Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Seminarausarbeitung

Sören Voskuhl

Entwicklung einer Architektur für Context-Aware Systeme

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

1 Motivation

Durch die steigende Anzahl von Computern in privaten Wohnräumen, realisiert sich die Vision der Allgegenwärtigkeit von Rechenleistung in Haushalten zunehmend. Diese im „Ubiquitous Computing“ eingesetzten Informationstechnologien haben die Aufgabe die Bewohner bei ihren alltäglichen Aufgaben unterstützen. Hierzu ist es häufig hilfreich, wenn die eingesetzten Computer ihr Verhalten dem aktuellen Kontext anpassen können. Anwendungen, die eine solche Fähigkeit besitzen, werden als Context-Aware Systeme bezeichnet. Die verschiedenen Komponenten dieser Systeme erheben hierbei zur Erstellung eines Bezugsrahmens Daten über ihre Umgebung und stellen sie weiteren Teilnehmern zur Verfügung. Aus diesen einzelnen Sensorwerten soll letztlich ein ganzheitlicher Bezugsrahmen erstellt werden, was eine Verknüpfung der einzelnen Daten, eine sog. „Sensorfusion“, erforderlich macht. Eine Schwierigkeit, die an dieser Stelle entsteht, ist das Bereitstellen konsistenter Informationen für alle im Netz beteiligten Komponenten. Aus diesem Grund ist es sinnvoll eine zentrale Einheit einzusetzen, welche die Kontexterstellung und die Kommunikation unter den Teilnehmern koordiniert.
Die Entwicklung einer solchen Middleware für Context-Aware Systeme in intelligenten Wohnräumen ist dabei mit einigen Herausforderungen verbunden. Zum einen besteht die Kontexterkennung aus einer Kaskade von Prozessen, in der fortlaufend neue Daten mit den bisherigen in Verbindung gesetzt werden, mit dem Ziel neues Wissen über die Umgebung zu erlangen. Zum anderen muss die Middleware neue Informationen unter den Netzteilnehmern kommunizieren und ggf. angemessene Reaktionen des Systems initiieren können.
Die grundlegenden Gesichtspunkte bei der Realisierung einer Middleware sollen in der folgenden Ausarbeitung vorgestellt und im Hinblick auf einen Einsatz im „Living Place Hamburg“ untersucht werden. Beim Living Place handelt es sich um ein Projekt der HAW Hamburg, welches bereits seit einigen Jahren in verschiedenen Laboren bestand hat. Seit Anfang 2009 gibt es hier eine im Aufbau befindliche, ca. 130m^2 große Wohnung, in der verschiedene Ansätze des modernen Lebens untersucht werden sollen. In diesem Wohnbereich werden eine Reihe von Context-Aware Systemen zum Einsatz kommen, weshalb eine oben beschriebene Middleware einen wesentlichen Aspekt für die Infrastruktur des Living Place darstellt. Deshalb soll in dieser Seminarausarbeitung auf der Basis bisher erlangter Erfahrungen eine Vorgehensweise für die Masterarbeit erstellt und damit verbundene Herausforderungen vorgestellt werden.

2 Analyse

In diesem Abschnitt sollen die Charakteristiken von Kontextinformationen näher betrachtet sowie die Herausforderungen bei der Erstellung von Context-Aware Systemen erläutert werden.

2.1 Charakteristiken von Kontextinformationen

Kontextinformationen bilden die Grundlage für Context-Aware Systeme, was zur Folge hat, dass deren Eigenschaften eine entscheidende Rolle für die Entwicklung von Applikationen dieser Art spielen. Aus diesem Grund werden im Folgenden einige dieser Charakteristiken vorgestellt, vgl. (Henricksen u. a., 2002).

Zeitliche Eigenschaften
: Kontextinformationen lassen sich anhand ihrer zeitlichen Eigenschaften in die Kategorien „statisch“ und „dynamisch“ einteilen. Statische Informationen beschreiben dabei invariante Aspekte wie den Geburtstag einer Person, während sich dynamische Eigenschaften, wie beispielsweise die Position oder der soziale Status eines Menschen, ändern können. Bei der Verarbeitung von dynamischen Daten ist die Aktualität dieser ein signifikanter Aspekt. Sollten veraltete Werte in die Kontexterstellung einbezogen werden, kann dies zu einer falschen Schlussfolgerung führen und somit eine inkorrekte Reaktion des Systems hervorrufen.

Unvollständigkeit
: Dieser Gesichtspunkt beinhaltet eine Reihe verschiedener Bereiche. Zum einen können die Daten einen falschen Zustand gegenüber der Realität widerspiegeln. Dies würde dazu führen, dass ein falscher Kontext verwendet wird. Des Weiteren können bei einer Sensorfusion inkonsistente Daten vorliegen. Beispielsweise kann ein Sensor melden, dass die aktuelle Raumtemperatur sinkt, während ein weiterer Sensor das Gegenteil behauptet. Außerdem können Werte, die zur Festlegung des Bezugsrahmens einbezogen werden sollen, fehlen.

Repräsentationsformen
: Kontexterstellung bedeutet für Context-Aware Systeme, dass die Werte einzelner Sensoren zu einer Information zusammengefasst werden. Diese kann daraufhin in verschiedenen Formen repräsentiert werden. Hier ist es denkbar, dass eine Applikation direkt mit Sensordaten arbeitet, während eine andere Anwendung eine verarbeitete Form der Daten benötigt. Somit entstehen z.B. für Positionsdaten verschiedene Repräsentationsformen. Einerseits können sie als unverarbeitete Koordinaten vorliegen und auf der anderen Seite bereits so verarbeitet sein, dass der Raum, in dem sich die Person befindet, vorliegt.

2.2 Herausforderungen bei Entwicklung von Context-Aware Systemen

Bei der Realisierung von kontextsensitiven Systemen und der damit einhergehenden Kontexterfassung ergeben sich einige Aspekte, welche für Entwickler relevant sind, die neue Anwendungen dieser Art erstellen und in ein bestehendes System integrieren möchten. In Satyanarayanan (2001) werden einige dieser Herausforderungen aufgegriffen.

Wie wird der Kontext intern repräsentiert und wo wird er gespeichert?
: Zum Empfangen und Veröffentlichen von Kontextinformationen muss sowohl der zentrale Speicherort als auch die benötigten Datenstrukturen zur Verarbeitung der Daten bekannt sein.

Welche Services werden von der Umgebung bereitgestellt?
: In Sensornetzen, wie man sie beispielsweise in intelligenten Wohnbereichen vorfindet, wird eine Vielzahl an Services angeboten. Die Schwierigkeit zur Nutzung dieser Services liegt darin, nützliche Dienste zu identifizieren und die entsprechenden Anbieter zu ermitteln.

Wie häufig sollen Kontextinformationen publiziert oder erfragt werden?
: Der Entwickler muss entscheiden, zu welchem Zeitpunkt die Informationen an weitere Teilnehmer versendet werden sollen. In diesem Zusammenhang ist es wichtig, dass nicht zu viele Daten in das Netzwerk verschickt werden, damit der Overhead möglichst gering bleibt. Auf der anderen Seite sollen den Komponenten keine relevanten Informationen vorenthalten werden. Zwischen diesen Aspekten gilt es ein Gleichgewicht zu schaffen. Gleiches gilt für das konsultieren von Kontextinformationen.

Diese Punkte sollen die Grundlage bilden, Anforderungen für eine Infrastruktur zu ermitteln, die dem Entwickler die Implementation von Context-Aware Systemen erleichtert.

2.3 Anforderungen an eine Middleware

Aufgrund der verteilten Natur und den in 2.2 beschriebenen Schwierigkeiten bei der Entwicklung von kontextsensitiven Applikationen lassen sich einige Anforderungen ableiten, die eine Middleware erfüllen sollte, vgl. Henricksen u. a. (2005). Hierzu gehören u.a. eine hohe Performance, eine Unterstützung heterogener Ressourcen und eine hohe Fehlertoleranz.

Services

Neben den allgemeinen Anforderungen lassen sich aus den Herausforderungen einige Services ableiten, die von der Middleware bereitgestellt werden sollten. In einer lebendigen Architektur, wie sie im Living Place Hamburg zu finden ist, entstehen regelmäßig neue Applikationen die Dienste für weitere Komponenten zur Verfügung stellen. Bei der Integration dieser neuen Anwendungen liegt eine grundlegende Aufgabe darin, bisher installierte Services und Geräte zu identifizieren. Aus diesem Grund soll die bereitgestellte Middleware ein „Service Discovery Protocol“ (SDP) anbieten (Zhu u. a., 2005). Ein solches Protokoll dient dazu Geräte und Dienste in einem Netzwerk zu suchen, zu konfigurieren und mit ihnen zu kommunizieren. Das primäre Ziel dieses Protokolls liegt somit darin, die Heterogenität der Hard-, Software und Netzwerkprotokolle aufzulösen. Mit dem Ziel einen bestimmten Service zu finden, gibt der Client in einem Discovery Prozess den Namen oder Attribute an, die den gesuchten Dienst spezifizieren und erhält daraufhin eine entsprechende Antwort über das Protokoll. Um die Funktionen von Services im Netzwerk bekannt zu geben, arbeiten viele SDP mit einem sog. „Template-Based-Ansatz“. Dieses Verfahren legt ein Format fest, wie Namen und Attribute der Dienste definiert werden müssen. Außerdem bietet dieser Ansatz eine Reihe allgemeiner Attribute an, mit Hilfe derer neue Services beschrieben werden können.

Eine weitere wesentliche Funktion der hier vorgestellten Middleware stellt das „Context Reasoning“ dar. Das Ziel dieser Teilkomponente besteht darin, auf der Grundlage der gesammelten Kontextinformationen neues Wissen über den Bezugsrahmen zu generieren. Zur Realisierung dieser Anwendungen stehen verschiedene Verfahren zur Verfügung, wobei in dieser Ausarbeitung der Fokus auf das sog. „Rule Based Reasoning“ gelegt wird. Eine solche regelbasierte Vorgehensweise lässt sich dabei unter Anwendung eines „Event-Control-Action Architecture Pattern“ (ECA) umsetzen, vgl. (Daniele u. a., 2007). Dieses aus drei Elementen bestehende Muster enthält zum einen das Event-Modul, welches dazu dient die Kontextdaten zu erfassen, entsprechend zu verarbeiten und es an das Control-Modul weiterzuleiten. Letzteres enthält die „Condition Rules“ in der Form „*if <condition> then <action>*“. Hier werden somit die Situationen definiert, in denen bestimmte Aktionen des

Systems erlaubt bzw. notwendig sind. Darüber hinaus initiiert es die Reaktionen der Applikationen und beschreibt hierdurch das Verhalten des Gesamtsystems. Die vom Control-Modul veranlassten Aktionen werden daraufhin vom Action-Modul als Antwort auf die Kontextveränderung durchgeführt.

Zur Realisierung einer „Context Reasoning“-Komponente ist es essentiell, dass die Möglichkeit besteht, auf vorherige Informationen zurückzugreifen. Zum einen werden diese Daten benötigt, um anhand vergangener Kontexte Regeln spezifizieren zu können. Zum anderen werden in diesem Verfahren die neuen Nachrichten mit vorangegangenen in Verbindung gesetzt, mit dem Ziel Schlussfolgerungen daraus zu ziehen.
Diese beiden Aspekte machen den Einsatz eines Nachrichtenspeichers unumgänglich. Außerdem wird eine solche Persistenzschicht für alle Anwendungen ein bedeutender Bestandteil sein, die mit Veränderungen von Werten, wie etwa einer Temperaturveränderung, arbeiten.

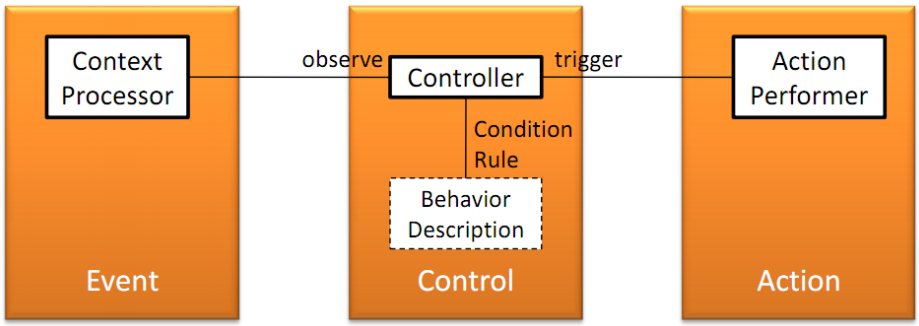

Abbildung 2.1: „Event-Control-Action Architecture Pattern“, nachempfunden aus (Daniele u. a., 2007)

3 Vorgehensbeschreibung

3.1 Ziele der Masterarbeit

Das Ziel der Arbeit liegt darin, eine Architektur für die kontextsensitiven Anwendungen im Living Place Hamburg zu entwickeln. Hierzu wird ein iterativer Entwicklungsprozess erforderlich sein, in dem die einzelnen Teilsysteme realisiert und in einem weiteren Schritt in die bestehende Architektur eingebunden werden. Dabei wird der Fokus auf den im Abschnitt 2.3 vorgestellten Komponenten liegen. Zur konkreten Kontexterstellung wird darüber hinaus ein „Context Interpreter“ eingesetzt. Dieser kann Datenbankabfragen auf der MongoDB (siehe 3.1) durchführen, um erforderliche Kontextinformation zu erlangen, mit der Absicht diese Daten mit der „regelbasierten Reasoning“-Komponente zu untersuchen und ggf. neues Kontextwissen zu erstellen. An dieser Stelle ist es zudem sinnvoll eine Ontologie einzusetzen, in die aktuelle Kontexte gespeichert werden können. Diese Komponente wird jedoch in einem weiteren Teilprojekt diskutiert und erarbeitet, siehe (Ellenberg, 2010).
Eine Darstellung der Beziehungen unter den Komponenten liefert die Abbildung 3.1.

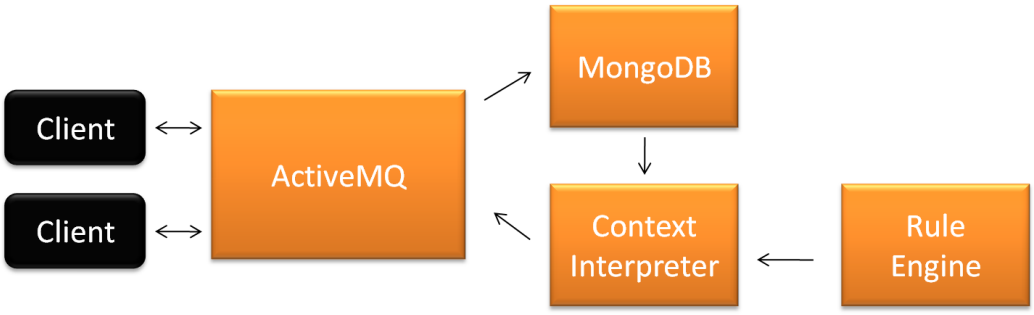

Abbildung 3.1: Übersicht der Komponenten für die Middleware des Living Place

Da die Handhabung der Komplexität einer verteilten Anwendung eine große Herausforderung darstellt, gilt es dem Entwickler eine möglichst große Hilfestellungen bei der Realisierung von neuen kontextabhängigen Applikationen zu bieten. In diesem Zusammenhang soll ein transparenter Zugriff auf die bereitgestellten Dienste der Middleware gewährleistet werden.

Nach der Integration der Infrastruktur in den Living Place soll evaluiert werden, ob sie den beschriebenen Anforderungen gerecht wird. Zu diesem Zweck wird ein Szenario implementiert (siehe 3.2.2), mit Hilfe dessen verschiedene Dienste, wie beispielsweise die Kommunikationsschnittstelle und die Kontextinterpretation der Middleware auf ihre Verwendbarkeit untersucht werden.
Neben der Definition konkreter Ziele der Masterarbeit ist die Eingrenzung des Themas ein weiterer Gesichtspunkt in der Vorbereitung. Aus diesem Grund werden im folgenden Abschnitt einige Aspekte, welche im ersten Schritt nicht implementiert werden sollen, diskutiert.

Abgrenzung

Vorerst ist es nicht geplant Lernverfahren zur semantischen Interpretation der Informationen zu entwickeln. Hier wird einführend der Fokus auf eigens vordefinierte Präferenzen der Bewohner gesetzt. Das bedeutet, dass die Person eine Möglichkeit erhält zu wählen, in welcher Situation welche Reaktion des Systems gewünscht wird. Außerdem sollen die Regeln der „Context-Reasoning“-Komponente im ersten Schritt vom Anwender/Entwickler spezifiziert und nicht automatisch erlernt werden.
Des Weiteren wird die hier entwickelte Middleware Nachrichten nach dem „Best-Effort“-Verfahren ausliefern. Dies beinhaltet, dass vorerst keine „Quality of Service“-Dienste implementiert werden, sondern die Daten schnellstmöglich und im Rahmen der zur Verfügung stehenden Ressourcen verarbeitet werden.

Schlussfolgerungen aus Vorarbeiten

Im Zuge der Bachelorarbeit (Voskuhl, 2009) und in Voskuhl (2010b) konnten bereits einige Erfahrungen in den Bereichen des „Ubiquitous Computing“ und der „Context Awareness“ in intelligenten Wohnumgebungen gesammelt werden. Während die Bachelorarbeit sich mit intuitiven Interaktionsmodi zwischen Mensch und Computer befasste, hat sich Voskuhl (2010b) intensiv mit dem „Blackboard“ als zentrale Kommunikationseinheit in einer Architektur für intelligente Wohnungen beschäftigt. Dabei wurde insbesondere die Interaktion zwischen den Sensoren („Knowledge Sources“) und dem Blackboard sowie eine mögliche Vorgehensweise bei der Kontexterkennung mit Hilfe der vorgestellten Kommunikationsschnittstelle betrachtet. Als Grundlage diente dazu die Blackboard-Architektur des Spracherkennungssystems „Hearsay-II“ aus Erman u. a. (1980).

Darauf aufbauend wurden in Voskuhl (2010a) verschiedene Ansätze für Architekturen des „Living Place Hamburg“ vorgestellt, in dem vergleichbare Forschungsprojekte im Hinblick auf ihre Middleware untersucht wurden. Die Bewertung dieser Ansätze machte deutlich, dass

es keine einheitliche Lösung zur Entwicklung von Architekturen für Context-Aware Systeme gibt, was eine Betrachtung weiterer Arbeiten sowie eigene Prototypen notwendig machte.

Mittels der bisherigen Erkenntnisse konnte daraufhin in Otto und Voskuhl (2010) mit der Kommunikationsschnittstelle eine grundlegende Komponente der Architektur bereitgestellt werden. Nachdem sich das bisherige Blackboard, der „IROS Eventheap“, aufgrund einiger Performance-Tests für den Living Place als ungeeignet herausgestellt hatte, konnte mit dem ActiveMQ[1] ein deutlich leistungsstärkerer Message-Broker zur Verfügung gestellt werden. Weitere Gründe für den Einsatz dieses Systems waren die Zugriffsmöglichkeiten durch diverse Programmiersprachen, die sehr gute Dokumentation sowie die offenen Schnittstellen zur Anpassung des ActiveMQ.

In Otto und Voskuhl (2011) stellte die Persistenz der Nachrichten die über den ActiveMQ versendet werden einen bedeutenden Teilbereich dar, da dieser Service eine entscheidende Rolle für eine Reihe weiterer Komponenten spielt. Umgesetzt wurde die Persistenzschicht mit Hilfe des dokumentenorientierten Datenbankservers „MongoDB“[2]. Hierbei handelt es sich um einen Performance orientierten schemalosen Objekt-Datenspeicher im JSON-Format. Der dokumentenorientierte Ansatz birgt dabei den Vorteil, dass die Nachrichten als einzelne Dokumente gespeichert werden und nicht, wie in einem relationalen Datenbankmanagementsystem, Tabellen mit Spalten definiert werden müssen. Hierdurch entsteht der positive Effekt, dass nicht jede Nachricht im Netzwerk einem festen Schema unterliegen muss, um persistiert zu werden.

3.2 Vorgehen in der Masterarbeit

3.2.1 Realisierung der Architektur

Im folgenden Abschnitt soll sowohl für das Service-Discovery-Protokoll als auch für die „Context-Reasoning“-Komponente eine denkbare Implementierung vorgestellt werden.

Service Discovery

Das SDP soll in der Masterarbeit eine ähnliche Architektur wie das Framework „Jini“ aufweisen. „Jini“ ist eine in Java entwickelte Anwendung und besitzt die Aufgabe, jedem Service im Netzwerk die Möglichkeit zu bieten mit jedem anderen Netzteilnehmer zu kommunizieren. Dabei soll die Interaktion unabhängig von Treibern, Protokollen oder Betriebssystemen

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

funktionieren, vgl. (Harihar und Kurkovsky, 2005). Dieser Grundgedanke soll dazu verwendet werden, ein eigenes Konzept für das „Service Discovery“ im Living Place zu erstellen.
In „Jini“ werden zur Verwirklichung der Funktionen die einzelnen Entitäten in einem Verbund zusammengefasst und über ein sog. Jini-Network verknüpft. Jeder Teilnehmer in diesem Verbund offeriert Services und verwendet Dienste anderer Mitglieder.
Zur dynamischen Netzwerkerstellung bietet Jini „discovery/join“-Protokolle an. Das „Discovery Protokoll“ wird von Clients und Services dazu verwendet einen „Lookup Service“ zu suchen. Sollte der Pfad zum „Lookup Services“ unbekannt sein, sendet der Suchende ein „Discovery“-Paket über ein Multicast-Protokoll ins Netzwerk und wartet auf eine Antwort des Dienstes. Sollte der Pfad bereits bekannt sein, wird das „Discovery Protokoll“ dazu verwendet ein Paket per Unicast-Protokoll an den „Lookup Service“ zu verschicken, mit dem Ziel einen gesuchten Service mit Netzwerk zu identifizieren. Über das „Join Protokoll“ wird spezifiziert wie die Clients sich beim „Lookup Service“ registrieren können.
Der „Lookup Service“ bildet somit die zentrale Einheit eines Verbundes und ist dafür zuständig, neben den Objekt-Referenzen auch die Service-Beschreibung und ggf. spezielle Attribute über die Dienste zu speichern. Er ist somit für die einfache Ortung von Objekten innerhalb des Jini-Verbundes verantwortlich.
Die Struktur dieses Frameworks soll im Living Place dazu dienen, eine eigene Anwendung zu implementieren, welche die große Anzahl an Anwendungen verwalten kann und neuen Diensten einen Mechanismus liefert, bestehende Services zu suchen und auf sich selbst aufmerksam zu machen.

Context Reasoning

In diesem Abschnitt soll eine mögliche Realisierung der regelbasierten „Reasoning“-Komponente präsentiert werden. Dieses „Rule Based System“ soll im Wesentlichen aus drei Basis-Komponenten bestehen. Neben einer „Rule Base“, welche eine Liste der bekannten Regeln enthält, gibt es einen „Rule Interpreter“. Dieser initiiert Maßnahmen aufgrund von erhaltenen Nachrichten über den ActiveMQ und der Rule Base. Der Interpreter arbeitet hierzu mit einem sog. „recognize-act cycle“ (Gupta u. a., 1986), welcher sich in drei Schritte unterteilt:

1. **Match:** In der ersten Phase wird jede Nachricht, welche über das Blackboard versendet wird, dahingehend geprüft, ob der Inhalt mit einem Eintrag in der Rule Base übereinstimmt. Darauf folgend erhält man ein sog. „Conflict Set“, welches Instanzen aller erfüllten „if-then-Regeln“ enthält.
2. **Conflict Resolution:** In dieser Phase wird eine „if-then-Regel“ aus dem „Conflict Set“ ausgewählt.
3. **Act:** In diesem dritten Schritt werden die Aktionen der gewählten Regel initiiert, indem eine entsprechende Nachricht an den ActiveMQ gesendet wird.

Die dritte Komponente bildet der „Regel Editor“. Dieser dient dazu neue Regeln einzufügen oder obsolete Regeln zu löschen. Er bietet somit die Möglichkeit direkt auf der „Rule Base“ zu arbeiten.
Neben der Funktion des „Regel Editors“ wäre es denkbar, dass neue Regeln zusätzlich mit Hilfe des Blackboards eingetragen werden können. In diesem Zusammenhang könnten Anwendungen in ihren Nachrichten in Form von Metadaten gewisse Regeln angeben, die für sie eine entscheidende Rolle spielen. Beispielsweise könnte eine Applikation eine Benachrichtigung anfordern, sobald sich die aktuelle Raumtemperatur um 5°C erhöht.
Die Interaktion der in der „Context Reasoning“-Komponente eingesetzten Elemente und die Kommunikation mit dem ActiveMQ wird auf der Abbildung 3.2 präsentiert.

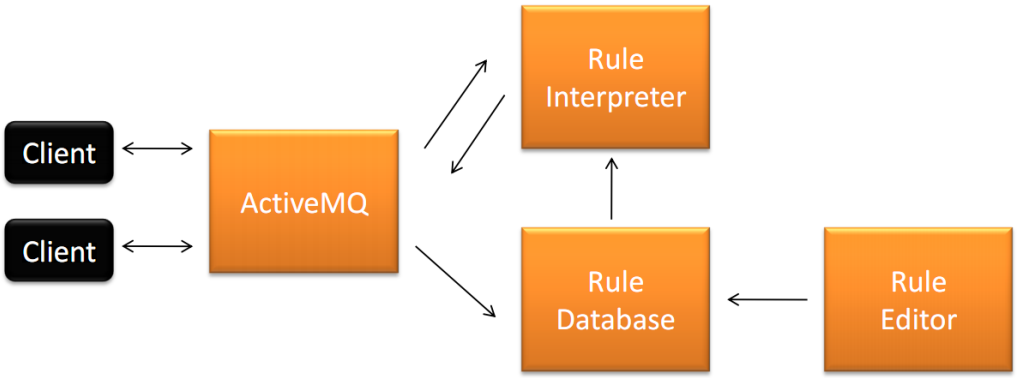

Abbildung 3.2: Kommunikation zwischen der „Context Reasoning“-Komponente und dem ActiveMQ

3.2.2 Entwicklung des Szenarios

Um das Zusammenspiel der verschiedenen Komponenten zu betrachten und die Bedeutung einer Middleware weiter zu verdeutlichen wird im Folgenden ein Szenario präsentiert, welches im Living Place realisiert werden soll.
An dieser Stelle wird eine Zusammenfassung des bereits in Voskuhl (2010b) vorgestellten Szenario verwendet. Hierin wird beschrieben, wie ein Wohnbereich reagieren könnte, wenn eine Person eine Wohnung bzw. einen Bereich darin betritt. Mögliche Reaktionen wären in diesem Fall, die zur Stimmung passende Musik zu spielen oder eine individuelle Begrüßung erfolgen zu lassen. Des Weiteren soll sich die gesamte Umgebung den Präferenzen des Bewohners anpassen sowie die Termine und persönlichen Nachrichten auf dem nächsten freien Abspielgerät präsentiert werden.

Zur Umsetzung des Szenarios werden verschiedene Kontextinformationen benötigt, deren Erfassung den Einsatz unterschiedlicher Sensortechnologien erfordert. Hierzu bietet sich u.a. das Echtzeit Ortungssystem der Firma Ubisense[3] an, durch welches Personen und Gegenstände geortet sowie authentifiziert werden können. Des Weiteren sollen Bewegungssensoren zum Einsatz kommen, anhand derer ein Benutzer mit Hilfe von Gesten Reaktionen des Systems initiieren oder beenden kann. Außerdem werden in diesem Szenario verschiedene Programmiersprachen zum Einsatz kommen, um die einzelnen Geräte, die auf den Bewohner reagieren sollen, anzusteuern.

3.3 Chancen und Risiken

Durch eine vollständige Entwicklung der Komponenten und einer erfolgreichen Zusammenarbeit mit einigen weiteren Projekten kann eine zuverlässige Architektur für das Living Place Hamburg gewährleistet werden. Durch das zu realisierende Szenario kann diese Middleware evaluiert werden, sodass sichergestellt wird, dass auf Basis dieser Infrastruktur weitere Applikationen erstellt werden können und die Architektur somit für den längerfristigen Einsatz geeignet ist. Außerdem kann durch das Szenario ein Wohnraum geschaffen werden, der sich den Präferenzen des Bewohners angleicht.

Die Risiken für die Middleware bestehen darin, dass die bisher gewählten Kommunikationskomponenten dem hohen Nachrichtenaufkommen nicht standhalten können und sich zu hohe Latenzen in der Auslieferung ergeben. Außerdem könnte deutlich werden, dass das SDP nicht in der Lage ist, die Heterogenität einiger Ressourcen zu überbrücken. Dies hätte zur Folge, dass einige Geräte nicht am Netzwerk teilnehmen können, bevor nicht die entsprechenden Schnittstellen zur Verfügung stehen. Für die Entwicklung des Szenarios ist es denkbar, dass der Benutzer sich von der Anzahl eingesetzter Computer „durchleuchtet“ fühlt. Hier gilt es diese Empfindung frühzeitig mit Hilfe von Usability-Tests zu erkennen und zu beseitigen.

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

4 Zusammenfassung

Die Realisierung von Context-Aware Systemen birgt einige Herausforderungen die vom Entwickler überwunden werden müssen. Diese liegen im Wesentlichen in der Koordination der eingesetzten Informationstechnologien und dem Umgang mit den Kontextdaten, welche einige besondere Charakteristiken aufweisen. Einige dieser Eigenschaften wurden im Abschnitt 2.1 vorgestellt.
Mit dem Ziel dem Programmierer möglichst viele Schwierigkeiten abzunehmen, soll eine Middleware bereitgestellt werden, welche bereits elementare Dienste zur Verfügung stellt. Hierzu gehören neben einer zuverlässigen Kommunikationsschnittstelle zusätzlich Services wie ein „Service Discovery Protocol“ oder eine „Context Reasoning“-Komponente. Letztere soll die Aufgabe der Erstellung des Bezugsrahmens übernehmen und die ermittelten Kontexte den weiteren Anwendungen im System zur Verfügung stellen. Auf Basis dieser Architektur soll in einem weiteren Schritt ein Szenario implementiert werden das sich den Präferenzen des Bewohners anpassen kann. Dabei soll die Fragestellung wie die Umgebung reagieren könnte wenn eine Person die Wohnung oder einen Bereich darin betritt im Vordergrund stehen. Die Implementation dieses Anwendungsfalles soll außerdem dazu dienen, die Middleware zu evaluieren und aufzuzeigen, dass mit Hilfe dieser Architektur eine einfache Integration von kontextsensitiven Systemen in das bestehende System möglich ist.
Dabei stellt die Infrastruktur des Living Place Hamburg mit seiner Vielfalt an zur Verfügung stehenden Technologien und Interaktionsmöglichkeiten einen idealen Rahmen für Context-Aware Systeme und eine unterstützende Middleware dar.

Literaturverzeichnis

[Daniele u. a. 2007] DANIELE, Laura ; COSTA, Patrícia D. ; PIRES, Luís F.: Towards a rule-based approach for context-aware applications. In: *Proceedings of the 13th open European summer school and IFIP TC6.6 conference on Dependable and adaptable networks and services*. Berlin, Heidelberg : Springer-Verlag, 2007 (EUNICE'07), S. 33–43. – URL http://portal.acm.org/citation.cfm?id=1779813.1779820. – ISBN 3-540-73529-1, 978-3-540-73529-8

[Ellenberg 2010] ELLENBERG, Jens: Vorarbeiten für den Wecker 2.0 / Hochschule für Angewandte Wissenschaften Hamburg. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2010-proj1/ellenberg.pdf, 2010. – 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

[Gupta u. a. 1986] GUPTA, A. ; FORGY, C. ; NEWELL, A. ; WEDIG, R.: Parallel algorithms and architectures for rule-based systems. In: *Proceedings of the 13th annual international symposium on Computer architecture*. Los Alamitos, CA, USA : IEEE Computer Society Press, 1986 (ISCA '86), S. 28–37. – URL http://dx.doi.org/10.1145/17407.17360. – ISBN 0-8186-0719-X

[Harihar und Kurkovsky 2005] HARIHAR, Karthik ; KURKOVSKY, Stan: Using Jini to enable pervasive computing environments. In: *Proceedings of the 43rd annual Southeast regional conference - Volume 1*. New York, NY, USA : ACM, 2005 (ACM-SE 43), S. 188–193. – URL http://doi.acm.org/10.1145/1167350.1167407. – ISBN 1-59593-059-0

[Henricksen u. a. 2005] HENRICKSEN, Karen ; INDULSKA, Jadwiga ; MCFADDEN, Ted ; BALASUBRAMANIAM, Sasitharan: Middleware for Distributed Context-Aware Systems. In: *International Symposium on Distributed Objects and Applications (DOA*, Springer, 2005, S. 846–863

[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://portal.acm.org/citation.cfm?id=646867.706693. – ISBN 3-540-44060-7

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

[Otto und Voskuhl 2011] OTTO, Kjell ; VOSKUHL, Soeren: Weiterentwicklung der Architektur des Living Place Hamburg. (2011)

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

[Voskuhl 2009] VOSKUHL, Sören: Bewegungsbasierte Computerinteraktion zur Navigation in Informationsbeständen / Hochschule für Angewandte Wissenschaften Hamburg. URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/bachelor/voskuhl.pdf, 2009. – Forschungsbericht

[Voskuhl 2010a] VOSKUHL, Sören: Architekturen für Context-Aware Systeme / Hochschule für Angewandte Wissenschaften Hamburg. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2010-aw2/Voskuhl/bericht.pdf, 2010. – Forschungsbericht

[Voskuhl 2010b] VOSKUHL, Sören: Bereitstellung einer Sensorwolke / Hochschule für Angewandte Wissenschaften Hamburg. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master09-10-aw1/Voskuhl/bericht.pdf, 2010. – Forschungsbericht

[Zhu u. a. 2005] ZHU, Fen ; MUTKA, Matt W. ; NI, Lionel M.: Service Discovery in Pervasive Computing Environments. In: *IEEE Pervasive Computing* 4 (2005), S. 81–90. – ISSN 1536-1268