iFlat - Eine dienstorientierte Architektur für intelligente Räume[1]

Sven Stegelmeier, HAW-Hamburg, Germany, stegel_s@informatik.haw-hamburg.de
Piotr Wendt, HAW-Hamburg, Germany, piotr.wendt@informatik.haw-hamburg.de
Kai von Luck, HAW-Hamburg, Germany, luck@informatik.haw-hamburg.de

Kurzfassung

Im Rahmen der Ambient Intelligence Labore der HAW Hamburg wird seit ca. 2 Jahren ein Modell einer intelligenten Wohnung (iFlat) untersucht und weiterentwickelt. Kern dieser Aktivitäten bildet eine Software Struktur, die als lose gekoppelte Verbindung einzelner Prozesse realisiert wird. Die Kommunikation verwendet eine Modifikation eines Blackboard-Systems der Univ. Stanford (iROS) als Nachrichtenmakler und Jade als Anwendungsprotokoll. Die einzelnen Komponenten des iFlat sind als intelligente Subsysteme auf jeweils eigenständigen Rechnern ausgeprägt, so dass die zentrale Steuerung auf die Modellierung von szenarienorientierten Abläufen beschränkt ist. Durch diesen Ansatz ist eine einfache Integration neuer Komponenten in bestehende Umgebungen gewährleistet.

Die Entwicklung des iFlat orientiert sich an unterschiedlichen konkreten Szenarien, die eine technologiebasierte Weiterentwicklung der Überlegungen von M. Weiser von 1991 darstellen. So kann Sal eine interessengeleitete Information auf Basis von Eyetracking-Analysen angeboten bekommen (TV vs. Newsticker), das Handy sich ein freies Display für die ungelesenen SMS suchen, die Türgegensprechanlage mit dem TV-Receiver um das Display konkurrieren. Ein mit RFID ausgestatteter Kühlschrank kann mit der PIM-Anwendung des Handys über die Gestaltung des Abendessens konferieren und Einkaufsaufträge an den Bewohner schicken. Ein Fernsehangebot macht nur Sinn, wenn ein Bewohner im Raum ist, wobei diese Kontextinformation aktuell sowohl über Eyetracker als auch über Indoor-Lokationssysteme erfolgt.

Szenarienbasierte Templates, wie wir diese verstehen, beschreiben die Details der möglichen Interaktionen zwischen dem Bewohner und dem Intelligenten Raum. Dabei werden die, zu Erreichung entsprechenden Ziele, notwendigen Zwischenschritte abstrahiert. Diese aufgabenorientierte, technologieneutrale Spezifizierung der Anwendungslogik einer Interaktion erlaubt uns die darunterliegenden Geräte und damit verbundenen Treiber nach belieben auszutauchen. Damit können zur Erreichung eines Ziels ein oder mehrere Geräte herangezogen werden, wobei die technischen Details eine untergeordnete Rolle spielen. Zum anderen kann der Austausch von erforderlichen Informationen zwischen den einzelnen Geräten proaktiv und unabhängig von der Aktivität erfolgen.

Intelligenter Räume werden somit als eine Zusammenstellung unterschiedlicher Dienste betrachtet, welche auf den Kontexten der individuellen Bewohner basieren. Ein Dienst ist hierbei ein in sich abgeschlossenes Softwaresystem welches in der Regel als Treiber für eine Hardware fungiert. Die Kontexterkennung ist ein zentraler Bestandteil der Architektur und wird durch eine schichtweise Verarbeitung von Rohsensordaten erreicht. Auf oberster Ebene gelangen Kontextdaten an eine Laufzeitumgebung. Auf Basis dieser Daten und einer definierten Interaktionslogik werden sämtliche installierte Anwendungen des intelligenten Hauses ausgeführt. Eine Editorkomponente ermöglicht durch das Verknüpfen von unterschiedlichen Diensten die Erstellung von Anwendungen für das intelligente Haus. Diese Vorgehensweise fällt in das Paradigma der Dienstorientierung und wird schon seit Jahren in der Geschäftswelt erfolgreich angewendet.

Die Konzepte der allgegenwärtigen Computer in intelligenten Räumen können unserer Meinung nach nur funktionieren, wenn die Interaktion in solchen intelligenten Räumen über eine geeignete kontextabhängige Auswahl von Interaktionsmodalitäten erfolgt. Im iFlat werden insbesondere auch Ergebnisse der Entwicklung nahtloser Interaktionstechniken des GamecityLabs Hamburg herangezogen, das als gemeinsame Aktivität der Informatik und der Medientechnik von der Wirtschaftsbehörde finanziert wird.

1 Einleitung

Softwarearchitekturen stehen seid über einem Jahrzehnt im Fokus der professionellen Softwareentwicklung. Der stark zunehmenden Komplexität von Geschäftsanwendungen musste strukturiert entgegengewirkt werden. Aus einfachen Schichtenmodellen kristallisierten sich Service orientierte Architekturen heraus, welche wiederum auf komplette Anwendungslandschaften eines Unternehmens abzielen. Damit wird eine lose Kopplung zwischen heterogenen Systeme angestrebt. Diese wiederum führt zur einer erhöhten Dynamik des Gesamtsystems und damit der Vereinfa-

[1] Published in Proc. of the VDE 2. Ambient Assisted Living Kongress mit Ausstellung, 27.01. – 28.01.2009 in Berlin

chung der Anpassung der bestehenden Systeme bzw. Integration von Neusystemen.
Parallel beginnt die IT langsam ihren Einzug in die Häuser und Wohnungen und damit auch in das Privatleben ihrer Bewohner zu haben. Seit den frühen Neunzigern [1] bis heute sind Ansätze entwickelt worden, welche unsere Wohnräume zunehmend in intelligente Räume verwandeln. Derzeit fehlt jedoch ein einheitlicher Ansatz für solche Systeme, wie er sich langsam in dem industriellen Bereich abzeichnet. Folglich sind solche Systeme Insellösungen und damit nicht handhabbar. Eine Architektur für eine Infrastruktur eines intelligenten Raumes würde die Komplexität der gesamten Anwendungslandschaft entschärfen und formalisieren. Dies würde deutlich die Entwicklung und Integration spezialisierter Teilsysteme in diesem Bereich vereinfachen, wie das in industriellen Bereichen zu beobachten ist.
Angelehnt an den aus der Geschäftswelt gesammelten Erkenntnissen können dort entwickelte Konzepte der Serviceorientierung dazu beitragen, eine solche Architektur zu entwerfen. Dabei muss berücksichtigt werden, dass die beiden Bereiche unterschiedliche Zielsetzungen verfolgen. Die hier vorgeschlagene Architektur muss die Geschäftsprozess-orientierte Sicht verlassen und zu einer Aktivitätsorientierten weiterentwickelt werden, welche den Menschen in seinen Alltagsaufgaben unterstützt. Darauf aufbauende Anwendungen müssen ein intelligentes Verhalten erkennbar machen. Dabei spielt der Kontext des Benutzers eine wesentliche Rolle. So sollen beispielsweise Informationen über diverse Ausgabekanäle wie Ton und Bild auf Basis der Position des Benutzers zur Verfügung gestellt werden. Die Erfassung solcher Kontextdaten stellt dabei die größte Herausforderung dar.
Ein weiterer essenzieller Aspekt eines intelligenten Raumes ist die nahtlose Interaktion. Anwendungen intelligenter Häuser sollen sich nahtlos in das Leben ihrer Bewohner integrieren. In unserem Ansatz von Context Aware Computing ist der Mensch als Bewohner des intelligenten Hauses Teil des Kontextes, über die direkte Informationen wie zum Beispiel Lokation, als auch indirekte Informationen über übliche Bodymonitoring Techniken weitergereicht werden. Zusätzlich werden Intentionen mit Hilfe von Indikatoren auf Basis von Eyetrackern u.ä Technologien ausgewertet. Analog der Arbeiten von Jeff Pierce et al. über persönliche Informationsumgebungen [2] nutzen wir ebenfalls in vielen Szenarien das Mobilphone als Fernbedienung. Hierzu ist die Anpassung bestehender Service orientierter Konzepte und Realisierungen erforderlich.
Dieses Papier beschreibt eine dienstorientierte Architektur für interaktive Räume. Zunächst wird in Abschnitt 2 ein konzeptioneller Überblick gegeben. Danach werden in den Unterabschnitten 2.1 bis 2.3 die unterschiedlichen Bausteine detailliert erläutert. Im Anschluss skizziert Abschnitt 3 ein Beispielszenario sowie deren Umsetzung in der vorgestellten Architektur. Am Schluss werden die vorgestellten Konzepte in Abschnitt 4 noch einmal zusammengefasst.

2 Konzeptionelle Architektur

Bis heute wurden unterschiedliche Architekturansätze für Softwaresysteme in intelligenten Räumen entwickelt (siehe auch [3]. Eine wesentliche Rolle dabei spielt die Möglichkeit zur Integration neuer Softwarekomponenten in ein bereits bestehendes intelligentes Haus. Statt einzelner unabhängiger Teil- und Subsysteme zu optimieren, geht unser Ansatz in Richtung einer Integration sehr atomarer Basiskomponenten als Aggeration von Services.
Eine Komposition solcher Services zu aufgabenspezifischen Anwendungen erfolgt analog zu SOA-Architekturen als Workflow-Orchestrierung. Eine so gestaltete Architektur fördert eine lose gekoppelte Hard- und Software Erstellung, wobei hierfür vorausgesetztes Wissen über den Einsatzkontext der Komponente in den Hintergrund tritt. Die folgende Abbildung skizziert eine konzeptionelle Sicht der hier vorgestellten Architektur.

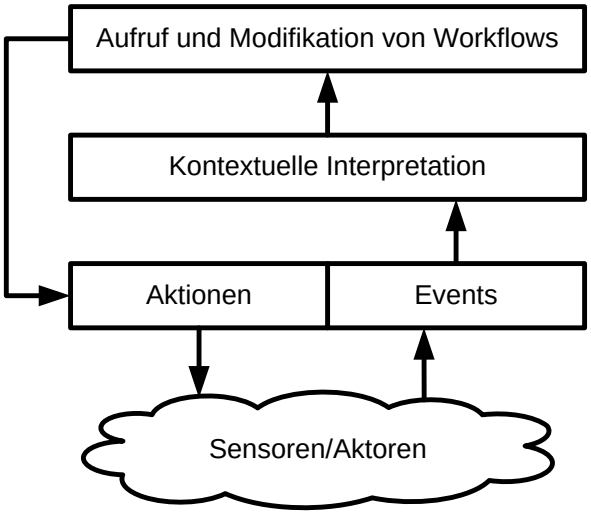

Bild 1 Konzeptionelle Architektur

Diese Architektur bildet einen geschlossenen Kreislauf und ähnelt dem Architekturmuster der Pipes & Filter. Drei Schichten bilden die Kernbestandteile unserer Architektur, die wiederum aus unterschiedlichen Hard- und Softwarekomponenten besteht. Auf der unteren Schicht bilden kontextbasierte Dienste die eigentlichen Bestandteile des intelligenten Hauses. Jedes dieser Komponenten wird durch Treiber angesteuert, welche konkret als Java-Agenten durch das Framework JADE [4] realisiert werden.
Aufgetretene Ereignisse werden in der mittleren Schicht an die Kontexterkennungskomponente übermittelt. Diese hat die Aufgabe, Ereignisse bestimmten Kontexten zuzuordnen und alle Ereignisse kontextspezifisch zu aggregieren. Ziel ist es der Prozess-Engine eine einheitliche Schnittstelle zu bieten und damit die Vielfalt unterschiedlicher Sensoren zu kapseln.
Auf der oberen Schicht befinden sich Workflowmodellierungen.
So induziert das Betreten eines Raumes als Event unter dem Kontext Abwesenheit von Tageslicht den Workflow

„Aufenthalt in einem beleuchteten Zimmer“ u.a. mit den Aktionen „Einschalten der Beleuchtung“, „Erzeugen von angenehmer Raumtemperatur“ und „Durchstellen einkommenden Telefonate auf das Zimmertelefon“.
Die Kommunikation zwischen den Komponenten basiert auf der Blackboard Architektur [5] in seiner Ausprägung iROS [6].

2.1 Services

Die Service Komponente der Architektur besteht aus einer Vielzahl von unterschiedlichen Softwarediensten. Beispielsweise kann der Hersteller eines Location-Tracking Systems durch das Einhalten bestimmter Schnittstellen sein Gerät allen Applikationen des intelligenten Hauses zur Verfügung stellen.
In der Service Komponente werden drei Arten der Kommunikation identifiziert:

- Entgegennahme von Anweisungen der Prozess-Engine
- Kommunikation von Agenten untereinander
- Melden von Ereignissen an die Kontexterkennung

Zudem findet eine technisch bedingte Kommunikation bezüglich der Beschreibung von Diensten statt. Bei der Orchestrierung von Diensten bzw. Datenzuordnung an bestimmte Kontexte werden dienstspezifische Beschreibungsinformationen darüber übermittelt.
Das Auftauchen von Events auf dem Blackboard induziert sowohl deren kontextuelle Interpretation als auch die Zielgenerierung für Agenten auf Basis von eventspezifischen Workflows (s. Abbildung 2). Die eben geschilderten Zusammenhänge in der Services Komponente werden in Abbildung 2 verdeutlicht.

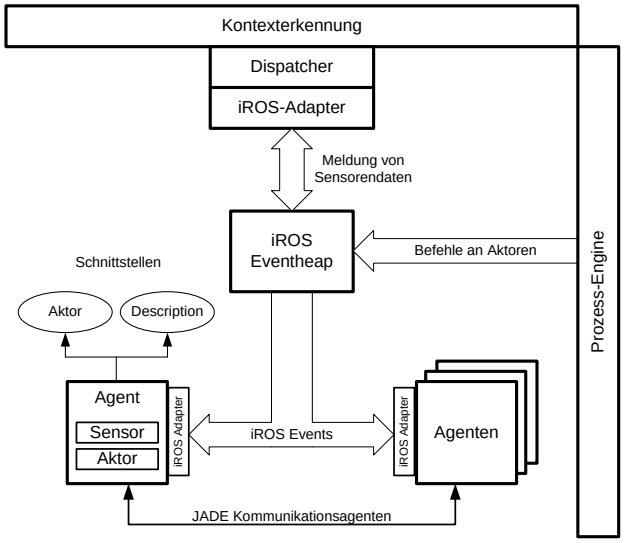

Bild 2 Konzeptionelle Sicht der Service-Komponente

Zentrales Element der Service Schicht ist der iROS-Eventheap. Dieser gehört im engeren Sinne nicht zu der Service Komponente, sondern ist vielmehr als zentraler Kommunikationskanal aller Architekturelemente zu verstehen. Der Eventheap ist über einen iROS-Adapter mit der Kontexterkennung verbunden. Der iROS-Adapter übernimmt die Transformation bestimmter Nachrichten in ein für die Kontexterkennung geeignetes Format. Über diese Verbindung werden hauptsächlich Daten von Sensoren über stattgefundene Ereignisse transportiert.
Auf der anderen Seite ist auch die Prozess-Engine über einen iROS-Adapter (hier nicht dargestellt) mit dem Eventheap verbunden. Hierüber werden allerdings keine Sensorendaten, sondern konkrete Anweisungen an bestimmte Aktoren übermittelt. So können z.B. Steuerbefehle an die Beleuchtung als Reaktion auf das Unterbrechen einer Lichtschranke abgesetzt werden. Hierbei sind die Steuerbefehle Anweisungen an einen Aktor, während die Unterbrechung einer Lichtschranke ein Ereignis darstellt.
Durch die Kapselung von Sensoren und Aktoren mit Hilfe von Agenten ist es möglich, die tatsächliche Heterogenität der involvierten Hard- und Software zu verbergen und eine einheitliche Schnittstelle für die Kontexterkennung sowie die Prozess-Engine anzubieten. Schnittstellen werden in diesem Zusammenhang nicht über konventionelle Konstrukte einer Programmiersprache definiert. Vielmehr werden hierbei Nachrichtenformate formal spezifiziert. Hierdurch können sich die Kontexterkennung und die Prozess-Engine auf die Kommunikation mit unterschiedlichen Agenten vorbereiten. Dabei verwendet die Prozess-Engine die Aktor-Schnittstelle der jeweiligen Agenten, welche die verfügbaren Dienste definiert. Die Kontexterkennung wiederum verwendet die Description Schnittstelle, über welche Informationen über die möglichen Sensorendaten zurückgegeben werden.

2.2 Kontexterkennung

Treten Ereignisse in der Service Komponente auf, werden diese an die Kontexterkennungskomponente weitergeleitet. Die zentrale Aufgabe dieser Komponente besteht darin, die unterschiedlichen Sensordaten geeigneten Kontexten zuzuordnen und in ein für die Prozess-Engine geeignetes Format zu transformieren. Dies ist erforderlich, um eine einheitliche Schnittstelle für die Prozess-Engine anbieten zu können. Lokationsinformationen können über diverse Sensoren wie Lichtschranke, RFIDs, Bilderkennung und Indoor-Positioning Systeme ermittelt werden. Anwendungen, welche von der Prozess-Engine verwaltet werden, interessiert aber nur eine Ortsinformation. So wird durch Sensor Fusion die Heterogenität dieser Sensoren verborgen und nur ein einziger Punkt im 3D Koordinatensystem angeboten. Auf diese Weise können auf Lokation basierende Anwendungen die aktuelle Position eines Einwohners einfach erkennen. Um dies zu erreichen müssen alle Lokationsinformationen aggregiert und in eine einzige überführt werden (Sensor Fusion). Die abschließenden Sensorwerte werden als Kontextinformationen interpretiert (z.B. Bewohner befindet sich im Wohnzimmer). Die folgende Abbildung verdeutlicht die konzeptionelle Sicht der Kontexterkennung.

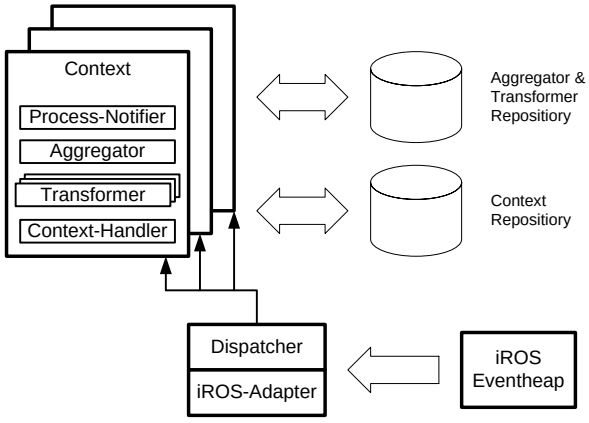

Bild 3 Konzeptionelle Sicht der Kontexterkennung

Wiedererkennbar ist der Eventheap als Kommunikationskanal. Über den iROS-Adapter gelangen Daten zum Dispatcher, welcher die Aufgabe hat den geeigneten Kontext für ein bestimmtes Datum auszusuchen. Eine Zuordnung von Daten zu Kontexten geschieht zur Konfigurationszeit über zuvor erwähnte Editorkomponente. Das Ergebnis wird zusammen mit anderen Daten im Context-Repository gespeichert. Hier werden pro Kontext folgende Informationen gespeichert und abrufbar gemacht:

- Beschreibende Informationen zu einem Kontext (ID, Name, …)
- Liste von Sensoren, deren Daten jeweils zu diesem Kontext gehören
- Einen Aggregator
- Liste von Transformatoren
- Informationen für die Transformation

Die Liste von Transformatoren wird für jeden Sensor in einem Kontext verwaltet. Sie geben an, welche Transformationsobjekte vom Context-Handler instantiiert und aufgerufen werden sollen um die Daten des Sensors in eine einheitliche Form zu überführen. Diese vereinheitlichten Daten werden dann von einem Aggregator zusammengefasst und in einen repräsentativen Wert verwandelt. Abschließend informiert die Prozess-Notifier Komponente die Prozess-Engine durch eine entsprechende iROS-Nachricht.
Während der Entwurfszeit eines Kontextes wird sein Name sowie dazugehörende Aggregatoren bestimmt. Anschließend werden Sensoren definiert. Jeder Sensor kann potentiell unterschiedliche Kontextdaten liefern, weshalb diese in ein einheitliches Format für den Aggregator transformiert werden müssen. Hierzu werden für jeden Sensor 1..n Transformatoren zusammen mit Hilfsinformationen definiert und im Repository gespeichert. Über Transformatoren und Aggregator wird eine formalisierte Schnittstelle an die Prozess-Engine geschaffen.

2.3 Prozess-Engine

Im Kontext eines intelligenten Hauses wird unter einer Anwendung eine Zusammenstellung (Orchestrierung) von Diensten verstanden, welche durch die Services Schicht abstrahiert werden. Das Wissen und die Verwaltung von Anwendungen werden in der Prozess-Engine gespeichert. Prozesse sind Instanzen von Anwendungen, welche als eine Schablone für einen Prozess gelten. Dies dient der personalisierten Ausführung von Anwendungen. So kann im Kontext der Lokation ein und dieselbe Anwendung für zwei unterschiedliche Einwohner gleichzeitig ausgeführt werden, wobei hier zwei Instanzen derselben Anwendung mit unterschiedlichen Parametern verwendet werden. Abbildung 4 zeigt eine konzeptionelle Sicht auf die Prozess-Engine.

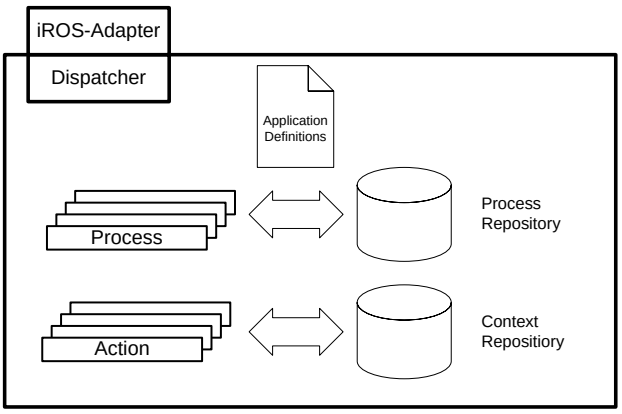

Bild 4 Konzeptionelle Sicht der Prozess-Engine

Anwendungen werden in XML spezifiziert. Beim Start der Prozess-Engine werden die Prozessschablonen aus dem Process-Repository entnommen und instanziiert. Über einen iROS-Adapter gelangen Ereignisse der Kontexterkennung zu einer Dispatcher Komponente. Diese kann anhand der Informationen aus dem Process-Repository entscheiden, welchen laufenden Prozessen die Kontextinformationen verfügbar gemacht werden sollen. Die eigentlichen Daten werden von Event-Handlern benötigt, die wiederum in der Anwendungsdefinition deklariert sind. Letztere entscheiden mit welchen Schritten der Prozess weiterverarbeitet wird sowie welche Dienste der Services Komponente als nächstes benötigt werden.
Prozesse führen Aktionen aus, welche zur Entwurfszeit aus dem Action Repository ausgewählt werden können. Über einen XML-Dialekt können Aktionen durch Kontrollstrukturen wie Sequenzen, Schleifen und Fallunterscheidung miteinander verknüpft werden. Durch diese Spezifikationssprache können sinnvolle Verknüpfungen der Agenten der Services Komponente definiert werden.

3 Szenario

Die Funktionsweise der vorgestellten Architektur soll anhand des Beispielszenarios „Besuch von Freunden“ veranschaulicht werden. Kündigen sich Freunde zu Besuch an, müssen möglicherweise einige Vorbereitungen getroffen werden. Bei einem kurzfristigen Besuch ist vielleicht nur für Essen & Trinken zu sorgen. Abbildung 5 skizziert eben erwähntes Szenario.

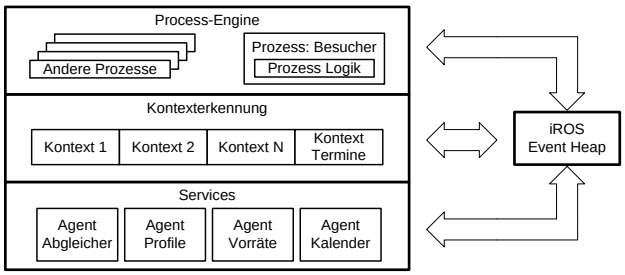

Bild 5 Szenario

Als erster tritt der Kalenderagent in Erscheinung, in dem dieser den neuen Termin im Kalender bemerkt. Dieser Agent macht die Daten über den angekündigten Besuch über den Eventheap verfügbar. Als nächstes bekommt die Kontexterkennung diese Daten und wertet diese aus. Dabei identifiziert diese die Art der Daten, die in diesem Fall zu dem Kontext „Termine“ gehören. Anschließend wendet diese eine entsprechende Transformation an um die Daten in eine standardisierte Form zu bringen. Darauffolgend gelangt die Process-Engine über den Eventheap an die Daten, nachdem diese von der Kontexterkennung auf diese aufmerksam gemacht wurde. Die Process-Engine untersucht die Nachricht und ordnet diese einem dediziert ausgeführten Prozess „Besucher“ zu, welcher wiederum eine Instanz der gleichnamigen Anwendung ist. Durch ihre Ablauflogik wird der nächste vom intelligenten Haus auszuführende Schritt ermittelt. Folglich wird die Nachricht untersucht und im Falle eines gewünschten Abendessens die folgenden Schritte aufgerufen:

1.) Ermittlung der Essensvorlieben durch einen Aufruf des Profilagenten.
2.) Ermittlung der momentanen Essensvorräte in Kühlschrank und Lagerräumen durch einen Aufruf des Vorratsagenten.
3.) Abgleich der Informationen von Schritt 1 und 2 durch einen Aufruf des Abgleichagenten.
4.) Basierend auf den Ergebnissen wird der Benutzer benachrichtigt.

Bei Schritt 4 wird der Benutzer in Kenntnis gesetzt, welches Gericht er zubereiten sollte und welche Zutaten dafür benötigt werden. Sollten Zutaten fehlen, wird dieser entsprechend informiert. An diesem Szenario sind eine Reihe von Agenten beteiligt wobei jeder für sich einen in sich geschlossenen Dienst anbietet. Beispielsweise bietet der Abgleichagent die Möglichkeit, Unterschiede in Datenmengen ausfindig zu machen. Durch diese Eigenschaft sind die Implementierungen der Agenten einfach und ihre Wiederverwendbarkeit relativ hoch. Der Nachteil bei dieser Realisierung liegt in der Notwendigkeit, eine entsprechend komplexe Prozesslogik in der Process-Engine abzubilden.

Es muss weiter untersucht werden, wie weit Intelligenz auf die Prozessebene oder auf die Agentenebene verlagert werden soll. Berücksichtigt man die heutigen SOA-Richtlinien, spricht die Steigerung der Wiederverwendbarkeit, sollte möglichst viel Logik in Prozesse und weniger in die einzelnen Dienste verlagert werden. Ob dieses Entwurfsprinzip auch auf Anwendungen für intelligente Häuser dauerhaft erfolgreich angewandt werden kann, muss noch näher untersucht werden.

4 Zusammenfassung

In diesem Papier wurde ein Vorschlag für Softwaresysteme in intelligenten Häusern gemacht. Drei Komponenten bilden ihre Kernbestandteile welche nach dem Kommunikationsmuster des Blackboards kommunizieren und gleichzeitig in der Gesamtheit das Pipes & Filter Architekturmuster abbilden. Auf der untersten Ebene sind Sensoren und Aktoren kleine Komponenten, welche durch Agenten zu Diensten gekapselt werden. Ihre Aufgabe ist die Rohdatenerfassung und das Ausführen von Anweisungen. Eine wichtige Fähigkeit von intelligenten Häusern ist die Kontexterkennung. Anwendungen basieren auf Kontextdaten und können nur auf diese Art und Weise die richtigen Entscheidungen treffen. Ein Kontext ist in diesem Zusammenhang eine Aggregation von Daten semantisch gleicher Aufgabenbereiche. Die erfassten Rohdaten gelangen in die Kontexterkennung und werden dort aggregiert sowie in ein einheitliches Datenformat transformiert. Diese Daten verarbeitet die Prozesssteuerung, welche sich durch eine workflowähnliche Definition von Anwendungen auszeichnet. Zu guter letzt schließen Dienstaufrufe von Aktoren durch laufende Prozesse den Kreislauf.

Im Rahmen eines einfachen Szenarios wurde die Funktionsweise der Architektur verdeutlicht. Als nächster Schritt ist geplant, im Rahmen des Aufbaus des Living Place Hamburg als intelligente Musterwohnung die hier vorgestellten Architekturen in Realexperimenten weiter zu entwickeln.

4 Literatur

[1] Weiser, M.: The Computer for the Twenty-First Century. Scientific American, 265, S. 94-104, 1991

[2] Pierce, J.: Personal Information Environments. – URL http://www.almaden.ibm.com/cs/projects/pie/. – Letzter Aufruf am 20.November.2008

[3] Taylor, A.; Harper, R.; Swan, L.; Izadi, S.; Sellen, A.; Perry M.: Homes that make us smart. Personal Ubiquitous Comput., Vol. 11, S. 383-393, London: Springer-Verlag, 2007

[4] JADE: Jade Development Environment. – URL http://jade.tilab.com/. – Letzter Aufruf am 9.Oktober.2008

[5] Erman, L.; Hayes-Roth, F.; Lesser, V.; Reddy, D.: The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty. ACM Computer. Surv., 12(2), S. 213-253, 1989

[6] Johanson, B.; Fox, A.: Extending Tuplespaces For Coordination in Interactive Workspace. Journal of Systems and Software, 2004