entsprechende Reaktion der Wohnung initiiert.

5.3.2 Zusammenspiel der Komponenten

Im Sequenzdiagramm 5.6 wird beispielhaft ein Zusammenwirken der einzelnen Level während der Interpretation von Positionsdaten präsentiert. Hier wird demonstriert, wie die Koordinaten an die Low Level-Komponente versendet werden. Diese hat daraufhin die Aufgabe, Trajektorien zu kreieren und die logischen und abgeleiteten Kontextinformationen zu ermitteln. Daraufhin liefert die erste Ebene die ihr bekannten Daten an das Intermediate Level. Hier werden die Daten mit semantischem Wissen bestückt, bevor sie zur dritten Stufe, der ontologiebasierten Interpretation, versendet werden. Auf diesem Level wird versucht, konkrete Aktivitäten zu identifizieren, damit die Aktoren im System auf den Kontext reagieren können. Hierzu veranlasst das High-Level das Versenden entsprechender Nachrichten über das Blackboard.

5.4 Aktivitäten und deren Reihenfolge in den Szenarien

Aus dem Analyse-Kapitel ist hervorgegangen, dass die einzelnen Szenarien aus einer Reihe von zu erkennenden Aktivitäten bestehen, die es im ersten Schritt zu identifizieren gilt und darauffolgend in eine zeitliche Reihenfolge zu bringen sind (siehe Abschnitt 3.2.5). Infolgedessen sollen in diesem Abschnitt die vom System zu erkennenden Aktivitäten innerhalb der Szenarien vorgestellt und gezeigt werden, welche zeitlichen Beziehungen zwischen den Aktionen herrschen.
Die Szenarien setzen sich aus den folgenden Aktivitäten zusammen:

- **Entspannter Morgen + Arbeitstag:** Aufstehen, BadezimmerBetreten, ZähnePutzen, DuschbadNehmen, BadezimmerVerlassen, Ankleiden, AusgiebigesFrühstückZubereiten, FrühstückVerzehren, NachrichtenLesen, WohnungVerlassen
- **Hektischer Morgen + Arbeitstag:** Aufstehen, BadezimmerBetreten, ZähnePutzen, Waschen, BadezimmerVerlassen, Ankleiden, KurzesFrühstückZubereiten, FrühstückVerzehren, WohnungVerlassen

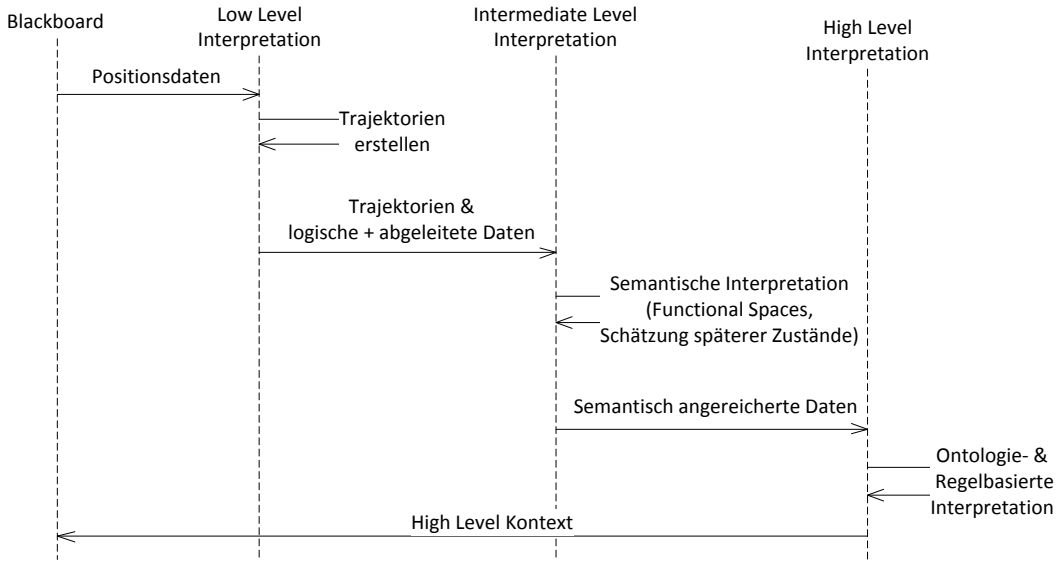


Abbildung 5.6: Beispielhaftes Zusammenspiel der drei Level bei der Interpretation von Positionsdaten

- **Morgen am Wochenende:** Aufstehen, BadezimmerBetreten, ZähnePutzen, DuschbadNehmen, BadezimmerVerlassen, Ankleiden, AusgiebigesFrühstückZubereiten, FrühstückVerzehren, Entspannen

Diese Szenarien enthalten einige Aktionen, die einem strikten zeitlichen Ablauf unterliegen und daher in einer chronologisch geordneten Reihenfolge auszuführen sind. Diese Tätigkeiten werden in dieser Arbeit als ein Tupel der Form *seq_strikt*(x_1,..., x_n) beschrieben. Andere Tätigkeiten können frei in ihrer Ausführungsreihenfolge kombiniert werden, da sie vollständig unabhängig voneinander sind und werden durch *seq_or*(x_1,...,x_n) dargestellt. Hierunter sind diejenigen Aktivitäten zu verstehen, die keine zeitgleichen Handlungen erlauben. Tätigkeiten, die parallel zueinander ausführbar sind, werden in Tupeln der Art *parallel*(x_1,...,x_n) zusammengefasst. Diese Tupel-Schreibweise soll im Folgenden dazu dienen, zwei der drei Szenarien anhand eines vereinfachten Beispiels in eine zeitlich partiell geordnete Reihenfolge zu überführen. Auf das Szenario „Hektischer Morgen + Arbeitstag“ wird verzichtet, da die zeitlichen Abhängigkeiten sehr ähnlich zu denen im Szenario „Hektischer Morgen + Arbeitstag“ aussehen.

1. **Entspannter Morgen + Arbeitstag:**

   seq_strikt*(**Aufstehen, BadezimmerBetreten, ***seq_or*(**DuschbadNehmen, ZähnePutzen**)**, BadezimmerVerlassen, ***seq_or*(**AusgiebigesFrühstückZubereiten, ***paral-

***lel*(**FrühstückVerzehren, NachrichtenLesen**)**, Ankleiden**)**, WohnungVerlassen**)**

2. **Morgen am Wochenende:**

   ***seq_strikt*(**Aufstehen, ***seq_or*(*seq_strikt*(**BadezimmerBetreten, ***seq_or*(** Duschbad-Nehmen, ZähnePutzen**)**, BadezimmerVerlassen**)**, ***seq_strikt*(** AusgiebigesFrühstückZubereiten, FrühstückVerzehren**))**, Entspannen**)**

5.5 Systementwicklung

Im *Living Place Hamburg* entstehen kontinuierlich neue Teilsysteme, die von verschiedenen Personen, in einem iterativen Entwicklungsprozess, realisiert werden. Ebenso werden bereits bestehende Systeme gewartet und stetig weiterentwickelt. Folglich gehört es zu den wesentlichen Zielen strukturierte und wartbare Systeme zu implementieren, um einen langfristigen Einsatz im *Living Place Hamburg* zu gewährleisten. Ein weiterer essentieller Aspekt ist die fehlerfreie Funktionalität der entwickelten Komponenten. Ein inkorrekt arbeitendes Teilsystem kann weitreichende Folgen haben, da dies eine falsche Kontexterstellung und somit eine ungeeignete Reaktion des Systems hervorrufen kann.
Die Konsequenz dieser Aspekte beinhaltet, dass die zum Einsatz kommende Software fundiert getestet und evaluiert sein muss. Zur Erfüllung dieser Anforderungen sollte die Systemerstellung von Beginn an entsprechende System- und Usabilitytests einbeziehen. Geeignete Entwicklungsmethoden hierzu lauten „Test-Driven Development“ (TDD) und „User-Centered Designprozess“ (UCD).
Erstere Methode wurde von Kent Beck (Beck (2002)) als ein Teil des Extreme Programming entwickelt und stellt das Testen bei der Entwicklung von Software in den Vordergrund. Sie zeichnet sich dadurch aus, dass vor dem Schreiben des Quellcodes Tests generiert werden, damit der Entwickler bereits frühzeitig detailliert spezifiziert, welche Funktionen der Code erfüllen soll. Die einzelnen iterativen Schritte des „Test-Driven Development“ werden auf der Abbildung 5.7 veranschaulicht.

Der Ansatz des „User-Centered Design“ befasst sich mit dem Ziel Software zu entwickeln, die eine hohe Usability aufweist. Grechenig u. a. (2010) identifizieren vier Phasen, die in diesem Prozess in mehreren Iterationen durchlaufen werden.

1. **Anforderungen:** Hier werden die Aufgaben der Software im Hinblick auf die Eigenschaften des Benutzers erfasst. Des Weiteren wird die Umgebung und der Bezugsrahmen der Nutzung definiert.
2. **Design:** Bei dem Design des Systems stehen die Wünsche, Erfahrungen und Fachwissen im Vordergrund.

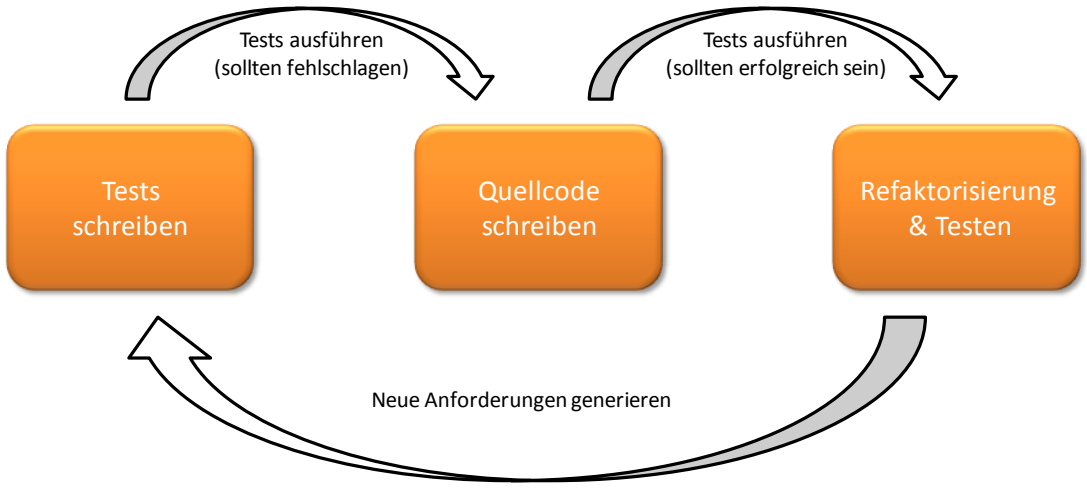


Abbildung 5.7: Iterative Schritte beim Test-Driven Development

3. **Prototyping:** An dieser Stelle werden die Designentscheidungen in Form von Prototypen implementiert. Diese können zwischen einfachen Entwürfen bis hin zu real benutzbaren Systemen variieren.
4. **Evaluierung:** In diesem Schritt werden die entwickelten Prototypen getestet und evaluiert. Die Rückmeldung wird im darauffolgenden Iterationsschritt dazu verwendet, die neuen Ideen in das System zu integrieren.

Durch das mehrfache Durchlaufen der Schritte ergeben sich auf der einen Seite längere Entwicklungszeiten, auf der anderen Seite bekommt man kontinuierlich ein Feedback des Anwenders, sodass Wünsche und Anregungen Schrittweise in das System eingebettet werden können.

Diese vorgestellten Entwicklungsmethoden erfordern eine umfangreiche Testumgebung, mit Hilfe derer sich die Systeme evaluieren lassen. Insbesondere bei der Realisierung von Anwendungen durch „Test Driven Development“ und einem „User-Centered Designprozess“ Ansatz, werden diesen stetig Tests unterzogen. Dabei tritt häufig der Fall auf, dass die Applikationen in Situationen zu testen sind, die aktuell nicht vorliegen. Dieser Aspekt spielt speziell in der Aktivitätserkennung der drei Morgen-Szenarien eine wesentliche Rolle. Hier soll das System in diversen Kombinationen von äußeren Einflüssen unterschiedliches Verhalten aufweisen, da die Veränderung einzelner Faktoren zwangsläufig den Kontext beeinflusst. Für die Tests ist es daher nicht sinnvoll, nur an Wochenenden das Szenario „Morgen am Wochenende“ zu evaluieren oder nur an sonnigen Tagen entsprechende Applikationen zu analysieren. Diese Vorgehensweise wäre äußerst ineffektiv und nur sehr schwer durchzuführen, da sich gewisse Situationen nur in Ausnahmefällen ergeben.

Für den *Living Place Hamburg* ist aus den genannten Gründen ein Kontextsimulator zur Erzeugung spezifischer Kontexte, die als Basis zur Generierung systematischer Testumgebungen dienen, unerlässlich. Dieser Simulator soll eine intuitive Bedienung aufweisen, sodass es Entwicklern innerhalb kurzer Zeit möglich ist, in der Wohnumgebung die gewünschte Situation zu erzeugen.
Da sich diese Arbeit u.a. mit der Bereitstellung von realen logischen, abgeleiteten und verarbeitenden Sensor-Rohdaten befasst, soll das hier vorgestellte System zusätzlich dazu in der Lage sein, diese zu simulieren. Dies bedeutet, dass in dieser Anwendung eine Funktionalität realisiert werden muss, mittels derer die Möglichkeit offeriert wird, jeglichen relevanten Teilnehmern die Einbeziehung der simulierten Daten vorzuschreiben. Für die Realisierung der in dieser Arbeit vorgestellten Szenarien müssen dementsprechend die folgenden Kontextinformationen real erfragt und simuliert werden können.

- **Datum:** Das aktuelle Datum kann den Kontext z. B. dahingehend verändern, dass ein Feiertag für das System wie ein Wochenendtag zu behandeln ist.
- **Wochentag:** Der Wochentag gibt Aufschluss darüber, ob es sich um einen Werktag oder um einen Wochenendtag handelt. Außerdem lassen sich häufig an gleichen Tagen identische Aktivitäten beobachten.
- **Tageszeit:** Dieser Aspekt ist in mehrerer Hinsicht von Bedeutung, weil sich aus der Tageszeit bereits ableiten lässt, welche Aktivitäten sich ergeben können. Beispielsweise lässt sich ein Morgen-Szenario am späten Abend ausschließen. Des Weiteren kann anhand der Tageszeit ermittelt werden, ob der Sonnenaufgang bereits stattgefunden hat und beispielsweise ein Einschalten der Lichter nicht notwendig ist.
- **Wetterbedingungen:** Wie bereits beschrieben, fließen die aktuellen Wetterbedingungen vor dem Hintergrund ein, dass die Weckzeit bei schlechtem Wetter entsprechend früher generiert wird als bei sonnigen Bedingungen, da die Anreise zu einem Termin in diesem Fall in der Regel mehr Zeit beansprucht. Zu Testzwecken sollte der Simulator deshalb sowohl reale Wetterdaten abfragen können, allerdings ebenso die Angabe fiktiver Wetterbedingungen erlauben.
- **Kalendereinträge:** Das System muss auf der einen Seite dazu in der Lage sein, Einträge wie Beginn und Ort eines Termins auszulesen, mit der Absicht diese den weiteren Teilnehmern zur Verfügung zu stellen. Auf der anderen Seite müssen aus dem Simulator Kalendereinträge mit Ort- und Zeitangabe angelegt werden können, falls ein gewünschter Termin noch nicht existiert.

5.6 Fazit

In diesem Kapitel wurde mit dem *Living Place Hamburg* das *Smart Home*, in dem diese Arbeit stattfindet, vorgestellt. Im Hinblick auf diese Umgebung ließen sich zentrale Designentscheidungen bezüglich der Architektur treffen. Dabei konnte aufgezeigt werden, dass sich für die Realisierung der Kommunikationsschnittstelle ein Blackboard eignet, über welche die Context Provider und Aktoren Nachrichten austauschen. Diese versendeten Nachrichten sollen in einem weiteren Schritt in einer dokumentenorientierten Datenbank gespeichert werden, damit sie zu einer weiterführenden Verarbeitung zur Verfügung stehen.
Des Weiteren konnten die Aufgaben der modellunabhängigen, der modellbasierten und der ontologiebasierten Kontextinterpretation zur Aktivitätserkennung sowie die Aktivitäten selbst identifiziert werden. Letztere wurden auf der Basis der in 3.1 vorgestellten Morgen-Szenarien ermittelt.

Zudem wurde in diesem Abschnitt ein Kontextsimulator vorgestellt, mit Hilfe dessen den Entwicklern während der Implementierung neuer Anwendungen die Möglichkeit geboten wird, den Teilkomponenten im *Living Place* gewisse Situationen vorzutäuschen, vor dem Hintergrund eine erforderliche Testumgebung zu simulieren. Das Ziel dieses Simulators liegt demnach im nahtlosen Wechsel zwischen simulierten und realen Daten, indem im *Living Place* ohne großen Aufwand erforderliche Testumgebungen herstellbar sind, ohne dass die Daten und Aufgaben realer Prozesse geändert werden müssen.
Hierzu wurden die zu manipulierenden Sensordaten vorgestellt und somit die Aufgaben des Simulators herausgearbeitet.

Die aus dem Design gewonnenen Erkenntnisse sollen im folgenden Kapitel die Grundlage für ein System bilden, auf dessen Basis insbesondere Aktivitäten des Menschen erkannt und systemseitig unterstützende Reaktionen initiiert werden können. Darüber hinaus sollen sämtliche Komponenten mit der Zielsetzung eines langfristigen Einsatzes entwickelt werden, sodass sie neben der Aktivitätserkennung ebenso an weiteren Kontextinterpretationen teilnehmen können.

6 Realisierung

In diesem Kapitel soll unter Berücksichtigung des Designs und der Analyse eine konkrete Umsetzung des vorgestellten Systems erfolgen. In diesem Zusammenhang wird im ersten Schritt die Integration einer Blackboard-Architektur in den *Living Place Hamburg* vorgestellt, bevor die hier gewählte NoSQL-Datenbank mit dem Ziel präsentiert wird, ein geeignetes Persistenzmodul in diese Applikation einzubetten. Außerdem wird die Realisierung einer modellunabhängigen Kontextinterpretation präsentiert sowie kurz auf die Kombination mit weiteren, in Abschnitt 5.3.1 vorgestellten, Interpretationslevel eingegangen.

6.1 Blackboard

Für die Realisierung einer Blackboard-Architektur spielt der Begriff des *Publish/Subscribe-Modells* eine wesentliche Rolle. Dieses Modell setzt die Grundidee eines Blackboards in der Weise um, dass *Subscriber* ihr Interesse für Events oder Muster von Events bekunden und daraufhin benachrichtigt werden, sobald ein *Publisher* ein entsprechendes Datum veröffentlicht (Eugster u. a. (2003)).
Zur Identifizierung eines geeigneten Systems wurden hierzu in Otto und Voskuhl (2010) verschiedene Publish/Subscribe-Systeme evaluiert. Dabei konnte der Message Broker ActiveMQ[1] der Apache Foundation besonders an Profil gewinnen, da dieser eine ausführliche Dokumentation, eine Unterstützung vieler Programmiersprachen und eine hohe Performance anbietet. Der Nachrichtendurchsatz des ActiveMQ wird in Henjes u. a. (2007) unter verschiedenen Bedingungen, wie einer variierenden Anzahl Subscriber und Publisher und den Einsatz von Filtern, untersucht. Der ActiveMQ Message-Broker besteht aus mehreren Teilkomponenten, die auf der Abbildung 6.1 dargestellt werden.

Die „Connectors“ bieten eine Schnittstelle für diverse Plattformen an, mit dem Ziel den Zugriff auf die beiden Kommunikationsmechanismen zu gewährleisten. Die „Network Services“ dieses Systems befassen sich u.a. mit dem automatischen Auffinden von Clients und weiteren Brokern im Netzwerk sowie der Kommunikation zwischen Brokern. Der „Message Storage“ stellt die Persistenzschicht dar und wird entweder durch die interne KahaDB[2] oder einer

[1] http://activemq.apache.org/
[2] http://activemq.apache.org/kahadb.html

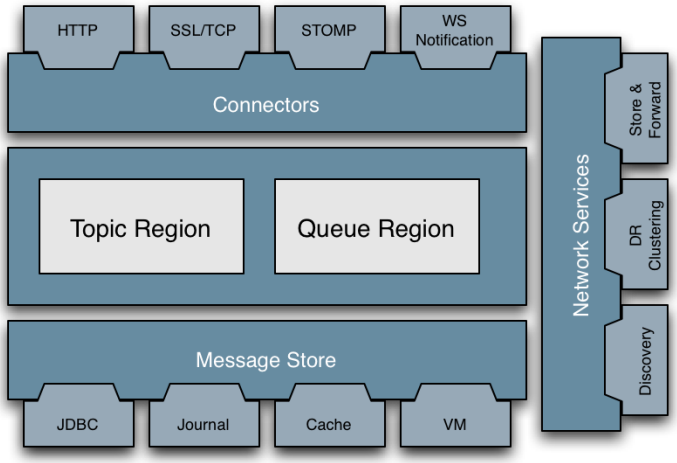


Abbildung 6.1: Architektur des ActiveMQ (The Apache Software Foundation (2011))

JDBC-Datenbank realisiert. Diese Ansätze zur Persistierung dienen jedoch nicht als reine Nachrichtenspeicher. Nach den bisherigen Erkenntnissen liegt die Aufgabe dieser Datenbank darin, nach einem Ausfall des Systems sämtliche Kommunikationskanäle und nicht ausgelieferte Nachrichten wiederherzustellen. Aus diesem Grund ist der Einsatz einer externen Datenbank weiterhin erforderlich.
Die zentrale Einheit der Abbildung 6.1 bilden die „Regions“, welche die beiden Kommunikationsmechanismen des ActiveMQ beschreiben. Diese unterteilen sich zum einen in *Topics* und zum anderen in *Queues*.

1. **Queues**
   Das Funktionsprinzip einer Queue ähnelt der gleichnamigen Datenstruktur dahingehend, dass *Producer* ihre Daten in eine Warteschlange schreiben und diese von *Consumern* aufgenommen werden. Der Grundgedanke der Queue liegt beim ActiveMQ in einer „n-zu-1“-Beziehung zwischen Producern und Consumern, was bedeutet, dass es zwar mehrere Erzeuger von Daten gibt, allerdings lediglich einen Konsumenten. Sollten sich dennoch mehrere Verbraucher an einer Queue anmelden, wird die Nachricht lediglich ein Mal ausgeliefert, wodurch sie nur ein Teilnehmer erhält.
   Sollte der beim ActiveMQ angemeldete Consumer bei der Erzeugung eines neuen Datums nicht erreichbar sein, werden die nicht ausgelieferten Informationen so lange gespeichert, bis der Konsument aufnahmebereit ist.

2. **Topics**
   Im Gegensatz zur Queue wird beim ActiveMQ in einem Topic von einer „n-zu-m“-Beziehung zwischen *Publishern* und *Subscribern* ausgegangen. Hier können sich folglich mehrere Interessenten an einem Topic anmelden und die Nachrichten verschiedener Publisher konsumieren. Dies bedeutet, dass jeder Subscriber jede erzeugte Nach-

richt, unabhängig von der Anzahl der Verbraucher auf dem Topic, erhält. In diesem Zusammenhang lassen sich die Konsumenten in *Durable Subscriber* und *Non-Durable Subscriber* einteilen. Erstere erhalten nachträglich sämtliche Nachrichten, die über das Topic versendet wurden, sollten sie zu einem Zeitpunkt inaktiv und somit nicht erreichbar sein. Letztere verlieren diejenigen Nachrichten, die innerhalb des Zeitintervalls ihrer Abwesenheit versendet wurden.

Der Unterschied zwischen dem Konzept einer Queue und einem Topic soll ergänzend in der Grafik 6.2 veranschaulicht werden.

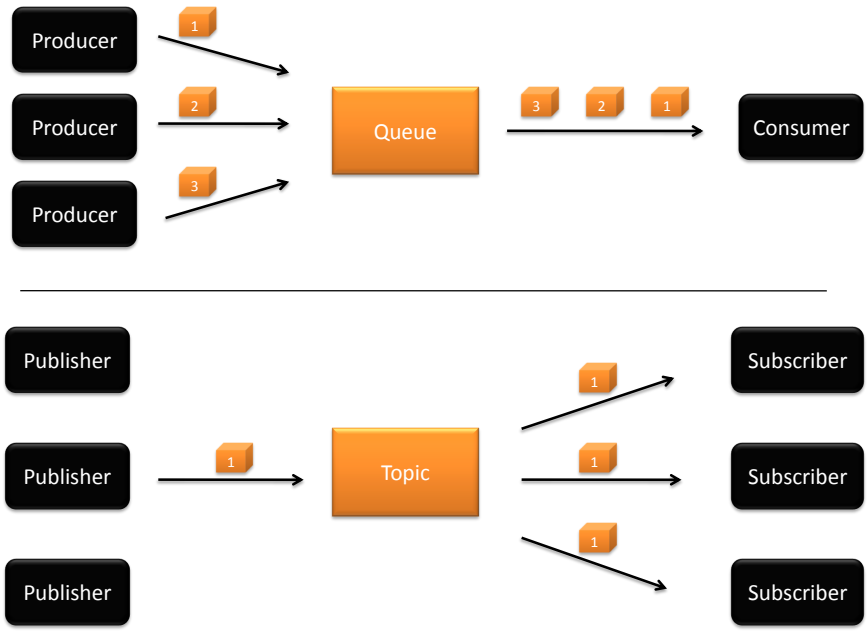


Abbildung 6.2: Grafische Darstellung der Teilnehmer eine Queue und einem Topic

Für weiterführende Informationen über den ActiveMQ Message-Broker sei an dieser Stelle auf das Buch „ActiveMQ in Action“ (Snyder u. a. (2011)) verwiesen.

6.1.1 ActiveMQ im Living Place Hamburg

Der ActiveMQ wurde auf einem Dell PowerEdge R610 Rack-Server installiert. Dieser Server verfügt über einen Intel Xeon L5630 Prozessor und 16GB Arbeitsspeicher. Zur Verwaltung verschiedener Betriebssysteme besitzt diese Einheit den *VMware ESX Server*[3], auf wel-

[3] http://www.vmware.com/products/vsphere/esxi-and-esx/index.html

Anzahl Topics	Publisher pro Topic	Subscriber pro Topic	Empfangene Nachrichten	Ausgelieferte Nachrichten	Datenverlust in Prozent	Programm Laufzeiten
1	1	2	1000	2000	0%	0,892 s
3	1	3	30000	90000	0%	4,289 s
5	1	3	100000	300000	0%	11,305 s
3	2	3	40000	120000	0%	6,031 s

Tabelle 6.1: Empfangene und ausgelieferte Nachrichten über Topics des ActiveMQ

chem ein *Windows 7 Enterprise* installiert wurde, um als Betriebssystem für den ActiveMQ zu fungieren.

Der ActiveMQ wurde im *Living Place Hamburg* in der Version 5.5.0 installiert und aufgrund der bisherigen Erfahrungen kann von einer erfolgreichen Integration in die Laborumgebung gesprochen werden. Der Message-Broker wurde sowohl zu Testzwecken als auch unter realen Bedingungen von verschiedenen Anwendungen, zur Kommunikation mit weiteren Netzteilnehmern, verwendet. Die Programmiersprachen, die in diesen Applikationen zum Einsatz kamen, waren u.a. Java, C# und Python. Damit eine weitestgehende Sprachunabhängigkeit bei dem Nachrichtenaustausch zwischen den Systemen herrscht, werden die Informationen in der *JavaScript Object Notation*[4] (JSON) über das Blackboard versendet.
Des Weiteren wurden Testanwendungen geschrieben, mit Hilfe derer die Zuverlässigkeit der Nachrichtenauslieferung des ActiveMQ getestet werden sollte.

Die Tabelle 6.1 beschreibt das Resultat einer Java-Anwendung, in der entsprechend der Anzahl an Publishern und Subscribern Threads erzeugt werden, welche daraufhin Nachrichten generieren bzw. konsumieren. Die empfangenen sowie ausgelieferten Daten beziehen sich in dieser Tabelle auf den ActiveMQ. Beispielsweise entsteht die zweite Zeile der Tabelle folgendermaßen: Es wurden drei Topics erstellt mit jeweils einem Publisher und drei Subscribern. Insgesamt hat der ActiveMQ 30000 Nachrichten erhalten, d.h. von jedem Publisher wurden 10000 Nachrichten versendet. Da auf jedem Topic drei Interessenten registriert sind, versendet der Message-Broker jede erhaltene Nachricht drei Mal. Der Datenverlust beträgt Null Prozent, was bedeutet, dass alle Nachrichten korrekt ausgeliefert wurden. Die letzte Spalte beschreibt, wie lange die Applikation für die Ausführung des Programms benötigt hat.

Die Erfahrungen mit dem Apache ActiveMQ zeigen, dass dieses System einen zuverlässigen Nachrichtenaustausch gewährleistet und eine hohe Skalierbarkeit aufweist, weshalb sich dieser Message-Broker ausgezeichnet für den langfristigen Einsatz im *Living Place Hamburg* eignet. Sollte sich bei der stetig wachsenden Anzahl an Komponenten dennoch

[4] http://json.org/

herausstellen, dass der Einsatz des ActiveMQ auf einem Server nicht ausreicht, lassen sich Netzwerke aus Brokern bilden, die verteilte Topics und Queues verwenden und die Nachrichten untereinander kommunizieren.

6.2 Persistenz

Aus den Anforderungen und dem Designkapitel geht hervor, dass für die Persistierung der versendeten Nachrichten eine dokumentenbasierte Datenbank gewählt werden sollte. Infolgedessen wurden mit der „MongoDB“ und der „CouchDB“ zwei bekannte Vertreter der NoSQL-Datenbanken untersucht und ihre Vor- und Nachteile gegeneinander abgewogen. Hierzu wurde den Datenbanken in Otto und Voskuhl (2011) ein Performance-Test unterzogen, in dem sich zeigte, dass MongoDB bei kleineren Dokumenten eine höhere Leistungsfähigkeit aufweist. Aus diesem Grund galt es diese Datenbank in die Architektur des *Living Place* zu installieren. Folglich soll in diesem Abschnitt die in dieser Arbeit verwendete MongoDB vorgestellt werden.

Das Ziel von MongoDB liegt darin, dem Benutzer eine dokumentenorientierte Datenbank zu liefern, die umfassende Abfragemöglichkeiten besitzt, leistungsstark ist und eine hohe Skalierbarkeit aufweist. Sie ist in der Programmiersprache C++ geschrieben und gehört zu den document-based stores. Die hier angelegten Datenbanken gliedern sich in einzelne Collections, in denen die Dokumente gespeichert werden. Erstere sind vergleichbar mit Tabellen aus relationalen Datenbanken, während die Dokumente analog zu den Tupeln in den Tabellen zu betrachten sind, vgl. Edlich u. a. (2010). Die jeweiligen Dokumente werden in der MongoDB im BSON-Format[5] gespeichert. BSON steht für „Binary JSON“ und bietet eine binärcodierte Serialisierung von JSON ähnlichen Dokumenten an. Dieses Format erlaubt das Speichern von bis zu 4MB binärer Daten in einem Dokument. Sollte diese Größe nicht ausreichen, bietet MongoDB mit GridFS einen Mechanismus, der die Daten in sog. *Chunks* aufteilt und über mehrere Dokumente verteilt (Plugge u. a. (2010)). Für die Abfrage von Datensätzen lässt sich zum einen der Map/Reduce Ansatz verwenden, falls eine verteilte Suche über eine große Menge von Daten erforderlich ist und zum anderen wird eine eigene Syntax für einfache Abfragen zur Verfügung gestellt. Eine Interaktion mit der von MongoDB ausgelieferten Shell präsentiert das Beispiel 6.1. Hier wird in der Datenbank „amqmessages“ der Collection „myCollection“ ein Dokument hinzugefügt und abgefragt. An dieser Stelle sei erwähnt, dass die verwendete Collection nicht deklariert werden muss bevor sie verwendet wird, sondern beim ersten Einfügen eines Dokumentes zur Laufzeit generiert wird.

Listing 6.1: Beispielhafte Interaktion mit der MongoDB Shell

```
0 MongoDB shell version: 1.8.2
```

[5] http://bsonspec.org/

```
connecting to: test
> use amqmessages
    switched to db amqmessages
> db.myCollection.insert( {"Hello" : "LivingPlaceHamburg" } )
> db.myCollection.count()
    1
> db.myCollection.find( {"Hello" : "LivingPlaceHamburg" } )
    { "_id" : ObjectId("4e9ee2ed7e1ff1591dd9d9e0"), "Hello" : "LivingPlaceHamburg" }
```

Für komplexere Interaktionen mit der Datenbank unterstützt MongoDB zudem den Zugriff über verschiedene Programmiersprachen, wie z. B. C#, Java und Python.

Neben der binären Dokumentenspeicherung versucht MongoDB die Leistungsfähigkeit der Datenbanken durch die Form der Aktualisierung von Dokumenten zu erhöhen. Eine große Anzahl weiterer Datenbanken verwendet das *multi-version concurrency control* (MVCC), welches verschiedene Versionen zur Bearbeitung der Daten bereitstellt. Hiermit wird sichergestellt, dass Datensätze nicht verändert werden, während eine andere Anwendung eine Transaktion ausführt. Allerdings erfordert es dieser Ansatz, dass mehrere Kopien der Daten existieren, was die Komplexität erhöht und die Performance mindert. MongoDB hingegen verwendet ein sog. *in-place Update*. Dieses beinhaltet, dass ein Dokument lediglich an derjenigen Stelle aktualisiert wird, an der Veränderungen stattgefunden haben. Somit muss kein neuer Speicher für eine Kopie des Objektes allokiert werden und die Indizes können ebenso unberührt bleiben. Des Weiteren arbeitet diese Datenbank mit *lazy writes*, was zur Folge hat, dass die Daten möglichst selten auf die Festplatte geschrieben werden.

6.2.1 MongoDB im Living Place Hamburg

Die MongoDB wurde im *Living Place Hamburg* in der Version 1.8.3 auf demselben Server wie der Apache ActiveMQ Message-Broker installiert (6.1.1). Da zur weiteren Verarbeitung sämtliche versendete Nachrichten in der Datenbank gespeichert werden sollen, ist in Otto und Voskuhl (2011) ein „Messaging Wrapper“ entwickelt worden. Dieser vereinfacht auf der einen Seite den Zugriff auf den ActiveMQ und nimmt dem Entwickler die Verantwortung ab, die Daten zu persistieren, indem die verschickten Daten automatisch in die MongoDB geschrieben werden.
Des Weiteren wurde ein Performance-Test auf einem „Apple Mac Pro“ mit einem Intel Xeon x5472 Prozessor mit 10GB RAM ausgeführt, in dem es gelungen ist 100.000 Dokumente in 1,86 Sekunden zu archivieren. Hier ist erkennbar, dass die MongoDB Datensätze in kürzerer Zeit speichert, als der ActiveMQ sie verteilt, weshalb die Performance der Datenbank für das hier vorgestellte *Smart Home* vollständig ausreicht. Bei der hohen Anzahl an generierten Sensordaten spielt die Größe der Datenbank auf der Festplatte ebenfalls eine gewichtige Rolle. Aus diesem Grund wurden einige Testapplikationen implementiert, die eine unterschiedliche Anzahl sowie verschiedenartige Dokumente in die MongoDB schreiben. Das Resultat dieses Tests wird in der Tabelle 6.2 veranschaulicht. Wie erwartet, hat

Anzahl Dokumente	Key/Value Paare	Speicherbedarf
1.000.000	4	ca. 142 MB
1.000.000	8	ca. 179 MB
2.000.000	8	ca. 348 MB
5.000.000	12	ca. 1032 MB

Tabelle 6.2: Speicherbedarf verschiedenartiger Dokumente in der MongoDB

die Menge der zu speichernden Dokumente sowie deren Größe Einfluss auf den benötigten Festplattenspeicher.
Außerdem zeigt das Ergebnis, dass während eines langfristigen Einsatzes der vorhandene Speicherplatz nicht ausreichen wird und der Entwickler einer Anwendung in Eigenverantwortung einschätzen muss, in welchen Fällen die Kontextinformationen im Netzwerk zu verschicken sind. Beispielsweise ist es nicht sinnvoll, die Raumtemperatur in Sekundenintervallen zu publizieren, da sich diese in nicht so kurzen Abständen ändert wie die Position einer Person, welche sich innerhalb weniger Sekunden signifikant von vorherigen Werten unterscheiden kann.
Darüber hinaus sollen die Datenbanken in festen Zeitabschnitten auf externen Medien gespeichert werden, sodass auf dem Server nicht bereits hinfälliger Speicher in Anspruch genommen wird. Zudem muss der dauerhafte Einsatz der Datenbank und die damit einhergehenden Erfahrungen im Umgang mit der Datenfülle zeigen, wie lange vergangene Sensordaten archiviert werden sollten.
Die MongoDB wurde im *Living Place Hamburg* im ersten Schritt auf einem einzelnen Server installiert. Sollte diese Variante in Zukunft nicht mehr ausreichen, bietet sich aufgrund der horizontalen Skalierbarkeit die Möglichkeit, wie bereits beim ActiveMQ, ohne erheblichen Aufwand ein Cluster mit mehreren Rechnern zu bilden, um die Leistungsfähigkeit zu optimieren.

6.3 Indoor Positioning

In Abschnitt 5.3.1 wurde bereits erwähnt, dass die Interpretation von Positionsdaten einen zentralen Aspekt in dieser Arbeit einnimmt. Hierzu soll zunächst erläutert werden, wie diese Informationen im *Living Place* gewonnen werden.

Der Bereich des Indoor Positioning stellt in der Informatik einen aktuellen Forschungsschwerpunkt dar, in welchem verschiedene Verfahren zum Einsatz kommen, mit der Absicht Personen und weitere Objekte innerhalb eines Gebäudes zu detektieren. In diesem Zusammenhang gehören u.a. „WLAN Fingerprinting“ (Honkavirta u. a. (2009)), „Floor Sensors“ (Val-

tonen und Vanhala (2009)) und „Ultra Wide Band“-Systeme (UWB) (Fontana u. a. (2003)) zu bekannten Mechanismen zur Positionsbestimmung. An der HAW Hamburg wird zu diesem Zweck ein System der Firma Ubisense[6] eingesetzt, welches auf der Basis von UWB-Signalen *Location Tags* lokalisiert.
Das hier eingesetzte System verfügt über ein Netzwerk von sechs Sensoren, an das die Tags ihre Position senden. Die Anordnung dieser fest installierten Sensoren, innerhalb des Wohnbereichs, zeigt die Abbildung 6.3 und wurde im Hinblick auf eine möglichst exakte Abdeckung aller Bereiche gewählt. Ein weiteres Ziel bestand darin, in einem Großteil des *Smart Homes* von mehr als einem Sensor erfasst zu werden, da die Ausbreitungsgeschwindigkeit von elektromagnetischen Wellen durch Wasser enthaltene Materialien, wie der zu lokalisierende Mensch, signifikant reduziert wird.

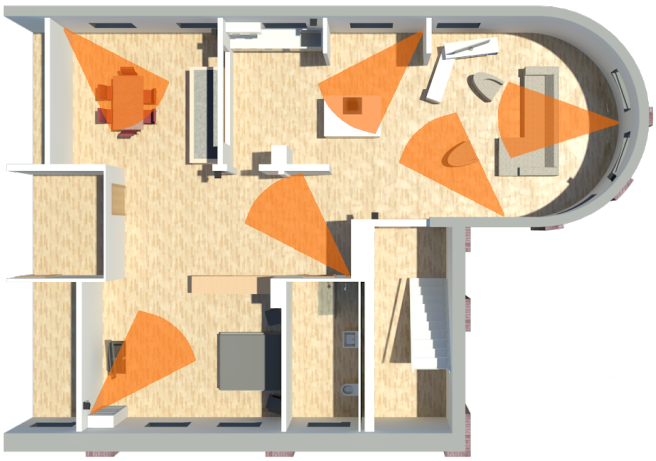


Abbildung 6.3: Anordnung der Ubisense-Sensoren im Living Place Hamburg

Im Folgenden soll die Positionsberechnung des Ubisense-Systems etwas genauer erklärt werden, vgl. Steggles und Gschwind (2005).
Die sog. *Ubisensors* kennen ihre feste Position innerhalb der Wohnung und kommunizieren via Standard Ethernet. Des Weiteren verfügen sie über einen „Radio Frequency Transceiver“ sowie mehrere UWB-Empfänger. Diese Sensoren werden in einer *Cell* zusammengefasst, welche den Bereich markiert, der vom System abgedeckt werden soll. Jede *Cell* benötigt einen Sensor der als *Master* fungiert und ein „Time Division Multiple Access“- Netzwerk (TDMA) über den RF-Kanal koordiniert, was bedeutet, dass in bestimmten Zeitintervallen die Daten verschiedener Sender auf einem Kanal übertragen werden können. Innerhalb des TDMA-Netzwerkes erhält jeder Tag innerhalb einer *Cell* einen angemessenen Zeitslot, in dem er aktiv ist und RF-Nachrichten mit seiner Identität zusammen mit einer UWB-Puls Sequenz sendet, anhand derer die *Ubisensors* die Position des Tags bestimmen (siehe Ab-

[6] http://www.ubisense.net/en/

bildung 6.4). Die Sensoren verwenden hierzu eine Kombination aus der „Time-Difference of Arrival“ (TDOA) und der „Angle of Arrival“ (AOA) Technik. Erstere misst den Laufzeitunterschied des Signals zwischen dem Master und den weiteren Sensoren. Hierzu sendet jeder Sensor seine empfangenen Positionsdaten an die zentrale Einheit, welche diese daraufhin verarbeitet. Das AOA Verfahren ermittelt den Einfallswinkel des ankommenden Signals. Da die Sensoren ihre Abstände zueinander kennen, kann anhand der erhaltenen Daten mittels Triangulation die Entfernung des Tags bestimmt werden.
Bezüglich der Zeitintervalle, in denen die einzelnen Tags ihre Positionsdaten senden können, sei an dieser Stelle erwähnt, dass sich diese individuell bestimmten lassen, sodass sich häufig bewegende Sender größere Abschnitte erhalten, als Tags, die sich selten bewegen. Zudem lassen sich die Zeitslots im laufenden Betrieb an die Anforderungen von Anwendungen anpassen.

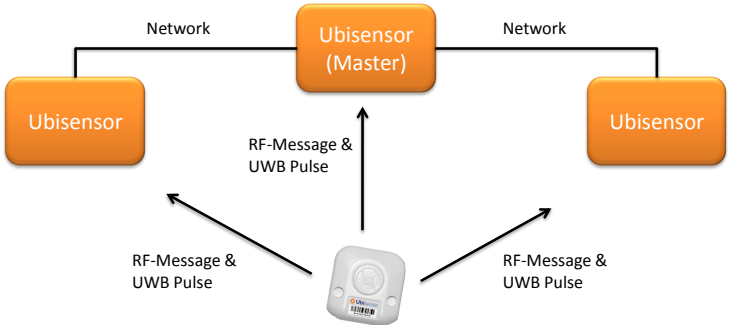


Abbildung 6.4: Beteiligte Ubisense-Komponenten bei der Positionsbestimmung

Im *Living Place Hamburg* hat sich das Ubisense bereits in früheren Szenarien als erfolgreiches Indoor-Positioning System bewährt. Die bisherigen Erfahrungen zeigen eine Genauigkeit von bis zu 30cm, was für die Aktivitätserkennung eine ausreichende Präzision darstellt. Wie bereits im Kapitel 4 diskutiert wurde, sind bei einer großen Menge von Sensordaten mit einer hohen Wahrscheinlichkeit Ausreißer unter diesen zu finden. Da diese Beobachtung gleichermaßen in der Aufzeichnung der Positionsdaten dieses Ortungssystems gemacht werden konnten, wird sich der Abschnitt 6.4.1 intensiv mit der Detektion von inkorrekten Werten in Lokationsinformationen befassen.

6.4 Kontexterkennung im Living Place Hamburg

Im Hinblick auf die Erkennung der Aktivitäten in den Morgen-Szenarien, soll in diesem Abschnitt beschrieben werden, inwiefern die einzelnen Kontextinformationen auf dem modellunabhängigen Level bezogen, verarbeitet und höheren Interpretationslevel zur Verfügung zu

stellen sind. Da die Verarbeitung der Positionsdaten des Ubisense eine bedeutende Rolle spielt, soll zunächst der Umgang mit dieser Form von Daten beleuchtet werden, bevor der weitere Verlauf dieses Abschnitts sich mit der Bereitstellung von logischen und abgeleiteten Sensordaten befasst.

6.4.1 Bewegungsprofile

Zur Erstellung von Bewegungsprofilen des Bewohners des *Living Place Hamburg* müssen die einzelnen Informationen im ersten Schritt in der Form zusammengefasst werden, dass die zurückgelegten Wege sowie die Aufenthaltsdauer in einem bestimmten Bereich innerhalb der Wohnung abgebildet werden.
Vor diesem Hintergrund gilt es Trajektorien aus den Daten zu erstellen, durch die sich der beschrittene Pfad erkennen lässt. Allerdings liegt bereits die erste Herausforderung darin, die Eigenschaften einer Trajektorie aus Positionsdaten zu definieren. Aus der modellunabhängigen Interpretationssicht wurde zunächst ein Ansatz in Betracht gezogen, in dem die Pfade entweder aus einer festgelegten Anzahl von Koordinaten oder einem strikten Zeitintervall bestehen. Hier wurde eine Applikation entwickelt, die 50 Positionsdaten zu einer Trajektorie zusammengefasst und abgespeichert hat. Diese Vorgehensweise hat sich für die ersten Tests und das anschließende Clustering (siehe 6.4.1) von Trajektorien als sinnvoll erwiesen, da sich Pfade mit identischer Anzahl an Positionsdaten problemlos vergleichen lassen. Für die Aktivitätserkennung ist diese Methodik jedoch nicht praktikabel, da sich beispielsweise aus dem Weg vom Schlafbereich in das Wohnzimmer mehrere Pfade ergeben könnten, obwohl es sich letztlich um einen einzelnen Weg handelt. Zudem wäre es denkbar, dass das Wohnzimmer erreicht wird, ohne dass die 50 Daten erreicht worden sind, wodurch sich keine Trajektorie ergeben würde, obgleich der Weg zwischen diesen Bereichen einen relevanten Gesichtspunkt für das Bewegungsprofil darstellt. Bei der zeitlichen Definition einer Trajektorie ergeben sich ähnliche Schwierigkeiten. Zwar würde hier eine Trajektorie zwischen Schlafbereich und Wohnzimmer erkannt werden, da die festgelegte Zeit zu einem gewissen Zeitpunkt erreicht wird, allerdings stellte sich die Bestimmung eines geeigneten Intervalls als äußerst schwierig dar. Sollte das Intervall zu groß gewählt werden, wäre es möglich, dass mehrere Pfade als ein einzelner erkannt wird. Bei einem zu kurzen Zeitslot ist es vorstellbar, dass ein einzelner Pfad nicht erkannt wird, sondern als eine Reihenfolge unterschiedlicher Wege.

Für die Realisierung des hier vorgestellten Systems musste folglich in einem weiteren Schritt die Definition einer Trajektorie überarbeitet werden. Zu diesem Zweck wurden „Regions of Interest“ (RoI) definiert, anhand derer der Beginn und das Ende eines Pfades bestimmt werden. Der Grundgedanke liegt hier darin, zu betrachten, in welchem Bereich sich die ersten Positionsdaten nach einer Ruhephase lokalisieren lassen und in welchem Areal die letzten Informationen auftreten, bevor sich eine gewählte Zeitspanne lang keine signifikanten Positionsänderungen ergeben. Ein Zeitintervall dieser Form zu wählen erwies sich dabei als

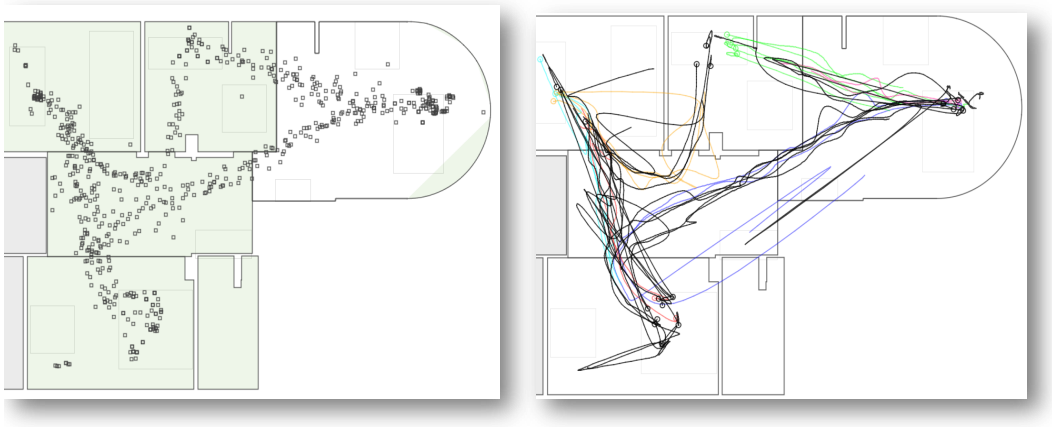
weitaus unkomplizierter als die Definition eines Zeitslots für eine gesamte Trajektorie. In dieser Arbeit wurde eine Dauer von 5 Sekunden gewählt, innerhalb der sich die Position nicht entscheidend verändern darf, um das Ende einer Trajektorie zu markieren. Der Bereich in dem Pfade häufig enden, wird daraufhin als „Region of Interest“ gekennzeichnet. Der Beginn einer Trajektorie zeichnet sich daher durch das Verlassen einer RoI aus. Da die Entwicklung der RoI's mit einer hohen Anzahl von Testdaten verbunden ist, die zu Beginn dieser Arbeit jedoch nicht vorlagen, wurden im ersten Schritt feste Regionen definiert zwischen denen sich die Pfade ergaben und somit die Verarbeitung der Positionsdaten fortgesetzt werden konnte. Auf dieser Basis wurde ein Satz von Trajektorien erstellt, welcher in einem nächsten Schritt mit einem Clustering Algorithmus analysiert wurde. Der folgende Abschnitt befasst sich entsprechend mit dem Ergebnis des Clusterings.

Clustering der Trajektorien

Im Kapitel der algorithmischen Verfahren wurde bereits eine Einführung in die Clusteranalyse präsentiert, siehe 4.2. Hier konnte das Verfahren „Density-Based Spatial Clustering of Applications with Noise“ (DBSCAN) mit der Eigenschaft bestechen, dass die Anzahl der Cluster nicht im Vorhinein bekannt sein muss. Hier ergeben sich die Cluster im Verlauf der Analyse, indem sog. Kernobjekte, dichte-erreichbare Objekte und Rauschpunkte identifiziert und zugeordnet werden.

Zur Anwendung des DBSCAN auf die Positionspfade muss einführend konkretisiert werden, inwiefern die Trajektorien miteinander vergleichbar sind. Bei einer festen Länge, wie sie in den ersten Tests vorlagen, lassen sich identische Aufzeichnungen über die Zeit untersuchen, was bedeutet, dass inkrementell die jeweils nächsten Koordinaten innerhalb von Pfaden verglichen werden. Jedoch haben sich, wie bereits beschrieben, in dieser Arbeit Wege variabler Länge ergeben, was einen Vergleich aller Punkte zweier Trajektorien ausschließt. Folglich konnten in einigen Fällen lediglich Teilstücke von gegangenen Wegen geclustert werden. Mit dem Ziel, eine Ähnlichkeit von sehr kurzen Wegen mit längeren zu unterbinden, wurde hier entschieden, dass zur Eigenschaft einer Trajektorie eine Mindestlänge von zehn Positionsdaten vorliegen muss. Sollte dieses Merkmal erfüllt sein, werden für den Vergleich zweier Wege die Punkte auf identischer Position (z. B. dritte gespeicherte Koordinate der jeweiligen Trajektorie) gegenübergestellt, bis die Länge des kürzeren Pfades erreicht ist. Falls sich die Wege bis zu dieser Situation entsprechend gleichen, werden sie in dieser Arbeit in einem Cluster zusammengefasst und gelten somit als ähnlich. Diese Methode konnte in dieser Anwendung gewählt werden, da hier davon ausgegangen wird, dass sich innerhalb des *Living Place* ausschließlich Pfade zwischen „Regions of Interests“ ergeben, da es als unwahrscheinlich gilt, dass der Bewohner sich längerfristig in einem Bereich eines Raumes aufhält, in dem er keine Tätigkeit ausüben kann. Sollten sich dennoch mehrere Wege in ein bisher unbekanntes Areal des Wohnbereichs ergeben, wird dieser Bereich als Konsequenz darauf als eine neue

RoI gekennzeichnet. Daher ergeben sich generell entweder stark voneinander abweichende Wege oder ähnlich lange. Für eine möglichst optimale Segmentierung von Trajektorien verschiedener Länge sei an dieser Stelle auf Lee u. a. (2007) verwiesen. Hier wird algorithmisch versucht Trajektorien an Stellen zu partitionieren, an denen sich das Verhalten der Trajektorie rapide ändert.
In einem weiteren Schritt wurden die Parameter ϵ und *minPoints* des DBSCAN-Algorithmus festgelegt. Erster Wert wurde dabei mit 1,5m definiert, während *minPoints* auf die Zahl 3 festgelegt wurde. Dies bedeutet, dass zur Ähnlichkeit von zwei Trajektorien kein Punkt auf identischer Position um mehr als 1,5m von seinem Gegenüber entfernt sein darf. Der Parameter *minPoints* beschreibt hingegen, dass ein Cluster aus mindestens drei Trajektorien bestehen muss.
Mit Hilfe dieser Parameter wurden daraufhin erste Tests mit Pfaden verschiedener Länge durchgeführt. Das Resultat dieses Tests präsentiert die Grafik 6.5 und zeigt die erstellten Trajektorien in einer Spline-Darstellung. Die Pfade einer Farbe stellen in diesem Fall ein



Positionsrohdaten Cluster nach Anwendung des DBSCAN

Abbildung 6.5: Anwendung des DBSCAN auf reale Positionsdaten

Cluster dar, während die schwarzen Trajektorien keiner Gruppe zugeordnet werden konnten und einen wesentlichen Teil des Ergebnisses der Clusteranalyse darstellen. Weiterhin zeigt die Abbildung, dass die einzelnen Wege häufig Informationen beinhalten, die kein gewöhnliches Bewegungsverhalten darstellen, da sehr große Sprünge zwischen den einzelnen Daten auftreten. Anhand der ersten Resultate konnte daher festgestellt werden, dass die Rohdaten zunächst von Ausreißern zu bereinigen sind, bevor die Clusteranalyse durchgeführt wird.

Positionsdaten von Ausreißern bereinigen

Zur Identifizierung von Ausreißern innerhalb der Trajektorien wurden verschiedene Filter implementiert und auf die Positionsdaten angewandt. Diese Vorgehensweise soll gewährleisten, dass keine fehlerbehafteten Werte in die Clusteranalyse einfließen und das Ergebnis dahingehend beeinflussen, dass einzelne Trajektorien keinem Cluster zugeordnet werden können, da vereinzelt starke Abweichungen zu anderen Pfaden bestehen, die in der Realität nicht existieren.
Die folgend vorgestellten Filtermechanismen wurden nachträglich auf die Datensätze angewandt, um eine Vergleichbarkeit zwischen den originalen und bereinigten Daten zu erhalten. Zu diesem Zweck wurde der in Abbildung 6.5 vorgestellte Datensatz drei in dieser Arbeit implementierten Filtern unterzogen.
Im ersten Schritt wurde ein Filter implementiert, der mit dem arithmetischen Mittel arbeitet. Dies beinhaltet, dass jeder Punkt innerhalb eines Pfades durch den Durchschnittswert, der sich aus ihm selbst und seinen Nachbarwerten ergibt, ersetzt wird. Zur Berechnung des Wertes wird hierzu über die jeweilige Trajektorie iteriert und falls vorhanden, die zwei zuvor sowie die zwei anschließend aufgezeichneten Werte einbezogen. Sollten nicht ausreichend vorherige oder nachfolgende Nachbarwerte zur Verfügung stehen, wird das Verhältnis dahingehend angepasst, dass zu jeder Zeit vier Nachbarn in die Berechnung inkludiert werden.
Die Anwendung dieses Filters konnte auf der einen Seite einige Ausreißer abschwächen und somit dafür sorgen, dass weitere Pfade in Cluster kategorisiert werden konnten. Auf der anderen Seite wurden die Extremwerte jedoch nicht vollständig entfernt, sodass Trajektorien mit stark abweichenden Werten weiterhin nicht in Cluster eingebunden wurden.

In Abschnitt 4.1 wurde erläutert, dass im Gegensatz zum arithmetischen Mittel der Median nicht von Extremwerten beeinflusst wird. Diese Eigenschaft sowie die unzureichenden Resultate des oben beschriebenen Filters, haben dazu geführt, dass zur Eliminierung der Ausreißer ein Median-Filter implementiert wurde. Hier wird zunächst der erste und der letzte Punkt der betrachteten Trajektorie gespeichert, indem diese unberührt der neu entstehenden Trajektorie hinzugefügt werden. Daraufhin wird jeweils ein Fenster von fünf aufeinander folgenden Werten betrachtet, von denen der Median in den neuen Pfad aufgenommen wird. Die Anwendung dieses Filters hat demnach hat zur Folge, dass sowohl die Ausreißer aus den aufgezeichneten Daten entfernt werden als auch eine Vielzahl korrekter Koordinaten verloren geht. Aus diesem Grund ist bei der Anwendung dieses Filters zu beachten, dass die neu entstandenen Trajektorien vermutlich in einigen Bereichen vom real gegangenen Weg abweichen.

Aufgrund des hohen Verlustes an Positionsdaten beim Median-Filter, wurde in diesem System eine dritte Methode entwickelt, welche zwar die Ausreißer aus dem Datensatz entfernt, allerdings möglichst sämtliche fehlerfreien Daten erhält. Vor diesem Hintergrund ist ein Radius-Filter implementiert worden, mit Hilfe dessen sich jede einzelne Position im Hin-

blick auf ihre Korrektheit untersuchen lässt. Diesbezüglich wird eine Trajektorie inkrementell bearbeitet, indem geprüft wird, ob das folgende Positionsdatum im Radius von 1,5m des aktuell betrachteten oder den zwei vorhergegangenen Werten liegt. Sollte dieses nicht der Fall sein, wird davon ausgegangen, dass es sich bei der folgenden Koordinate um eine Fehlmessung handelt. Da das Indoor-Positioning System ca. jede 0,5 Sekunden ein Positionsupdate versendet, müsste der Bewohner in diesem Zeitintervall mehr als 1,5m zurücklegen, was bei einer normalen Bewegungsgeschwindigkeit als unwahrscheinlich gilt. Deshalb werden Werte, die nicht im Radius einer zuvor aufgezeichneten Koordinate liegen, verworfen. Nach der Eliminierung dieses Wertes ist es einerseits denkbar, dass der folgend betrachtete Punkt im Radius des aktuellen Datums liegt und die Trajektorie auf der Basis dieses Punktes weiter untersucht wird. Andererseits besteht die Möglichkeit, dass die nächste Koordinate wiederum nicht im entsprechenden Radius liegt. In diesem Fall kann nicht entschieden werden, ob es sich um einen weiteren Ausreißer handelt oder er allein wegen des verworfenen Wertes nicht im erforderlichen Radius liegt. Zur Lösung dieses Problems wird hier nach drei verworfenen Werten das folgende Datum als korrekt betrachtet, mit der Absicht anhand dieses Wertes die weitere Trajektorie zu analysieren.
In dieser Implementierung besitzen die drei Werte, mit denen die neue Koordinate verglichen wird, eine identische Gewichtung. Für zukünftige Realisierungen wäre es dagegen denkbar, Gewichtungen im Sinne eines Gauß-Filters einzusetzen, sodass das zuletzt aufgezeichnete Datum eine höhere Gewichtung, als weiter zurückliegende Koordinaten, besitzt. Mit dieser Vorgehensweise ließe sich eine Glaubwürdigkeit der Werte erzeugen (siehe 3.2.4), indem Koordinaten die im Radius der zuletzt beobachteten Position liegen, eine hohe und Daten die sich im Radius weiterer vorhergegangener Daten befinden, eine geringere Glaubwürdigkeit erhalten.
Die positiven Erfahrungen mit dem Radius-Filter haben dazu geführt, dass neben der Möglichkeit des nachträglichen Bereinigen eine Applikation implementiert wurde, mit Hilfe derer jede neu generierte Koordinate vor einer Speicherung zuvor mit der vorherigen verglichen wird. Dieses hat zur Folge, dass ein Versenden lediglich dann über das Blackboard stattfindet, falls sie im vordefinierten Radius liegt und andernfalls nicht von anderen Anwendungen wahrgenommen wird.

Die Abbildung 6.6 zeigt die Ergebnisse der Clusteranalyse mit dem DBSCAN-Algorithmus, nachdem der im *Living Place Hamburg* aufgezeichnete Datensatz unter Anwendung der jeweiligen Filter bereinigt wurde. Hier fällt auf, dass nach der Kombination zwischen Radius und anschließendem Median-Filter, bis auf eine Trajektorie, sämtliche Pfade einem Cluster zugeordnet werden konnten. Das Zusammenspiel dieser beiden Methoden eignet sich folglich insbesondere für den Fall, dass möglichst jede Trajektorie zu kategorisieren ist. Der Grund für die hohe Anzahl eingeteilter Trajektorien liegt darin, dass die Pfade durch die Abfolge von Filtern kontinuierlich von der Realität abstrahieren. Dies ist u.a. daran zu erkennen, dass bei der hier gewählten Kombination einige Wege durch Wände führen, was physikalisch ausgeschlossen ist. Im Gegensatz hierzu bilden die Trajektorien nach alleiniger Anwendung

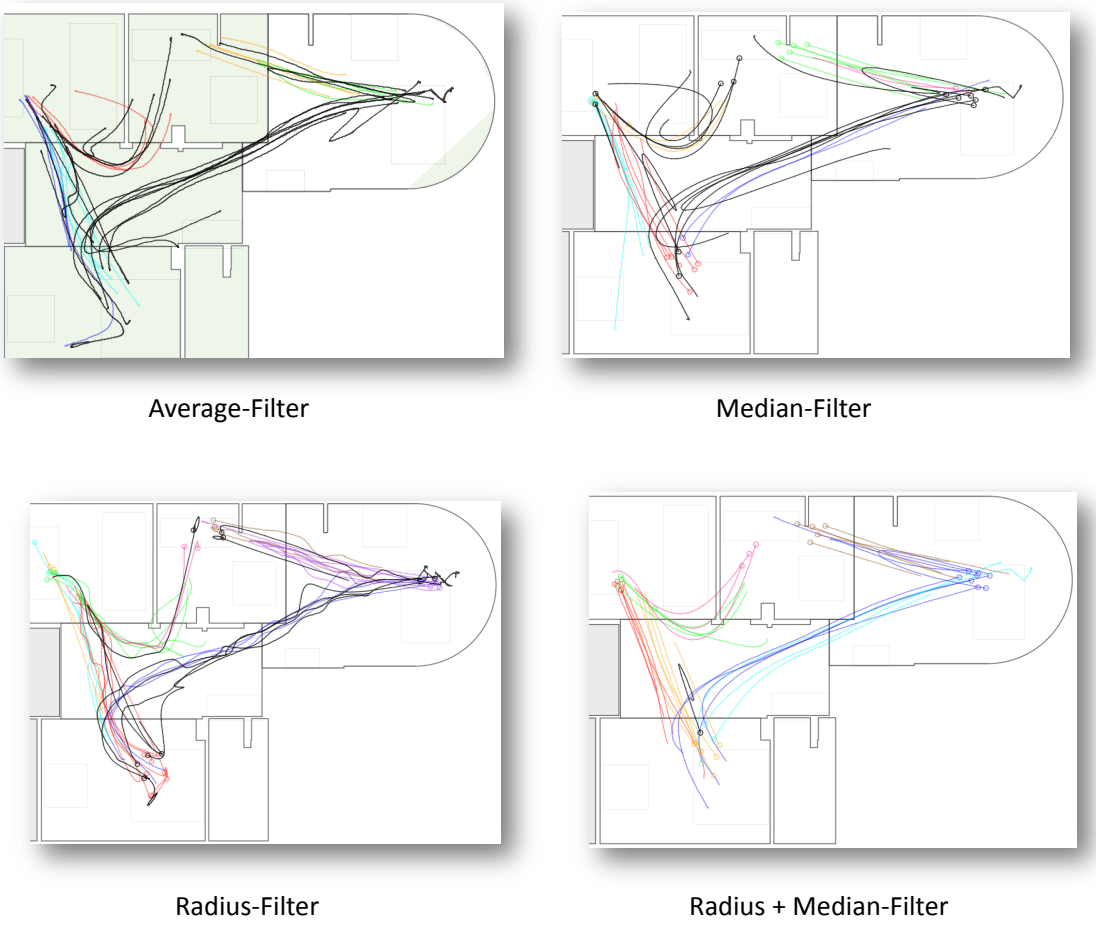


Abbildung 6.6: Resultate der Clusteranalyse nach Anwendung der jeweiligen Filter

des Radius-Filters weiterhin einen realistischen Weg ab. Jedoch werden hier in der Regel weitaus weniger Trajektorien geclustert.
Da es für den *Living Place Hamburg* im ersten Schritt von wesentlicher Bedeutung ist, möglichst jede Trajektorie zu kategorisieren, um Bewegungsprofile des Bewohners und Schätzungen über zukünftige Wege durchzuführen, sollen in diesem System die aus der Kombination der Filterung stammende Cluster an das höher gelegene Level weitergeleitet werden. An dieser Stelle sollen die Kategorien weiter mit semantischem Wissen angereichert werden, indem zunächst die in einem Cluster gebündelten Trajektorien übertragen werden. Inwiefern die Cluster auf weiteren Level verarbeitet werden und zur Realisierung der Morgen-Szenarien beitragen, wird in Karstaedt (2012) und Ellenberg (2011) detailliert beschrieben.
Nachdem die modellunabhängige Interpretation der Positionsdaten erfolgt ist, soll im fol-

genden Abschnitt die Bereitstellung von logischen und abgeleiteten Kontextinformationen im Vordergrund stehen.

6.4.2 Logische und abgeleitete Kontextinformationen

Zur Bereitstellung der logischen und abgeleiteten Kontextinformationen wurde in dieser Arbeit eine Applikation implementiert, mit Hilfe derer zum einen real herrschende Bedingungen als Kontext an die im *Living Place* vertretenen Anwendungen versendet werden, zum anderen jedoch ebenso die Möglichkeit besteht zu Testzwecken verschiedene Zustände zu simulieren. Die Abbildung 6.7 zeigt den zu diesem Zweck implementierten Kontextsimula-

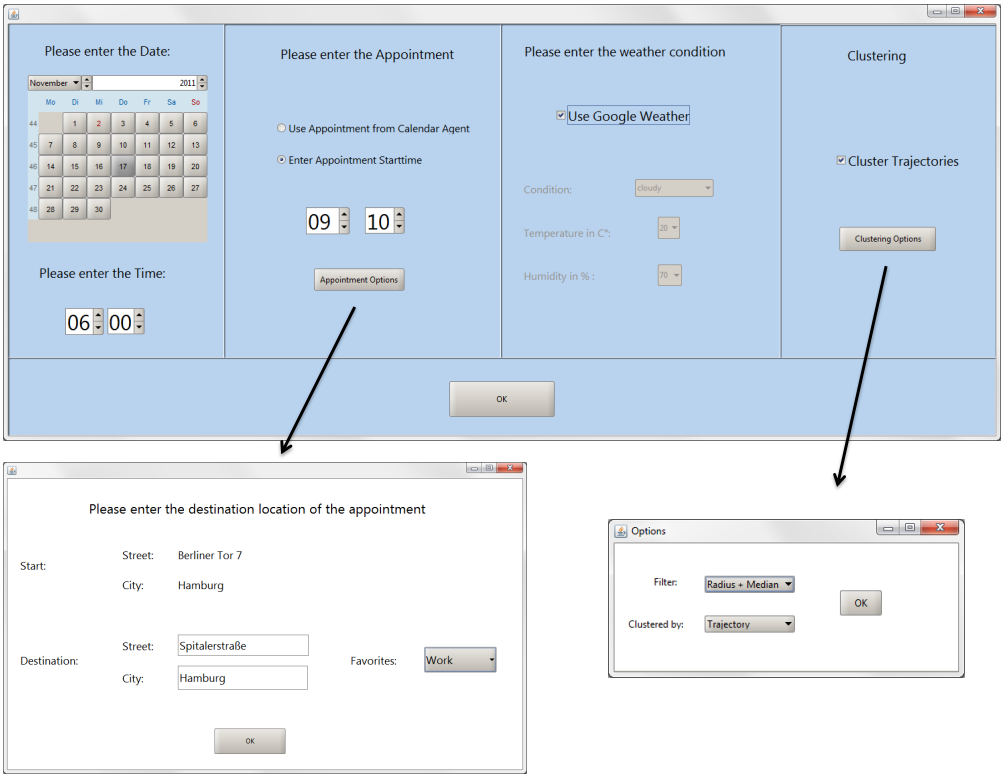


Abbildung 6.7: Darstellung des entwickelten Kontextsimulators

tor, in dem sich einige Einflüsse des Kontextes manipulieren lassen. Hier bietet sich dem Anwender u.a. die Möglichkeit, das gewünschte Datum sowie die zu verwendende Uhrzeit einzustellen. Des Weiteren lässt sich der erste Termin des Bewohners am festgelegten Tag definieren, wozu sowohl eine Startzeit als auch der Ort eingetragen werden muss. Neben diesem anwenderseitigen Festlegen von Terminen kann an dieser Stelle darüber hinaus auf den persönlichen Kalender des Bewohners zugegriffen werden. Dieser wurde mittels der

„Zimbra Collaboration Suite Open Source Edition“[7] realisiert und verwaltet sämtliche Termine, dadurch dass mehrere Personen Zugriff auf den Zimbra-Webclient besitzen, in dem Einträge hinzugefügt oder modifiziert werden können.
Ferner kann der Anwender des Kontextsimulators darüber entscheiden, ob er die Wetterbedingungen in Eigenverantwortung vorgeben möchte. In diesem Fall bieten sich die Optionen die allgemeine Wetterlage (beispielsweise bewölkt, sonnig, regnerisch), die Temperatur und die Luftfeuchtigkeit zu bestimmen. Sollte der Benutzer diese Einstellungen nicht vornehmen, wird über die „Google Weather API“[8] das aktuelle Wetter für die Stadt Hamburg erfragt.
Abschließend kann der Anwender entscheiden, ob zu den bisher gewählten Bedingungen passende Trajektorien ermittelt und Cluster gebildet werden sollen. Dies ist insbesondere der Fall, falls entweder neue Datensätze von Koordinaten aufgezeichnet wurden, es die häufig auftretenden Wege in den vorgebenden Situationen zu betrachten gilt oder auf der Basis der Einstellungen eine Aktivitätserkennung durchzuführen ist. Unter dem Menüpunkt „Clustering Options“ kann einerseits entschieden werden, welcher Filter für die Analyse zu verwenden ist und nach welchen Kriterien die Kategorisierung stattfindet. Vor diesem Hintergrund wurden zwei Methoden implementiert, anhand derer die Pfade geclustert werden. Ersteres Verfahren vergleicht, wie im Abschnitt 6.4.1 beschrieben, den räumlichen Abstand der Trajektorien zueinander. Eine weitere Methodik die in dieser Arbeit realisiert wurde, beinhaltet, dass Trajektorien die sich in einer ähnlichen Geschwindigkeit ergeben haben zusammengefasst werden. Zu diesem Zweck wird der Geschwindigkeitsvektor zu jeder Trajektorie berechnet. Da diese Information nur in seltenen Fällen eine wesentliche Rolle spielt, werden in diesem Fall die Kriterien miteinander kombiniert, sodass Trajektorien nur dann als ähnlich gelten, sollten sie sich räumlich und in ihrer Geschwindigkeit gleichen.

Nachdem die Einstellungen über den OK-Button bestätigt wurden, beginnt der Simulator diese in mehreren Schritten zu verarbeiten. Aufgrund des Datums und der Uhrzeit wird zunächst berechnet, ob der Sonnenaufgang in der Stadt Hamburg zu diesem Zeitpunkt bereits stattgefunden hat. Diese Berechnung wurde auf der Basis der in Barmettler (2011) vorgestellten Formel durchgeführt, die unter Berücksichtigung der geographischen Lage u.a. die Aufgangszeit der Sonne berechnen kann. Diese Information hat in den Morgen-Szenarien in der Form Einfluss auf die Reaktion des Gesamtsystems, dass nach dem Aufstehen nicht automatisch das Licht in dem Bereich eingeschaltet wird den der Bewohner betritt, falls der Sonnenaufgang länger als 30 Minuten zurückliegt.
Gesetzt den Fall, dass der erste Termin des Tages auf dem persönlichen Kalender zu erfragen ist, werden im nächsten Schritt, mittels einer in Barnkow (2010) erstellten API, die benötigten Informationen vom Zimbra Server bezogen. Mit dem Ziel die korrekte Weckzeit zu ermitteln, stellt die hier vorgestellte Anwendung an dieser Stelle eine Anfrage an den Internetdienst „Google Maps“[9], um die theoretisch benötigte Fahrzeit mit dem Auto von der

[7] http://www.zimbra.com
[8] http://www.google.de/ig/api?weather=Hamburg
[9] http://maps.google.com/

Wohnung zur entfernten Adresse in Erfahrung zu bringen. In diesem Zusammenhang wird eine XML-Datei verwendet, in der jede Antwort des Services gespeichert wird, damit bei wiederkehrenden Adressen keine erneute Anfrage über das Internet gestellt werden muss, sondern aus dem Speicher gelesen werden kann.
Ein weiterer Faktor mit Auswirkungen auf die Weckzeit ist das Wetter. Im Abschnitt 5.5 wurde bereits erläutert, dass die Weckzeit entsprechend früher sein sollte, sofern die äußeren Bedingungen eine längere Anreisezeit vermuten lassen. Daher wird für die Fahrtzeit zu einer Adresse bei Regen 15 Minuten und bei Schnee 30 Minuten mehr kalkuliert. Die aus diesen Aspekten generierte Zeit kann daraufhin mit der bisher eingestellten Alarmzeit verglichen und gegebenenfalls angepasst werden. Zur Realisierung dieses Vergleichs, muss die Anwendung des Weckers lediglich das betreffende Topic des ActiveMQ abonnieren und kann dementsprechend die errechnete Zeit umgehend einbeziehen.
Das Clustering der Trajektorien bezieht sich auf den Wochentag und die herrschende Tageszeit. Somit lassen sich, wie in den Szenarien gefordert, die Pfade eines Werktages von denen am Wochenende unterscheiden. Ebenso kann beobachtet werden, ob sich die aufgezeichneten Wege in der Nacht von denen am Vor- oder Nachmittag unterscheiden. Die Grafik 6.8 zeigt eine beispielhafte Ausgabe des Kontextsimulators sowie den Aufbau der von ihm über den ActiveMQ versendeten Nachricht. Die Cluster werden dabei in einer gesonderten Nachricht übertragen, da diese Information in der Regel nicht von jeder Teilkomponente verarbeitet werden muss.

6.4.3 Aktivitätserkennung anhand des implementierten Systems

Im Hinblick auf die Realisierung der in Abschnitt 3.1 vorgestellten Szenarien, sollen die auf diesem modellunabhängigen Level erzeugten Kontextinformationen im weiteren Verlauf dazu dienen, die Aktivitäten des Bewohners zu detektieren. Aus diesem Grund sollen die bereitgestellten Daten im Folgenden dahingehend untersucht werden, inwiefern sie in die Aktivitätserkennung in den Morgen-Szenarien eingebunden werden können. Zudem soll die Interaktion mit dem „Intermediate Level“ und der „High-Level“ Interpretation bei einzelnen Aktivitäten erläutert werden sowie auf weitere Arbeiten im *Living Place Hamburg* verwiesen werden, mit Hilfe derer sich die Tätigkeiten der Szenarien erkennen lassen. Vor diesem Hintergrund werden nun die Aktivitäten des Szenarios „Entspannter Morgen + Arbeitstag“ bezüglich ihrer Erkennbarkeit untersucht.

- **Aufstehen:** Dieses Ereignis wird durch das intelligente Bett (Hardenack (2011)) erkannt, indem die kapazitiven Sensoren keine Bewegungen mehr detektieren. Außerdem kann in einer Zusammenarbeit zwischen dem „Low Level“ und dem „Intermediate Level“ erkannt werden, dass sich eine Trajektorie vom Bett angefangen in eine bestimmte Richtung ergibt.

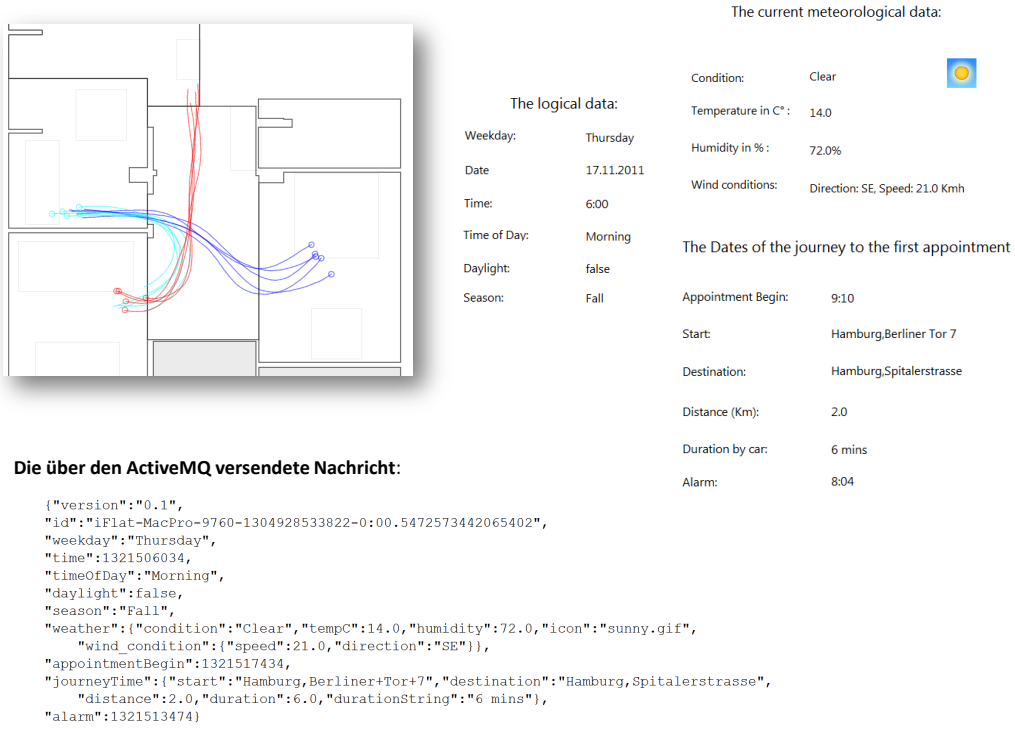


Abbildung 6.8: Ausgabe des Kontextsimulators sowie die versendete Nachricht

- **BadezimmerBetreten:** Da im Badezimmer des *Living Place* kein Ubisense-Sensor installiert ist, kann in diesem Bereich keine Positionsbestimmung ausgeführt werden. Jedoch wird auf unterster Ebene bemerkt, dass sich keine neuen Positionsdaten ergeben, weshalb die Trajektorie an das höher liegende Level weitergeleitet wird. Anhand des 3D-Gebäudemodells lässt sich nun erkennen, dass die Trajektorie unmittelbar vor dem Bad endet, was darauf schließen lässt, dass sich der Bewohner dort befindet. Das Verlassen dieses Raumes wird infolgedessen mittels Trajektorien mit dem Beginn vor dem Badezimmer entdeckt.
  Für das Detektieren der Aktivitäten *ZähnePutzen* und *DuschbadNehmen* gibt es zum aktuellen Zeitpunkt keine Implementation. Zukünftig ist hierzu allerdings der Einsatz von Durchflusssensoren, die in der Lage sind, Flüssigkeit zu detektieren, vorgesehen.
- **Ankleiden:** In dieser Arbeit wird davon ausgegangen, dass das Ankleiden im Schlafbereich der Wohnung stattfindet. Hierzu können die beiden unteren Interpretationslevel detektieren, dass die Person zum Kleiderschrank gegangen ist, während die Kameras weiterhin Bewegungen in diesem Bereich feststellen (Teske (2010)) ohne dass sich neue Positionsinformationen ergeben. Die Rule-Based Engine der „High Level“

Interpretation kann daraufhin die Informationen der beiden Context Provider kombinieren und daraus schlussfolgern, dass es sich um die Aktivität *Ankleiden* handelt.

- **FrühstückZubereiten:** Das ausgiebige Frühstück in diesem Szenario sieht vor, dass der Bewohner Kaffee kocht, den Tisch im Essbereich deckt und die Mahlzeit sitzend auf einem Stuhl zu sich nimmt. Dabei kann die Aktivität *KaffeeKochen* identifiziert werden, indem erkannt wird, dass Strom aus der Steckdose, an der die Kaffeemaschine angeschlossen ist, fließt. Ein gedeckter Esstisch kann zudem unter Anwendung von Object-Tracking (z. B. Teller, Butterdose) nachgewiesen werden (Najem (2011)). Der Ort, an dem der Bewohner sein Frühstück zu sich nimmt, kann erneut mit Hilfe einer Positionsbestimmung der beiden unteren Level stattfinden. Unter Zuhilfenahme dieser Reihe von Kontextinformationen kann das ontologiebasierte Interpretationslevel feststellen, dass ein ausgiebiges Frühstück stattgefunden hat. Die Tätigkeit *NachrichtenLesen* kann in diesem System entdeckt werden, indem elektronische Geräte ihre Benutzung über das Blackboard bekannt geben oder eine gedruckte Zeitung per Object-Tracking erkannt wird.

Neben diesen identifizierten Aktivitäten kann das „Intermediate Level“ beim Auftreten neuer Positionsdaten jederzeit die aktuellen Cluster des „Low Level“ anfordern, mit der Absicht die Zielposition des Bewohners zu bestimmen und mittels Nachrichten über das Blackboard vorbereitende Maßnahmen, wie beispielsweise das Einschalten des Lichts, zu initiieren.
In der obigen Beschreibung der Aktivitäten wird mehrfach auf das Zusammenspiel der drei Interpretationslevel hingewiesen, weshalb die Abbildung 6.9 einen beispielhaften Prozess präsentiert, in dem die Trajektorien Schritt für Schritt mit semantischem Wissen angereichert werden.

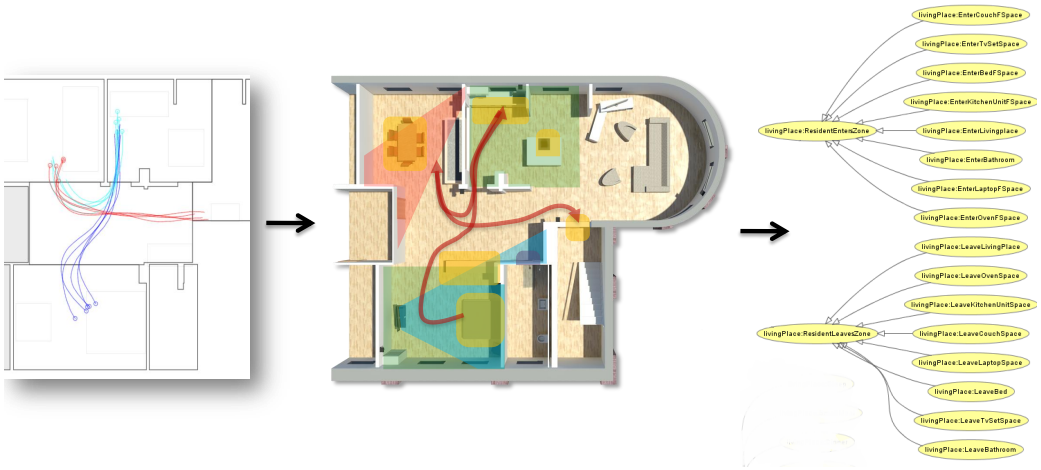


Abbildung 6.9: Beispielhafte Kommunikation der drei Interpretationslevel

Hier werden zunächst auf dem modellunabhängigen Level Cluster dadurch gebildet, dass die Rohdaten des Ubisense-Systems interpretiert werden. Anhand dieser Kategorien kann das „Intermediate Level“ unter Zuhilfenahme eines semantischem dreidimensionalen Gebäudemodells erkennen, innerhalb welcher *Functional Spaces* sich der Bewohner häufig befindet und zwischen welchen Bereichen des Wohnraums er sich fortbewegt. Die „High Level“ Interpretation kann auf dieser Basis in einem weiteren Schritt ermitteln, welche Aktivitäten sich während der erkannten Pfade ergeben haben, indem die mit der Ontologie verbundene Rule-Based Engine befragt wird.

6.5 Auswertung

Im hier präsentierten System ist es gelungen, eine Vielzahl von den in der Analyse sowie im Design erfassten Anforderungen umzusetzen. Mit der Einführung des Messagebrokers ActiveMQ in das *Living Place Hamburg* konnte das Kernstück des Blackboard-Architekturmusters verwirklicht und evaluiert werden. Zudem wurde mit der MongoDB eine Persistenzschicht realisiert, in der sämtliche über das Blackboard versendete Nachrichten gespeichert werden, ohne dass der Entwickler hierfür Sorge tragen muss, falls er den für das Projekt bereitgestellten ActiveMQ-Wrapper verwendet.
Neben den diversen Performance-Tests die den beiden Komponenten einzeln unterzogen wurden, wurde das Zusammenspiel der MongoDB und des ActiveMQ bei der offiziellen Eröffnung des *Living Place Hamburg* im November 2011 unter realen Bedingungen getestet. Hier wurden innerhalb von zwei Tagen über 760.000 Kontextnachrichten verschiedener Applikationen ausgetauscht. Hierunter zählten u.a. die Koordinaten verschiedener Tags des Ubisense-Systems, Steuerungsbefehle an die Lichtinstallationen sowie Anfragen an das semantische dreidimensionale Gebäudemodell. Dabei kam es während des gesamten Tests weder zu nennenswerten Engpässen in der Leistungsfähigkeit des Systems, noch konnte ein Nachrichtenverlust im Netzwerk festgestellt werden. Folglich wurde gezeigt, dass sich diese beiden Komponenten für den langfristigen Einsatz in diesem Projekt eignen und weiterhin einen elementaren Teil der Infrastruktur bilden werden.
Durch die stetig wachsende Zahl an Teilsystemen im *Living Place Hamburg* wird die Menge der Kontextinformationen und damit einhergehend, die Anzahl versendeter Nachrichten kontinuierlich steigen. Infolgedessen wurde in diesem Kapitel die Möglichkeit aufgezeigt, dass sowohl der ActiveMQ als auch die MongoDB skalierbar sind, dadurch dass ihre Funktion auf mehrere Recheneinheiten verteilt wird.
Nach der Einführung dieser Komponenten wurden im weiteren Verlauf die modellbasierte Kontextinterpretation und das Bereitstellen weiterer Kontextinformationen fokussiert. Zu diesem Zweck konnte ein Clustering-Verfahren von Trajektorien umgesetzt werden, anhand derer sich u.a. Bewegungsprofile des Bewohners erstellen lassen. Die Interpretation der Daten erfolgt dabei modellunabhängig, was den wesentlichen Vorteil beinhaltet, dass es zum

einen nicht erforderlich ist ein aufwändiges Modell der Umgebung zu erstellen und sich zum anderen die implementierten Algorithmen ohne Anpassungen in weitere Wohnumgebungen einbetten lassen.
Durch die Kombination mit weiteren in diesem Kontext realisierten Arbeiten, konnte in diesem Kapitel außerdem gezeigt werden, dass auf der Basis des hier vorgestellten Systems nahezu alle Aktivitäten der Morgen-Szenarien erkannt und somit kontextsensitive Reaktionen der Wohnumgebung initiiert werden können.
Darüber hinaus konnte in dieser Arbeit ein sogenannter Kontextsimulator implementiert werden, der für den Aufbau einer Testumgebung für die Komponenten in der Wohnung, neben den real herrschenden Bedingungen, verschiedene Zustände simulieren kann. Dieser Simulator ist daher in der Lage, eine Art Austausch der Kommunikationsplattform vorzunehmen, indem er es unterbindet, dass reale Daten in das Netzwerk versendet werden und im Gegenzug die definierten Werte kommuniziert. Hierzu ist es jedoch erforderlich, dass die jeweiligen Applikationen sowohl auf das Topic der realen Informationen als auch auf das simulierter Daten lauschen. Darüber hinaus müssen sie dazu in der Lage sein, als Reaktion auf eine globale Nachricht ihr Verhalten den Nachrichten eines Topics anzupassen, mit dem Ziel, lediglich aktuell relevante Daten in die Kontextinterpretation einzubeziehen. Zukünftig gilt es ferner zu untersuchen, inwiefern die Bereitstellung eines Blackboards, welches ausschließlich zu Testzwecken verwendet wird, sinnvoll ist. Zusammenfassend würden dementsprechend drei grundsätzliche Kommunikationskanäle resultieren, über welche jede Komponente mit weiteren Teilnehmern Nachrichten austauschen kann. Ersterer stellt dabei ein sogenanntes *Produktions-Topic* bereit, über welche die realen Sensordaten im Netzwerk verteilt werden, während das *Simulations-Topic* ausschließlich simulierte Werte an die Anwendungen versendet. Der dritte Kanal soll durch eine eigenständige ActiveMQ Instanz realisiert werden, worüber neu integrierte Komponenten Nachrichten austauschen können, ohne den regulären Informationsaustausch zu beeinflussen.

Des Weiteren gilt es Schritt für Schritt zu analysieren, ob die Daten des Kontextsimulators für die gewünschten Änderungen des Gesamtkontextes ausreichen. In diesem Zusammenhang müssen die Erfahrungen anderer Projekte im *Living Place Hamburg* zeigen, welche Informationsflüsse es zu simulieren gilt, damit notwendige Testumgebungen ohne großen Aufwand erstellt und verwendet werden können. Dabei zeigen weitere Projekte wie ein Ubisense-Simulator (Karstaedt (2011)), mit Hilfe dessen sich fiktive Positionsdaten erzeugen lassen, dass anhand simulierter Daten realitätsnahe Tests durchführbar und die Resultate zu einem wesentlichen Teil auf die Realität übertragbar sind.
Darüber hinaus wird aktuell ein „Skriptsimulator“ für den *Living Place Hamburg* entwickelt (Basener (2011)). Während der hier vorgestellte Kontextsimulator mit der Absicht verwendet wird, bestimmte Rahmenbedingungen zu erstellen, dient der Skriptsimulator dazu Aktivitäten und Tagesabläufe zu simulieren. Vor diesem Hintergrund lassen sich hier Skripte definieren, die entweder in einer realen, beschleunigten oder verlangsamten Zeit ausgeführt werden und die entsprechenden Sensordaten zu den gewählten Tätigkeiten erzeugen. Die Infor-

mationen dieser beiden Kontexterzeuger gilt es in einem weiteren Schritt in der Form zu verschmelzen, dass zunächst die logischen und abgeleiteten Bedingungen definiert werden können und daher die Grundlage für die gewählten Aktivitäten des Skriptsimulators bilden. Hierzu kann letzterer die über den ActiveMQ versendeten Nachrichten des Kontextsimulators abonnieren, diese um die ausgeführten Tätigkeiten erweitern, bevor sie daraufhin an die weiteren Anwendungen verschickt werden. Die jeweiligen Applikationen müssten sich folglich ausschließlich am Topic des Skriptsimulators anmelden.
Der in dieser Arbeit implementierte Kontextsimulator bietet seinem Benutzer neben der Definition aller angebotenen Werte die Möglichkeit, einzelne Werte simulieren, da es in verschiedenen Testfällen denkbar ist, dass lediglich eine partielle Simulation gewünscht ist. Deshalb bietet das hier vorgestellte System zu jedem Datum die Option, es mit fiktiven oder realen Sensorwerten zu belegen.

7 Schluss

In diesem Kapitel werden im Rahmen einer Schlussbetrachtung die grundlegenden Vorgehensweisen sowie die in dieser Arbeit erzielten Erkenntnisse zusammengefasst. Darüber hinaus wird ein Ausblick auf mögliche Erweiterungen und Optimierungen im Hinblick auf das hier implementierte System präsentiert.

7.1 Zusammenfassung

In dieser Arbeit konnte ein System entwickelt werden, welches eine elementare Infrastruktur für die Kommunikation zwischen den Anwendungen im *Living Place Hamburg* bietet. Ferner wurde eine Applikation entwickelt, die auf der Basis der zentralen Kommunikationsschnittelle eine modellunabhängige Kontextinterpretation durchführt. Die Resultate dieser Interpretation werden daraufhin mit weiteren Interpretationsschichten kombiniert, um verschiedene Aktivitäten innerhalb mehrerer Morgen-Szenarien zu detektieren. Zur Herstellung diverser, für diese und weitere Szenarien erforderlichen Rahmenbedingungen und Testumgebungen, wurde in dieser Arbeit zudem ein Kontextsimulator erstellt, mit dessen Hilfe ein nahtloses Wechseln zwischen realen und simulierten Sensordaten ermöglicht wird.
In der Analyse (3) konnten drei Szenarien herausgearbeitet werden, in denen verschiedene Morgenaktivitäten der fiktiven Bewohnerin des *Living Place Hamburg* beschrieben wurden und als Grundlage für eine Anforderungsanalyse an ein System, welches diese Tätigkeiten erkennen und darauf reagieren kann, herangezogen. Hier konnte ermittelt werden, dass der Kontext einen essentiellen Aspekt bei der Aktivitätserkennung darstellt, weshalb verschiedene „Context Provider“ im Hinblick auf ihre Fähigkeiten untersucht wurden. Zudem konnten die Herausforderungen im Umgang mit Kontextinformationen und die Anforderungen an eine Architektur für kontextsensitive Systeme identifiziert werden.
Im weiteren Verlauf der Analyse wurden mit der modellunabhängigen, der modellabhängigen und der ontologiebasierten Kontextinterpretation drei unterschiedliche Ansätze zur Verarbeitung der vorliegenden Informationen vorgestellt und verdeutlicht, dass eine Kombination der Vorgehensweisen bei der Kontexterstellung sinnvoll ist.
Bedingt durch die hohe Anzahl an Sensordaten, die bei der Interpretation von Kontexten in *Smart Homes* generiert wird, befasst sich der Abschnitt 4 mit dem in großen Datensätzen

üblicherweise auftretenden Rauschen. Hierzu wurden verschiedene Mechanismen zur Detektion und zur Behandlung von Ausreißern vorgestellt. Des Weiteren liefert das Kapitel eine Einführung in den Bereich der Clusteranalyse sowie in die Disziplin der Sensorfusion.
Die in der Analyse identifizierten Architekturanforderungen wurden in 5 aufgegriffen, mit dem Ziel zentrale Designentscheidungen im Hinblick auf die Kommunikation und Persistierung von Sensordaten zu treffen. Zur Realisierung der Kommunikationskomponente konnte die Blackboard-Architektur an Profil gewinnen, da hier Interessenten einer Nachrichtenform automatisch benachrichtigt werden, sobald ein entsprechendes Datum auftritt, ohne dass sich hierzu der Sender und der Empfänger kennen müssen. Für die Persistenzebene wurde ein dokumentenbasierter Ansatz gewählt, da Kontextdaten oftmals unstrukturiert vorliegen und daher lediglich mit großem Aufwand in ein relationales Modell überführt werden können. Weiterhin wurden die Aufgaben der drei Interpretationslevel bestimmt sowie die zu erkennenden Aktivitäten der Morgen-Szenarien ermittelt und in eine zeitliche Reihenfolge gebracht.
Im Realisierungsteil (6) dieser Arbeit wurden die im Design getroffenen Entscheidungen in konkrete Anwendungen eingebettet. Die Integration der implementierten Komponenten in den *Living Place Hamburg* sowie deren Evaluation konnte dabei Anregungen liefern, inwiefern das hier vorgestellte System erweitert und verbessert werden kann.

7.2 Fazit und Ausblick

Der Schwerpunkt dieser Arbeit lag in der Untersuchung, in welchem Umfang ubiquitäre Informationstechnologien in private Wohnräume integrierbar sind und wie diese dem Bewohner beim Erledigen von alltäglichen Aufgaben behilflich sein können. Dabei liegt ein elementares Ziel in der intuitiven Interaktion zwischen Mensch und Computer, was eine selbständige und kontextabhängige Reaktion der Systeme erstrebenswert macht. Dies bedeutet, dass die Person lediglich als Teil der Umgebung empfunden wird und keine zusätzlichen Benutzereingaben für systemseitige Aktionen erforderlich sind. Die Computer entschwinden demnach der menschlichen Wahrnehmung als technisches Gerät und werden als im Hintergrund agierende Dienstleister betrachtet.

Je automatisierter Computersysteme ihr Verhalten in Eigenverantwortung bestimmen, desto größer wird der gefühlte Kontrollverlust des Anwenders. Dennoch zeigen Forschungsergebnisse dass viele Benutzer bereit sind einen hohen Grad an Autonomie zu akzeptieren, solange der Nutzen der Anwendung den Aufwand einer partiellen Kontrolle übersteigt, vgl. Barkhuus und Dey (2003).
Allerdings darf dem Anwender keinesfalls vollständig die Steuerung des Systems entzogen werden. Es gilt hier daher zukünftig eine Balance zwischen den automatisierten Verhaltensweisen der Systeme und den Eingriffsmöglichkeiten des Anwenders zu schaffen. Diesbezüglich ist es zu erforschen, wie Interaktionsmechanismen konstruiert werden können, so-

dass der Nutzer intuitiv eine Rückmeldung hinsichtlich des Systemverhaltens liefern kann. Vor diesem Hintergrund sind Herausforderungen wie die Umsetzung neuartiger Interaktionskonzepte und die Auflösung von Hindernissen beim Beenden von Anwendungen, wie das „Midas Touch Problem“ (Jacob (1990)), bei dem der Benutzer mit jeder Interaktion eine neue ungewollte Reaktion des Systems auslöst, von essentieller Bedeutung und eine zu beantwortende Fragestellung in der zukünftigen Interaktion zwischen dem Menschen und den ihn umgebenen Computern.

Zur Realisierung von Context-Aware Systemen ist die Ermittlung des gegenwärtigen Kontextes ein grundlegender Vorgang. Für diesen Prozess existiert bereits eine Reihe von Informationsquellen, deren Daten zur Ermittlung des Bezugsrahmens beitragen können. Die Verknüpfung der einzelnen Informationen zu einem allgemeingültigen Kontext und der damit verbundenen Kaskade von Interpretationsschritten stellt dagegen ein aktuelles Forschungsgebiet dar, für welches bisher keine einheitliche Lösungsansätze bestehen. In dieser Arbeit wurde eine auf drei Schichten basierende Interpretation konzipiert, die es erlaubt zahlreiche Umgebungszustände und Aktivitäten zu erkennen. Ein *Smart Home* stellt jedoch ein sehr dynamisches Umfeld dar, in dem sich regelmäßig dem System unbekannte Situationen ergeben können. Aus diesem Grund muss weiterhin eruiert werden, welche Daten in den Kontext einzubeziehen sind und was adäquate systemseitige Handlungen auf einen Zustandswechsel darstellen. In diesem Bereich gilt es zum einen zu erkunden, inwieweit eine automatische Detektion von Kontexten realisierbar ist. Zum anderen ist zu beleuchten, in welcher Weise die Systeme durch den Einsatz von z. B. *Reinforcement Learning* (Sutton und Barto (1998)) eigenständig lernen können, welche Reaktion zu einem vorliegenden Zustand passt.

Die in dieser Arbeit beschriebenen Szenarien und Anwendungen wurden vor dem Hintergrund entwickelt, den Komfort für den Bewohner eines intelligenten Wohnbereichs zu erhöhen. Dabei werden weitere Entwicklungen zeigen, ob sich die hier vorgestellten Konzepte für die, in Abschnitt 2 erläuterten, weiteren Segmente (Energiemanagement, Ambient Assisted Living, Sicherheit) im Bereich der *Smart Homes* übertragen lassen. Darüber hinaus ist zu überprüfen, in welchem Umfang das hier vorgestellte System in weitere intelligente Umgebungen, wie Büros oder öffentliche Gebäude, integrierbar ist.

Neben der Bewältigung dieser technischen Herausforderungen, ist die soziale Akzeptanz der Technologien ein weiterer entscheidender Faktor für den Erfolg des ubiquitären Computerzeitalters. Zur Verwirklichung dieses Ziels ist es maßgebend, dass der Benutzer fortwährend positive Erfahrungen mit den Systemen dieser Art sammelt und sich eine gute *User Experience* ergibt. In der ISO 9241-210 wird die *User Experience* als die Empfindungen und Reaktionen einer Person auf die Nutzung oder der Erwartungshaltung bezüglich der Verwendung eines Produktes, eines Systems oder einer Dienstleistung beschrieben und resultieren folglich aus subjektiven Eindrücken. In diesem Zusammenhang drückt Donald Norman, der den Begriff der *User Experience* entscheidend geprägt hat, die Anforderungen an Computersysteme in Norman (2004) wie folgt aus:

> *„It is not enough that we build products that function, that are understandable and usable, we also need to build products that bring joy and excitement, pleasure and fun, and, yes, beauty to people's lives."*

Die Bearbeitung der beschriebenen Herausforderungen erfordert eine Kooperation verschiedenster Forschungsdisziplinen aus Bereichen wie Informatik, Design und Sozialwissenschaft. Hierzu bildet das *Living Place Hamburg* eine lebendige und offene Laborumgebung und liefert somit eine Plattform für einen interdisziplinären und kreativen Ideenaustausch, um neuartige Informationstechnologien und Aspekte des *Ubiquitous Computing* zu erforschen. In diesem Kontext werden über diese Arbeit hinaus in der Zukunft einige elementare Fragestellungen zu beantworten sein. Hierzu gehört die Beantwortung der Frage, ob die Evolution zu immer kleineren und leistungsfähigeren Computern in Verbindung mit der Generation der *Digital Natives* automatisch einen nahtlosen Übergang in das Zeitalter des *Ubiquitous Computing* bedeutet oder ob hierzu revolutionäre Innovationen in der Computertechnik sowie vollständig neue Interaktionskonzepte unabdingbar sind.
Außerdem gilt es weiterhin zu erforschen, was Komfort in einem Wohnbereich für den einzelnen Menschen bedeutet und wie die Selbstbestimmung des Menschen gewährleistet wird, sodass er sich selbst als eigenständig handelndes Individuum begreift, wenn ihn kontinuierlich Computer umgeben, die kontextbezogen auf die äußeren Gegebenheiten reagieren.

Literaturverzeichnis

[Abowd u. a. 1999] ABOWD, Gregory D. ; DEY, Anind K. ; BROWN, Peter J. ; DAVIES, Nigel ; SMITH, Mark ; STEGGLES, Pete: Towards a Better Understanding of Context and Context-Awareness. In: *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*. London, UK : Springer-Verlag, 1999 (HUC '99), S. 304–307. – URL http://dl.acm.org/citation.cfm?id=647985.743843. – Zugriffsdatum: 13.12.2011. – ISBN 3-540-66550-1

[Allen 1983] ALLEN, James F.: Maintaining knowledge about temporal intervals. In: *Commun. ACM* 26 (1983), November, S. 832–843. – URL http://doi.acm.org/10.1145/182.358434. – Zugriffsdatum: 13.12.2011. – ISSN 0001-0782

[Anderson u. a. 2010] ANDERSON, J. C. ; LEHNARDT, Jan ; SLATER, Noah: *CouchDB: The Definitive Guide Time to Relax*. 1st. O'Reilly Media, Inc., 2010. – ISBN 0596155891, 9780596155896

[Bao und Intille 2004] BAO, Ling ; INTILLE, Stephen S.: Activity Recognition from User-Annotated Acceleration Data. In: *Pervasive*, 2004, S. 1–17

[Bardram 2004] BARDRAM, Jakob E.: Applications of context-aware computing in hospital work: examples and design principles. In: *Proceedings of the 2004 ACM symposium on Applied computing*. New York, NY, USA : ACM, 2004 (SAC '04), S. 1574–1579. – URL http://doi.acm.org/10.1145/967900.968215. – Zugriffsdatum: 13.12.2011. – ISBN 1-58113-812-1

[Barkhuus und Dey 2003] BARKHUUS, Louise ; DEY, Anind: Is Context-Aware Computing Taking Control away from the User? Three Levels of Interactivity Examined. In: DEY, Anind (Hrsg.) ; SCHMIDT, Albrecht (Hrsg.) ; MCCARTHY, Joseph (Hrsg.): *UbiComp 2003: Ubiquitous Computing* Bd. 2864. Springer Berlin / Heidelberg, 2003, S. 149–156. – URL http://dx.doi.org/10.1007/978-3-540-39653-6_12. – Zugriffsdatum: 13.12.2011. – 10.1007/978-3-540-39653-6_12. – ISBN 978-3-540-20301-8

[Barmettler 2011] BARMETTLER, Arnold: *Eine einfache Formel zu Sonnenaufgang und Untergang*. 2011. – URL http://lexikon.astronomie.info/zeitgleichung/. – Zugriffsdatum: 13.12.2011

[Barnkow 2010] BARNKOW, Lorenz: Eine Multitouch-fähige Küchentheke: Vorbereitende Arbeiten für den Tagesplaner / Hochschule für Angewandte Wissenschaften Hamburg. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2010-proj1/barnkow.pdf. – Zugriffsdatum: 13.12.2011, 2010. – Forschungsbericht

[Basener 2011] BASENER, Andreas: Entwicklung eines Skriptsimulators für das Living-Place / Hochschule für Angewandte Wissenschaften Hamburg. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2011-proj1/basener.pdf. – Zugriffsdatum: 13.12.2011, 2011. – Forschungsbericht

[Beck 2002] BECK: *Test Driven Development: By Example*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 2002. – 192 S. – ISBN 0321146530

[Bettini u. a. 2010] BETTINI, Claudio ; BRDICZKA, Oliver ; HENRICKSEN, Karen ; INDULSKA, Jadwiga ; NICKLAS, Daniela ; RANGANATHAN, Anand ; RIBONI, Daniele: A survey of context modelling and reasoning techniques. In: *Pervasive Mob. Comput.* 6 (2010), April, S. 161–180. – URL http://dx.doi.org/10.1016/j.pmcj.2009.06.002. – Zugriffsdatum: 13.12.2011. – ISSN 1574-1192

[Bhatt u. a. 2009] BHATT, Mehul ; DYLLA, Frank ; HOIS, Joana: Spatio-terminological Inference for the Design of Ambient Environments. In: HORNSBY, Kathleen (Hrsg.) ; CLARAMUNT, Christophe (Hrsg.) ; DENIS, Michel (Hrsg.) ; LIGOZAT, Gerard (Hrsg.): *Spatial Information Theory* Bd. 5756. Springer Berlin / Heidelberg, 2009, S. 371–391. – URL http://dx.doi.org/10.1007/978-3-642-03832-7_23. – Zugriffsdatum: 13.12.2011. – 10.1007/978-3-642-03832-7_23. – ISBN 978-3-642-03831-0

[Biundo und Wendemuth 2010] BIUNDO, Susanne ; WENDEMUTH, Andreas: Von kognitiven technischen Systemen zu Companion-Systemen. In: *KI - Künstliche Intelligenz* 24 (2010), S. 335–339. – URL http://dx.doi.org/10.1007/s13218-010-0056-9. – Zugriffsdatum: 13.12.2011. – 10.1007/s13218-010-0056-9. – ISSN 0933-1875

[Bizer u. a. 2006] BIZER, J. ; GÜNTHER, O. u. a.: TAUCIS-Technikfolgenabschätzungsstudie Ubiquitäres Computing und Informationelle Selbstbestimmung. In: *B. f. B. u. Forschung. Berlin, Germany, Humboldt University Berlin, Unabhängiges Landeszentrum für Datenschutz Schleswig-Holstein (ULD)* (2006)

[Borrmann u. a. 2006] BORRMANN, André ; TREECK, Christoph V. ; RANK, Ernst: Towards a 3D Spatial Query Language for Building Information Models. In: *Proc. Joint Int. Conf. of Computing and Decision Making in Civil and Building Engineering (ICCCBE-XI)*, 2006

[Buschmann u. a. 1998] BUSCHMANN, Frank ; MEUNIER, Regine ; ROHNERT, Hans ; SOMMERLAD, Peter ; STAL, Michael: *Pattern-orientierte Software-Architektur . Ein Pattern-System*. Addison-Wesley, 1998. – 480 S. – ISBN 3827312825

[Castro und Munz 2000] CASTRO, P. ; MUNZ, R.: Managing context data for smart spaces. In: *Personal Communications, IEEE [see also IEEE Wireless Communications]* 7 (2000), Nr. 5, S. 44–46. – URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=878537. – Zugriffsdatum: 13.12.2011

[Cha 2007] CHA, Sung-Hyuk: *Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions*. 2007

[Chang u. a. 2006] CHANG, Fay ; DEAN, Jeffrey ; GHEMAWAT, Sanjay ; HSIEH, Wilson C. ; WALLACH, Deborah A. ; BURROWS, Mike ; CHANDRA, Tushar ; FIKES, Andrew ; GRUBER, Robert E.: Bigtable: A distributed storage system for structured data. In: *IN PROCEEDINGS OF THE 7TH CONFERENCE ON USENIX SYMPOSIUM ON OPERATING SYSTEMS DESIGN AND IMPLEMENTATION - VOLUME 7*, 2006, S. 205–218

[Chen 2004] CHEN, Harry: *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. University of Maryland, Baltimore County, Dissertation, 2004

[Chen u. a. 2003] CHEN, Harry ; FININ, Tim ; JOSHI, Anupam: An ontology for context-aware pervasive computing environments. In: *Knowl. Eng. Rev.* 18 (2003), Nr. 3, S. 197–207. – ISSN 0269-8889

[Codd 1970] CODD, E. F.: A relational model of data for large shared data banks. In: *Commun. ACM* 13 (1970), June, S. 377–387. – URL http://doi.acm.org/10.1145/362384.362685. – Zugriffsdatum: 13.12.2011. – ISSN 0001-0782

[Corkill 1991] CORKILL, Daniel: Blackboard Systems. In: *AI Expert* 6 (1991), January, Nr. 9, S. 40 –47. – URL http://mas.cs.umass.edu/paper/218. – Zugriffsdatum: 13.12.2011

[Coulouris u. a. 2005] COULOURIS, George ; DOLLIMORE, Jean ; KINDBERG, Tim: *Verteilte Systeme. Konzepte und Design*. 3., überarb. A. Pearson Studium, 2005. – ISBN 3827371864

[Dean und Ghemawat 2004] DEAN, Jeffrey ; GHEMAWAT, Sanjay: MapReduce: simplified data processing on large clusters. In: *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6*. Berkeley, CA, USA : USENIX Association, 2004, S. 10–10. – URL http://dl.acm.org/citation.cfm?id=1251254.1251264. – Zugriffsdatum: 13.12.2011

[Dearman und Pierce 2008] DEARMAN, David ; PIERCE, Jeffery S.: It's on my other computer!: computing with multiple devices. In: *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM, 2008 (CHI '08), S. 767–776. – URL http://doi.acm.org/10.1145/1357054.1357177. – Zugriffsdatum: 13.12.2011. – ISBN 978-1-60558-011-1

[DeCandia u. a. 2007] DECANDIA, Giuseppe ; HASTORUN, Deniz ; JAMPANI, Madan ; KAKULAPATI, Gunavardhan ; LAKSHMAN, Avinash ; PILCHIN, Alex ; SIVASUBRAMANIAN, Swaminathan ; VOSSHALL, Peter ; VOGELS, Werner: Dynamo: amazon's highly available key-value store. In: *SIGOPS Oper. Syst. Rev.* 41 (2007), October, S. 205–220. – URL http://doi.acm.org/10.1145/1323293.1294281. – Zugriffsdatum: 13.12.2011. – ISSN 0163-5980

[Dempster 2008] DEMPSTER, Arthur: Upper and Lower Probabilities Induced by a Multivalued Mapping. In: YAGER, Roland (Hrsg.) ; LIU, Liping (Hrsg.): *Classic Works of the Dempster-Shafer Theory of Belief Functions* Bd. 219. Springer Berlin / Heidelberg, 2008, S. 57–72. – URL http://dx.doi.org/10.1007/978-3-540-44792-4_3. – Zugriffsdatum: 13.12.2011

[Dey u. a. 2001] DEY, Anind K. ; ABOWD, Gregory D. ; SALBER, Daniel: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. In: *Hum.-Comput. Interact.* 16 (2001), December, S. 97–166. – URL http://dx.doi.org/10.1207/S15327051HCI16234_02. – Zugriffsdatum: 13.12.2011. – ISSN 0737-0024

[Dreschke 2011] DRESCHKE, Oliver: *Entwicklung kontextsensitiver Möbel für intelligente Wohnumgebungen*, Hochschule für Angewandte Wissenschaften Hamburg, Masterarbeit, 2011. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/dreschke.pdf. – Zugriffsdatum: 13.12.2011

[Edlich u. a. 2010] EDLICH, Stefan ; FRIEDLAND, Achim ; HAMPE, Jens ; BRAUER, Benjamin: *NoSQL: Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken*. Carl Hanser Verlag GmbH & CO. KG, 2010. – 304 S. – ISBN 978-3446423558

[Edwards und Grinter 2001] EDWARDS, W. K. ; GRINTER, Rebecca E.: At Home with Ubiquitous Computing: Seven Challenges. In: *Proceedings of the 3rd international conference on Ubiquitous Computing*. London, UK, UK : Springer-Verlag, 2001 (UbiComp '01), S. 256–272. – URL http://dl.acm.org/citation.cfm?id=647987.741327. – Zugriffsdatum: 13.12.2011. – ISBN 3-540-42614-0

[Ellenberg 2011] ELLENBERG, Jens: *Ontologiebasierte Aktivitätserkennung im Smart Home Kontext*, Hochschule für angewandte Wissenschaften Hamburg, Masterarbeit, 2011

[Ellenberg u. a. 2011] ELLENBERG, Jens ; KARSTAEDT, Bastian ; VOSKUHL, Sören ; LUCK, Kai von ; WENDHOLT, Birgit: An Environment for Context-Aware Applications in Smart Homes / Hamburg University of Applied Sciences. URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/papers/IPIN2011.pdf. – Zugriffsdatum: 13.12.2011, 2011. – Forschungsbericht

[Erman u. a. 1980] ERMAN, Lee D. ; HAYES-ROTH, Frederick ; LESSER, Victor R. ; REDDY, D. R.: The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. In: *ACM Comput. Surv.* 12 (1980), Nr. 2, S. 213–253. – ISSN 0360-0300

[Ester u. a. 1996] ESTER, Martin ; KRIEGEL, Hans peter ; SANDER, Jörg ; XU, Xiaowei: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: *Knowledge Discovery and Data Mining*, 1996, S. 226–231

[Eugster u. a. 2003] EUGSTER, Patrick T. ; FELBER, Pascal A. ; GUERRAOUI, Rachid ; KERMARREC, Anne-Marie: The many faces of publish/subscribe. In: *ACM Comput. Surv.* 35 (2003), June, S. 114–131. – URL http://doi.acm.org/10.1145/857076.857078. – Zugriffsdatum: 13.12.2011. – ISSN 0360-0300

[Fahy und Clarke 2004] FAHY, Patrick ; CLARKE, Siobhan: CASS - Middleware for Mobile Context-Aware Applications. (2004). – URL http://www.sigmobile.org/mobisys/2004/context_awareness/papers/cass12f.pdf. – Zugriffsdatum: 13.12.2011

[Fidge 1988] FIDGE, Colin J.: Timestamps in message-passing systems that preserve the partial ordering. In: *Proc. of the 11th Australian Computer Science Conference (ACSC'88)*, Februar 1988, S. 56–66

[Fontana u. a. 2003] FONTANA, R.J. ; RICHLEY, E. ; BARNEY, J.: Commercialization of an ultra wideband precision asset location system. In: *Ultra Wideband Systems and Technologies, 2003 IEEE Conference on*, nov. 2003, S. 369 – 373

[Galton 2008] GALTON, Antony: Pareto-Optimality of Cognitively Preferred Polygonal Hulls for Dot Patterns. In: FREKSA, Christian (Hrsg.) ; NEWCOMBE, Nora (Hrsg.) ; GÄRDENFORS, Peter (Hrsg.) ; WÖLFL, Stefan (Hrsg.): *Spatial Cognition VI. Learning, Reasoning, and Talking about Space* Bd. 5248. Springer Berlin / Heidelberg, 2008, S. 409–425. – URL http://dx.doi.org/10.1007/978-3-540-87601-4_29. – Zugriffsdatum: 13.12.2011. – 10.1007/978-3-540-87601-4_29. – ISBN 978-3-540-87600-7

[Gavrila 1999] GAVRILA, D. M.: The visual analysis of human movement: a survey. In: *Comput. Vis. Image Underst.* 73 (1999), January, S. 82–98. – URL http://dl.acm.org/citation.cfm?id=308376.308382. – Zugriffsdatum: 13.12.2011. – ISSN 1077-3142

[Glasberg und Feldner 2009] GLASBERG, Ronald ; FELDNER, Nadja: *Leitfaden zur Heimvernetzung*. BITKOM - Bundesverband Informationswirtschaft,Telekommunikation und neue Medien e. V., 2009. – URL http://www.bitkom.org/files/documents/BITKOM_Heimvernetzungs-Leitfaden_20091209.pdf. – Zugriffsdatum: 13.12.2011

[Grechenig u. a. 2010] GRECHENIG, Thomas ; BERNHART, Mario ; BREITENEDER, Roland ; KAPPEL, Karin: *Softwaretechnik - Mit Fallbeispielen aus realen Entwicklungsprojekten*. Pearson Studium, 2010. – 688 S. – ISBN 978-3-86894-007-7

[Greer 2008] GREER, Derek: *The Art of Separation of Concerns*. 2008. – URL http://www.aspiringcraftsman.com/2008/01/03/art-of-separation-of-concerns/. – Zugriffsdatum: 13.12.2011

[Grubbs 1969] GRUBBS, Frank: Procedures for Detecting Outlying Observations in Samples. In: *Technometrics* 11 (1969), Nr. 1, S. 1–21

[Gu u. a. 2004] GU, Tao ; PUNG, Hung K. ; ZHANG, Da Q.: A middleware for building context-aware mobile services. In: *Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th* Bd. 5, may 2004, S. 2656 – 2660 Vol.5. – ISSN 1550-2252

[Gu u. a. 2005] GU, Tao ; PUNG, Hung K. ; ZHANG, Da Q.: A service-oriented middleware for building context-aware services. In: *J. Netw. Comput. Appl.* 28 (2005), January, S. 1–18. – URL http://dl.acm.org/citation.cfm?id=1053030.1053031. – Zugriffsdatum: 13.12.2011. – ISSN 1084-8045

[Gu u. a. 2008] GU, Tao ; PUNG, Hung K. ; ZHANG, Daqing: Peer-to-Peer Context Reasoning in Pervasive Computing Environments. In: *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, march 2008, S. 406 –411

[Hansen und Meissner 2007] HANSEN, M. ; MEISSNER, S.: *Verkettung digitaler Identitäten*. Lulu Inc., 2007. – URL http://books.google.com/books?id=k49bPg1_vYcC. – Zugriffsdatum: 13.12.2011. – ISBN 9783000234064

[Hardenack 2011] HARDENACK, Frank: Das intelligente Bett - Sensorbasierte Detektion von Schlafphasen / Hochschule für Angewandte Wissenschaften Hamburg. URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/hardenack.pdf. – Zugriffsdatum: 13.12.2011, 2011. – Forschungsbericht

[Hartung u. a. 1999] HARTUNG, J. ; ELPELT, B. ; KLÖSENER, K.H.: *Statistik: Lehr- und Handbuch der angewandten Statistik*. Oldenbourg, 1999. – ISBN 3-486-24984-3

[Hawkins 1980] HAWKINS, D.M.: *Identification of outliers*. Chapman and Hall, 1980 (Monographs on applied probability and statistics). – URL http://books.google.de/books?id=fb0OAAAAQAAJ. – Zugriffsdatum: 13.12.2011. – ISBN 9780412219009

[Henjes u. a. 2007] HENJES, Robert ; SCHLOSSER, Daniel ; MENTH, Michael ; HIMMLER, Valentin: Throughput Performance of the ActiveMQ JMS Server. In: BRAUN, Torsten (Hrsg.) ; CARLE, Georg (Hrsg.) ; STILLER, Burkhard (Hrsg.) ; BRAUER, W. (Hrsg.): *Kommunikation in Verteilten Systemen (KiVS)*. Springer Berlin Heidelberg, 2007 (Informatik aktuell), S. 113–124. – URL http://dx.doi.org/10.1007/978-3-540-69962-0_10. – Zugriffsdatum: 13.12.2011. – 10.1007/978-3-540-69962-0_10. – ISBN 978-3-540-69962-0

[Henricksen u. a. 2002] HENRICKSEN, Karen ; INDULSKA, Jadwiga ; RAKOTONIRAINY, Andry: Modeling Context Information in Pervasive Computing Systems. In: *Proceedings of the First International Conference on Pervasive Computing*. London, UK : Springer-Verlag, 2002 (Pervasive '02), S. 167–180. – URL http://dl.acm.org/citation.cfm?id=646867.706693. – Zugriffsdatum: 13.12.2011. – ISBN 3-540-44060-7

[Hofer u. a. 2003] HOFER, Thomas ; SCHWINGER, Wieland ; PICHLER, Mario ; LEONHARTSBERGER, Gerhard ; ALTMANN, Josef ; RETSCHITZEGGER, Werner: Context-Awareness on Mobile Devices - the Hydrogen Approach. In: *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9 - Volume 9*. Washington, DC, USA : IEEE Computer Society, 2003 (HICSS '03), S. 292.1–. – URL http://dl.acm.org/citation.cfm?id=820756.821849. – Zugriffsdatum: 13.12.2011. – ISBN 0-7695-1874-5

[Honkavirta u. a. 2009] HONKAVIRTA, V. ; PERALA, T. ; ALI-LOYTTY, S. ; PICHE, R.: A comparative survey of WLAN location fingerprinting methods. In: *Positioning, Navigation and Communication, 2009. WPNC 2009. 6th Workshop on*, march 2009, S. 243 –251

[Hürsch u. a. 1995] HÜRSCH, Walter ; LOPES, Cristina ; HURSCH, Walter L. ; LOPES, Cristina V.: *Separation of concerns*. 1995. – URL ftp://ftp.ccs.neu.edu/pub/people/lieber/crista/techrep95/index.html

[Ishii und Ullmer 1997] ISHII, Hiroshi ; ULLMER, Brygg: Tangible bits: towards seamless interfaces between people, bits and atoms. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM, 1997 (CHI '97), S. 234–241. – URL http://doi.acm.org/10.1145/258549.258715. – Zugriffsdatum: 13.12.2011. – ISBN 0-89791-802-9

[Jacob 1990] JACOB, Robert J. K.: What you look at is what you get: eye movement-based interaction techniques. In: *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people*. New York, NY, USA : ACM, 1990 (CHI '90),

S. 11–18. – URL http://doi.acm.org/10.1145/97243.97246. – Zugriffsdatum: 13.12.2011. – ISBN 0-201-50932-6

[Jain u. a. 1999] JAIN, A. K. ; MURTY, M. N. ; FLYNN, P. J.: Data clustering: a review. In: *ACM Comput. Surv.* 31 (1999), September, S. 264–323. – URL http://doi.acm.org/10.1145/331499.331504. – Zugriffsdatum: 13.12.2011. – ISSN 0360-0300

[Jang und Woo 2003] JANG, Seiie ; WOO, Woontack: Ubi-UCAM: a unified context-aware application model. In: *Proceedings of the 4th international and interdisciplinary conference on Modeling and using context*. Berlin, Heidelberg : Springer-Verlag, 2003 (CONTEXT'03), S. 178–189. – URL http://dl.acm.org/citation.cfm?id=1763142.1763158. – Zugriffsdatum: 13.12.2011. – ISBN 3-540-40380-9

[Kalman 1960] KALMAN, Rudolph E.: A New Approach to Linear Filtering and Prediction Problems. In: *Transactions of the ASME–Journal of Basic Engineering* 82 (1960), Nr. Series D, S. 35–45

[Karstaedt 2011] KARSTAEDT, Bastian: Entwicklung und Integration des Indoor Spatial Information Services in den Living Place Hamburg / Hochschule für Angewandte Wissenschaften Hamburg. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master10-11-proj2/karstaedt.pdf. – Zugriffsdatum: 13.12.2011, 2011. – Forschungsbericht

[Karstaedt 2012] KARSTAEDT, Bastian: *Kontextinterpretation in Smart Home auf Basis 3D semantischer Gebäudemodelle*, Hochschule für angewandte Wissenschaften Hamburg, Masterarbeit, 2012

[Klein 1997] KLEIN, Karl-Josef: *Algorithmische Geometrie*. Addison-Wesley Longman Verlag GmbH, 1997. – 400 S. – ISBN 3-8273-1111-x

[Lakshman und Malik 2010] LAKSHMAN, Avinash ; MALIK, Prashant: Cassandra: a decentralized structured storage system. In: *SIGOPS Oper. Syst. Rev.* 44 (2010), April, S. 35–40. – URL http://doi.acm.org/10.1145/1773912.1773922. – Zugriffsdatum: 13.12.2011. – ISSN 0163-5980

[Lamport 1978] LAMPORT, Leslie: Ti clocks, and the ordering of events in a distributed system. In: *Commun. ACM* 21 (1978), July, S. 558–565. – URL http://doi.acm.org/10.1145/359545.359563. – Zugriffsdatum: 13.12.2011. – ISSN 0001-0782

[Leavitt 2010] LEAVITT, N.: Will NoSQL Databases Live Up to Their Promise? In: *Computer* 43 (2010), feb., Nr. 2, S. 12 –14. – ISSN 0018-9162

[Lee u. a. 2007] LEE, Jae-Gil ; HAN, Jiawei ; WHANG, Kyu-Young: Trajectory clustering: a partition-and-group framework. In: *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. New York, NY, USA : ACM, 2007 (SIGMOD

’07), S. 593–604. – URL http://doi.acm.org/10.1145/1247480.1247546. – Zugriffsdatum: 13.12.2011. – ISBN 978-1-59593-686-8

[Liao u. a. 2005] LIAO, Lin ; FOX, Dieter ; KAUTZ, Henry: Location-based activity recognition using relational Markov networks. In: *Proceedings of the 19th international joint conference on Artificial intelligence*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2005 (IJCAI’05), S. 773–778. – URL http://dl.acm.org/citation.cfm?id=1642293.1642417. – Zugriffsdatum: 13.12.2011

[Lowe 1999] LOWE, David G.: Object Recognition from Local Scale-Invariant Features. In: *Computer Vision, IEEE International Conference on* 2 (1999), August, S. 1150–1157 vol.2. – URL http://dx.doi.org/10.1109/ICCV.1999.790410. – Zugriffsdatum: 13.12.2011. ISBN 0-7695-0164-8

[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. – Zugriffsdatum: 13.12.2011, 2010. – Forschungsbericht

[MacQueen 1967] MACQUEEN, J. B.: Some Methods for Classification and Analysis of MultiVariate Observations. In: *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability* Bd. 1, University of California Press, 1967, S. 281–297

[Mattern 1989] MATTERN, Friedemann: Virtual Time and Global States of Distributed Systems. In: *Parallel and Distributed Algorithms*, North-Holland, 1989, S. 215–226

[Maybeck 1979] MAYBECK, Peter S.: *Mathematics in Science and Engineering*. Bd. 141: *Stochastic models, estimation, and control*. 1979

[Meyer und Rakotonirainy 2003] MEYER, Sven ; RAKOTONIRAINY, Andry: A survey of research on context-aware homes. In: *Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003 - Volume 21*. Darlinghurst, Australia, Australia : Australian Computer Society, Inc., 2003 (ACSW Frontiers ’03), S. 159–168. – URL http://dl.acm.org/citation.cfm?id=827987.828005. – Zugriffsdatum: 13.12.2011. – ISBN 1-920682-00-7

[Müller 2010] MÜLLER, Larissa: Interactive Design - Studien der interdisziplinären Zusammenarbeit von Design und Informatik / Hochschule für Angewandte Wissenschaften Hamburg. 2010. – Bachelorarbeit

[Moreira und Santos 2007] MOREIRA, Adriano J. C. ; SANTOS, Maribel Y.: Concave hull: A k-nearest neighbours approach for the computation of the region occupied by a set of points. In: *GRAPP (GM/R)*, 2007, S. 61–68

[Najem 2011] NAJEM, Hosnia: Modellbasiertes Suchen von Objekten in einer Smart-Home-Umgebung / Hochschule für Angewandte Wissenschaften Hamburg. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2011-proj1/najem.pdf. – Zugriffsdatum: 13.12.2011, 2011. – Forschungsbericht

[Norman 2004] NORMAN, Donald A.: Introduction to this special section on beauty, goodness, and usability. In: *Hum.-Comput. Interact.* 19 (2004), December, S. 311–318. – URL http://dx.doi.org/10.1207/s15327051hci1904_1. – Zugriffsdatum: 13.12.2011. – ISSN 0737-0024

[Otto und Voskuhl 2010] OTTO, Kjell ; VOSKUHL, Sören: Entwicklung einer Architektur für den Living Place Hamburg / Hamburg University of Applied Sciences. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2010-proj1/otto_voskuhl.pdf. – Zugriffsdatum: 13.12.2011, 2010. – Forschungsbericht

[Otto und Voskuhl 2011] OTTO, Kjell ; VOSKUHL, Sören: Weiterentwicklung der Architektur des Living Place Hamburg / Hamburg University of Applied Sciences. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master10-11-proj2/otto-voskuhl.pdf. – Zugriffsdatum: 13.12.2011, 2011. – Forschungsbericht

[Patterson u. a. 2003] PATTERSON, Donald J. ; LIAO, Lin ; FOX, Dieter ; KAUTZ, Henry A.: Inferring High-Level Behavior from Low-Level Sensors. In: DEY, Anind K. (Hrsg.) ; SCHMIDT, Albrecht (Hrsg.) ; MCCARTHY, Joseph F. (Hrsg.): *Ubicomp* Bd. 2864, Springer, 2003, S. 73–89. – URL http://dblp.uni-trier.de/db/conf/huc/ubicomp2003.html#PattersonLFK03. – Zugriffsdatum: 13.12.2011. – ISBN 3-540-20301-X

[Pearl 1988] PEARL, Judea: *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1988. – ISBN 1558604790

[Plugge u. a. 2010] PLUGGE, Eelco ; MEMBREY, Peter ; HAWKINS, Tim: *The Definitive Guide to Mongodb: The Nosql Database for Cloud and Desktop Computing*. Apress, 2010. – 328 S. – ISBN 978-1430230519

[Prensky 2001] PRENSKY, Marc: Digital Natives, Digital Immigrants. In: *On the Horizon* 9 (2001), Oktober, Nr. 5. – URL http://www.albertomattiacci.it/docs/did/Digital_Natives_Digital_Immigrants.pdf. – Zugriffsdatum: 13.12.2011

[Rahimi und Vogt 2011] RAHIMI, Mohammad A. ; VOGT, Matthias: Seamless Interaction - Natürliche Interaktionen in Smart Living Umgebungen / Hochschule für Angewandte Wissenschaften Hamburg. URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/rahimi-vogt.pdf. – Zugriffsdatum: 13.12.2011, 2011. – Masterarbeit

[Ranganathan u. a. 2004] RANGANATHAN, Anand ; AL-MUHTADI, Jalal ; CAMPBELL, Roy H.: Reasoning about Uncertain Contexts in Pervasive Computing Environments. In: *IEEE Pervasive Computing* 3 (2004), S. 62–70. – ISSN 1536-1268

[Ranganathan und Campbell 2003] RANGANATHAN, Anand ; CAMPBELL, Roy H.: A Middleware for Context-Aware Agents in Ubiquitous Computing Environments. In: ENDLER, Markus (Hrsg.) ; SCHMIDT, Douglas C. (Hrsg.): *Middleware* Bd. 2672, Springer, 2003, S. 143–161. – URL http://dblp.uni-trier.de/db/conf/middleware/middleware2003.html#RanganathanC03. – Zugriffsdatum: 13.12.2011. – ISBN 3-540-40317-5

[Rizou u. a. 2010] RIZOU, Stamatia ; HÄUSSERMANN, Kai ; DÜRR, Frank ; CIPRIANI, Nazario ; ROTHERMEL, Kurt: A System for Distributed Context Reasoning. In: *Proceedings of the 2010 Sixth International Conference on Autonomic and Autonomous Systems*. Washington, DC, USA : IEEE Computer Society, 2010 (ICAS '10), S. 84–89. – URL http://dx.doi.org/10.1109/ICAS.2010.21. – Zugriffsdatum: 13.12.2011. – ISBN 978-0-7695-3970-6

[Rowstron und Druschel 2001] ROWSTRON, Antony ; DRUSCHEL, Peter: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In: GUERRAOUI, Rachid (Hrsg.): *Middleware 2001* Bd. 2218. Springer Berlin / Heidelberg, 2001, S. 329–350. – URL http://dx.doi.org/10.1007/3-540-45518-3_18. – Zugriffsdatum: 13.12.2011. – 10.1007/3-540-45518-3_18. – ISBN 978-3-540-42800-8

[Ruser und León 2007] RUSER, Heinrich ; LEÓN, Fernando P.: Informationsfusion - Eine Übersicht. In: *tm - Technisches Messen* 74 (2007), Nr. 3/2007, S. 93–102. – URL http://www.atypon-link.com/OLD/doi/abs/10.1524/teme.2007.74.3.93. – Zugriffsdatum: 13.12.2011

[Russell und Norvig 2004] RUSSELL, Stuart J. ; NORVIG, Peter: *Künstliche Intelligenz: Ein moderner Ansatz*. 2. Pearson Education, 2004. – 1328 S. – ISBN 3827370892

[Satyanarayanan 2001] SATYANARAYANAN, M.: Pervasive Computing: Vision and Challenges. In: *IEEE Personal Communications* 8 (2001), S. 10–17

[Schilit u. a. 1994] SCHILIT, B. ; ADAMS, N. ; WANT, R.: Context-Aware Computing Applications. In: *Proceedings of the 1994 First Workshop on Mobile Computing Systems*

and Applications. Washington, DC, USA : IEEE Computer Society, 1994, S. 85–90. – URL http://dl.acm.org/citation.cfm?id=1439278.1440041. – Zugriffsdatum: 13.12.2011. – ISBN 978-0-7695-3451-0

[Schmidt u. a. 1999] SCHMIDT, Albrecht ; AIDOO, Kofi ; TAKALUOMA, Antti ; TUOMELA, Urpo ; VAN LAERHOVEN, Kristof ; VELDE, Walter Van de: Advanced Interaction in Context. In: GELLERSEN, Hans-W. (Hrsg.): *Handheld and Ubiquitous Computing* Bd. 1707. Springer Berlin / Heidelberg, 1999, S. 89–101. – URL http://dx.doi.org/10.1007/3-540-48157-5_10. – Zugriffsdatum: 13.12.2011. – 10.1007/3-540-48157-5_10. – ISBN 978-3-540-66550-2

[Schmidt u. a. 1998] SCHMIDT, Albrecht ; BEIGL, Michael ; GELLERSEN, Hans w.: There is more to Context than Location. In: *Computers and Graphics* 23 (1998), S. 893–901

[Shafer 1976] SHAFER, G.: *A mathematical theory of evidence*. Princeton university press, 1976. – 297 S. – ISBN 0691081751

[Sharp u. a. 2007] SHARP, Helen ; ROGERS, Yvonne ; PREECE, Jenny: *Interaction Design: Beyond Human-Computer Interaction*. 2. John Wiley & Sons, 2007. – 800 S. – ISBN 0470018666

[Snyder u. a. 2011] SNYDER, Bruce ; BOSANAC, Dejan ; DAVIES, Rob: *ActiveMQ in Action*. Manning, 2011. – 382 S. – ISBN 1933988940

[Spies 1993] SPIES, Marcus: *Unsicheres Wissen - Wahrscheinlichkeit, Fuzzy-Logik, neuronale Netze und menschliches Denken*. Spektrum Akademischer Verlag, 1993. – 1–389 S. – ISBN 978-3-86025-006-8

[Steggles und Gschwind 2005] STEGGLES, P ; GSCHWIND, S: *The Ubisense smart space platform*. S. 73–76. In: *Adjunct Proceedings of the Third International Conference on Pervasive Computing* Bd. 191, ACM Press, 2005. – URL http://www.pervasive.ifi.lmu.de/adjunct-proceedings/demo/p073-076.pdf. – Zugriffsdatum: 13.12.2011

[Steinmetz und Wehrle 2005] STEINMETZ, Ralf (Hrsg.) ; WEHRLE, Klaus (Hrsg.): *Peer-to-Peer Systems and Applications*. Bd. *3485*. Springer, 2005. (Lecture Notes in Computer Science). – ISBN 3-540-29192-X

[Stoica u. a. 2001] STOICA, Ion ; MORRIS, Robert ; KARGER, David ; KAASHOEK, M. F. ; BALAKRISHNAN, Hari: Chord: A scalable peer-to-peer lookup service for internet applications. In: *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA : ACM, 2001 (SIGCOMM '01), S. 149–160. – URL http://doi.acm.org/10.1145/383059.383071. – Zugriffsdatum: 13.12.2011. – ISBN 1-58113-411-8

[Streitz u. a. 2007] STREITZ, N.A. ; KAMEAS, A. ; MAVROMMATI, I.: *The disappearing computer: interaction design, system infrastructures and applications for smart environments*. Springer, 2007 (Lecture notes in computer science). – URL http://books.google.com/books?id=1epsB0uvG-wC. – Zugriffsdatum: 13.12.2011. – ISBN 9783540727255

[Streitz u. a. 2005] STREITZ, N.A. ; ROCKER, C. ; PRANTE, T. ; ALPHEN, D. van ; STENZEL, R. ; MAGERKURTH, C.: Designing smart artifacts for smart environments. In: *Computer* 38 (2005), march, Nr. 3, S. 41 – 49. – ISSN 0018-9162

[Strese u. a. 2010] STRESE, Hartmut ; SEIDEL, Uwe ; KNAPE, Thorsten ; BOTTHOF, Alfons: Smart Home in Deutschland / Institut für Innovation und Technik (iit) in der VDI/VDE-IT. URL http://www.iit-berlin.de/veroeffentlichungen/iit-studie-smart-home. – Zugriffsdatum: 13.12.2011, 2010. – Forschungsbericht

[Sutton und Barto 1998] SUTTON, R.S. ; BARTO, A.G.: *Reinforcement learning: an introduction*. MIT Press, 1998 (Adaptive computation and machine learning). – URL http://books.google.de/books?id=CAFR6IBF4xYC. – Zugriffsdatum: 13.12.2011. – ISBN 9780262193986

[Tapia u. a. 2004] TAPIA, Emmanuel M. ; INTILLE, Stephen S. ; LARSON, Kent: Activity Recognition in the Home Using Simple and Ubiquitous Sensors. In: FERSCHA, Alois (Hrsg.) ; MATTERN, Friedemann (Hrsg.): *Pervasive Computing* Bd. 3001. Springer Berlin / Heidelberg, 2004, S. 158–175–175. – URL http://dx.doi.org/10.1007/978-3-540-24646-6_10. – Zugriffsdatum: 13.12.2011

[Tarasewich 2003] TARASEWICH, Peter: TOWARDS A COMPREHENSIVE MODEL OF CONTEXT FOR MOBILE AND WIRELESS COMPUTING / College of Computer and Information Science Northeastern University. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.2854&rep=rep1&type=pdf. – Zugriffsdatum: 13.12.2011, 2003. – Forschungsbericht

[Teske 2010] TESKE, Philipp: Human Fall Detection / Hochschule für Angewandte Wissenschaften Hamburg. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2010-proj1/teske.pdf. – Zugriffsdatum: 13.12.2011, 2010. – Forschungsbericht

[The Apache Software Foundation 2011] THE APACHE SOFTWARE FOUNDATION: *Apache ActiveMQ Architecture*. 2011. – URL http://activemq.apache.org/code-overview.html. – Zugriffsdatum: 13.12.2011

[Valtonen und Vanhala 2009] VALTONEN, M. ; VANHALA, J.: Human tracking using electric fields. In: *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, march 2009, S. 1 –3

[Voskuhl 2009] VOSKUHL, Sören: Bewegungsbasierte Computerinteraktion zur Navigation in Informationsbeständen / Hochschule für Angewandte Wissenschaften Hamburg. 2009. – Bachelorarbeit

[Wang u. a. 2004] WANG, X.H. ; ZHANG, D.Q. ; GU, T. ; PUNG, H.K.: Ontology based context modeling and reasoning using OWL. In: *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, march 2004, S. 18 – 22

[Watzlawick u. a. 2007] WATZLAWICK, P. ; BEAVIN, J.H. ; JACKSON, D.D.: *Menschliche Kommunikation: Formen, Störungen, Paradoxien*. 11. Huber, 2007. – 271 S. – ISBN 9783456844633

[Weiser 1991] WEISER, Mark: *The Computer for the 21st Century*. 1991. – URL http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html. – Zugriffsdatum: 13.12.2011

[Weiser 1999] WEISER, Mark: *Schnittstelle zum Unterbewußtsein*. Telepolis. 1999. – URL http://www.heise.de/tp/artikel/5/5033/1.html. – Zugriffsdatum: 13.12.2011

[Weiser und Brown 1997] WEISER, Mark ; BROWN, John S.: *The coming age of calm technolgy*. S. 75–85. New York, NY, USA : Copernicus, 1997. – URL http://dl.acm.org/citation.cfm?id=504928.504934. – Zugriffsdatum: 13.12.2011. – ISBN 0-38794932-1

[Welch und Bishop 1995] WELCH, Greg ; BISHOP, Gary: An Introduction to the Kalman Filter. Chapel Hill, NC, USA : University of North Carolina at Chapel Hill, 1995. – Forschungsbericht

[Welzl 2005] WELZL, Michael: *Network Congestion Control - Managing Internet Traffic*. John Wiley & Sons Ltd., 2005. – ISBN 978-0-470-02528-4

[Witt 2011] WITT, Kristoffer: *Kontextabhängige multimodale Interaktion mit Schwerpunkt Spracherkennung im Smart-Home Umfeld*, Hochschule für Angewandte Wissenschaften Hamburg, Masterarbeit, 2011

[Yang 2009] YANG, Qiang: Activity recognition: linking low-level sensors to high-level intelligence. In: *Proceedings of the 21st international jont conference on Artifical intelligence*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2009, S. 20–25. – URL http://dl.acm.org/citation.cfm?id=1661445.1661450. – Zugriffsdatum: 13.12.2011

[Zadeh 1965] ZADEH, Lofti A.: Fuzzy Sets. In: *Information and Control* 8 (1965), S. 338–353. – URL http://www-bisc.cs.berkeley.edu/Zadeh-1965.pdf. – Zugriffsdatum: 13.12.2011

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 16. Dezember 2011
Ort, Datum

Unterschrift