Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Projektbericht Masterprojekt 2

Malte Kantak

Framework für einen Erreichbarkeitsagenten

28. Februar 2013

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

1. Einleitung

In einer Zeit, in der die Einflüsse durch Kommunikationsmedien vor allem im privaten Bereich stark zunehmen, wird es immer wichtiger die Kommunikationsversuche computergestützt zu bewerten und gewichtet dem Bewohner zukommen zu lassen. Dies soll in erster Linie den Effekt haben, den Bewohner zu entlasten und somit seine Effektivität bezüglich seiner aktuellen Handlung zu verbessern. Als aktuelles Beispiel aus der Wirtschaft kann hier die neue „Do Not Disturb“ Funktion des iOS 6 von Apple angeführt werden[1]. Diese ermöglicht es dem Benutzer in seinen mobilen Geräten Kontakte zu definieren, welche den Benutzer in einem bestimmten Zeitraum unterbrechen dürfen und welche nicht. Solche Anwendungen zeigen, dass das zunehmende Problem der Unterbrechungen durch unerwünschte Kontakte in bestimmten Situation, gegenüber der ständig erforderlichen Erreichbarkeit für andere Kontakte, schon jetzt im Bewusstsein der Firmen und Anwender verankert ist. Um neue Lösungsansätze bezüglich dieser Problemstellung zu entwickeln soll der im Folgenden beschriebene Erreichbarkeitsagent die aktuellen Erreichbarkeitszustände des Bewohners für verschiedene Kontextgruppen dynamisch ermitteln können. Dazu wurden in vergangenen Arbeiten Grundlagen- und Umfeldanalysen durchgeführt (vgl. [Kan12a] und [Kan12b]) um wiederverwendbare Techniken zu ermitteln. Darüber hinaus wurde in der vorangegangenen Projektarbeit [Kan12c] die grundlegende Infrastruktur für den zu entwickelnden Erreichbarkeitsagenten geschaffen. Das Projekt soll exemplarisch im Living Place Hamburg[2] installiert und evaluiert werden und ist an die vorhandene Infrastruktur angepasst.

1.1. Vision

In dieser Fortsetzung des Masterprojektes sollen die Grundvoraussetzungen für die Entwicklung und Realisierung eines Erreichbarkeitsagenten abschließend geschaffen werden. Die im ersten Teil des Masterprojektes umgesetzten Komponenten werden hier durch weitere ergänzt, sodass in der anschließenden Masterarbeit der Fokus auf die Klärung der relevanten Fragestellungen (vgl. [Kan13]) gelegt werden kann. Nach der Fertigstellung dieses Masterprojektes soll es durch einfache Konfigurationen und Anpassungen möglich sein, sowohl die verwendeten Sensoren, als auch den eingesetzten Reasoner an die Bedürfnisse des Bewohners anzugleichen. Es soll folglich eine Versuchsumgebung geschaffen werden, welche die einfache Durchführung von Versuchen im Zusammenhang mit der Erreichbarkeitserkennung eines Smart-Home Bewohners ermöglicht. Mithilfe dieser Versuche sollen verschiedene Sensoren und unterschiedliche Reasonertechnologien bezüglich Ihrer Verwendbarkeit untersucht und verglichen werden können.

[1] http://support.apple.com/kb/HT5463 – Zuletzt besucht: 19.02.2013

[2] ein Smart-Home der Hochschule für angewandte Wissenschaften Hamburg [LKG+10] – http://www.livingplace.org – Zuletzt besucht: 19.02.2013

2. Vorarbeiten

Im Rahmen des ersten Teils des Masterprojektes [Kan12c] wurde bei der Realisierung ein besonderer Fokus auf die Umsetzung des Nachrichtenverkehrs des Erreichbarkeitsagenten mit der Umwelt und seiner Komponenten untereinander gelegt. Vorangegangen war die Entwicklung eines Architekturentwurfs, welcher die benötigten Komponenten und Kommunikationskanäle beschrieb. In diesem Entwurf sind in Abbildung 1 die im ersten Projektteil realisierten Komponenten hervorgehoben.

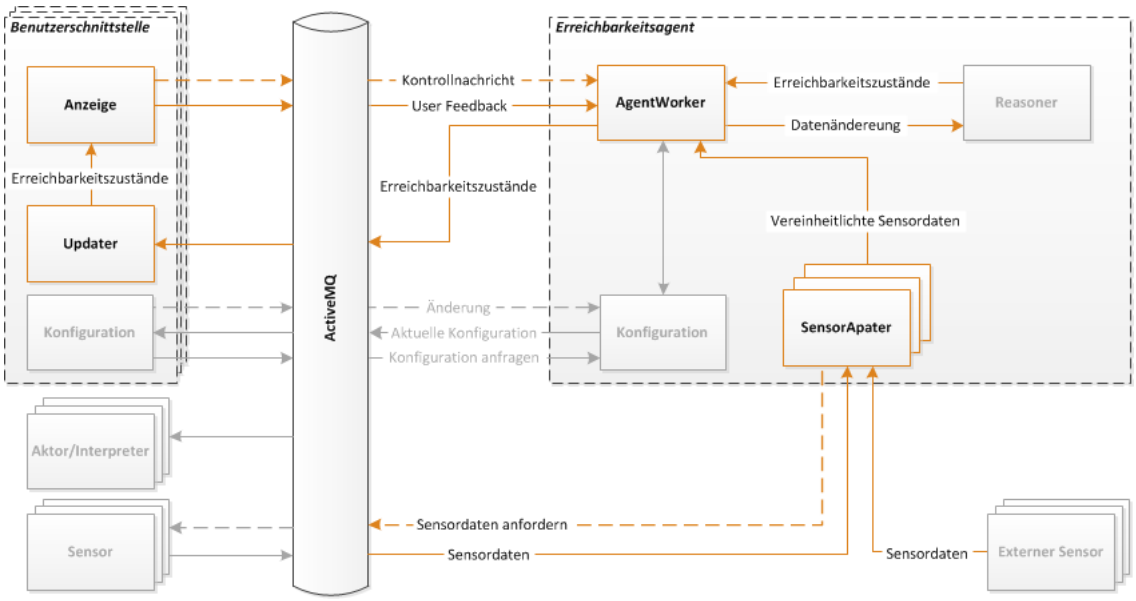

Abbildung 1: Implementierungsstand nach Projekt 1

Begonnen wurde in [Kan12c] mit der Realisierung der Anbindung vorhandener Sensoren des Living Place Hamburg. Da diese in der vorhandenen Umgebung größtenteils mithilfe von JSON[3] Nachrichten über eine ActiveMQ[4] kommunizieren, wurden die Anbindung der Sensoren so gestaltet, dass für jeden Sensortyp ein entsprechender *SensorAdapter* implementiert werden muss, welcher auf neue Messdaten des Sensors lauscht. Zusätzlich ermöglicht es diese Architektur auch andere Datenquellen anzubinden, welche nicht über die ActiveMQ kommunizieren. Empfängt ein *SensorAdapter* eine Nachricht, so wandelt er diese vom Quellformat des Sensors in eine einheitliche, interne Repräsentation des Wertes um. Diese Adapter können sehr schnell implementiert und in den Erreichbarkeitsagenten inte-

[3] **J**ava**S**cript **O**bject **N**otation – Ein Datenformat zur Kodierung von Objekten, welches sowohl von Menschen als auch Maschinen lesbar ist.

[4] System zur Weiterleitung von Nachrichten in so genannten Nachrichtenschlangen – http://activemq.apache.org/ – Zuletzt besucht: 19.02.2013

griert werden, sodass eine Erweiterung um neue Sensoren ohne nennenswerte Zeitverluste erfolgen kann.

Im Anschluss an die *SensorAdapter* wurde der *AgentWorker* realisiert. Dieser kontrolliert die Abläufe und die interne Nachrichtenkommunikation des Erreichbarkeitsagenten, und leitet die vereinheitlichten Sensordaten an den so genannten Reasoner weiter, welcher die tatsächliche Erreichbarkeitsbestimmung auf Basis der vorliegenden Sensorinformationen durchführt. Die Entwicklung des Reasoners war nicht Bestandteil des ersten Projektabschnitts, wohl aber dessen Kommunikation über den *AgentWorker* mit der Umwelt. Um die ermittelten Erreichbarkeitszustände anderen Systemen auf der zentralen ActiveMQ zur Verfügung zu stellen wurde ein JSON Format entwickelt, welches ausführlich in [Kan12c] beschrieben ist. Diese Nachrichten können zum Beispiel von anderen Systemen (z. B. einem Stressagenten [Lin12] oder einer intelligenten Türklingel [Bor12]) verwendet oder von einem Benutzerinterface dem Bewohner zur Anzeige gebracht werden.

Zum Abschluss des ersten Projektteils wurde eine Benutzerschnittstelle (siehe Abb. 2) implementiert, welche dem Bewohner die aktuell ermittelten Erreichbarkeitszustände visualisiert. Zudem bietet sie ihm vollständige Kontrolle über die Aktivitäten des Erreichbarkeitsagenten und ermöglicht durch direktes User Feedback aktives Einwirken auf seine aktuelle Erreichbarkeit.

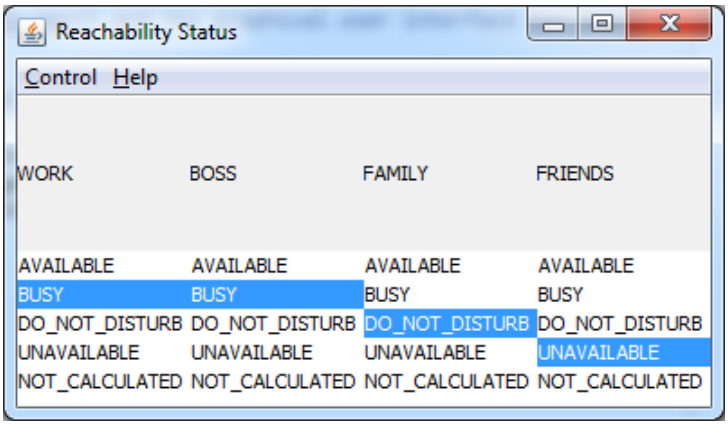

Abbildung 2: Benutzerschnittstelle nach Projekt 1

Im Rahmen der Arbeit [Kan12b] wurde unter Anderem ein Verfahren zur Simulation von nicht vorhandenen Sensoren vorgestellt. Dieses sogenannte „Wizard of Oz“ Verfahren[5] ermöglicht die Identifizierung von relevanten Sensoren, ohne dass diese physisch installiert und eingebunden werden müssen. Da aktuell noch nicht bekannt ist, welche Sensoren für die Erreichbarkeitsermittlung relevant sind, soll dieses Verfahren im weiteren Verlauf der Masterarbeit zum Einsatz kommen, um die relevanten Sensoren zu identifizieren.

[5] Technik zur Simulation von Sensorinputs (vgl. [FHA+05])

3. Erreichbarkeitsagent

Aufbauend auf die Vorarbeiten (siehe Kapitel 2) wurde der Erreichbarkeitsagent in diesem zweiten Teil um Funktionalitäten ergänzt, welche im ersten Projektteil noch nicht im Fokus standen. Neben diesen notwendigen Erweiterungen wurde eine Entwicklerschnittstelle realisiert, welche Sensorsimulationen nach dem „Wizard of Oz“ Verfahren ermöglicht. Es wurde begonnen einen regelbasierten Reasoner zu implementieren, sodass dieser dynamisch an die Vorlieben des Benutzer angepasst werden kann. Zusätzlich wurde eine weitere, stark vereinfachte Benutzerschnittstelle geschaffen, welche das intuitive Wahrnehmen der aktuellen Erreichbarkeitszustände ermöglichen soll. Abschließend wurden rudimentäre Testläufe mit dem von Benjamin Lindemann entwickelten Agenten zur Stresserkennung [Lin12] durchgeführt. Diese wiesen die vollständige Funktionsfähigkeit des Frameworks von der Entgegennahme der Sensordaten und deren Umwandlung über den regelbasierten Reasoner bis hin zur Bereitstellung und Anzeige der ermittelten Erreichbarkeitszustände nach.

3.1. Erweiterung

Neben der neuen Entwickler- und Benutzerschnittstelle wurde auch die Kernfunktionalität des Erreichbarkeitsagenten um einige Komponenten erweitert. Diese sind in Abbildung 3 hervorgehoben.

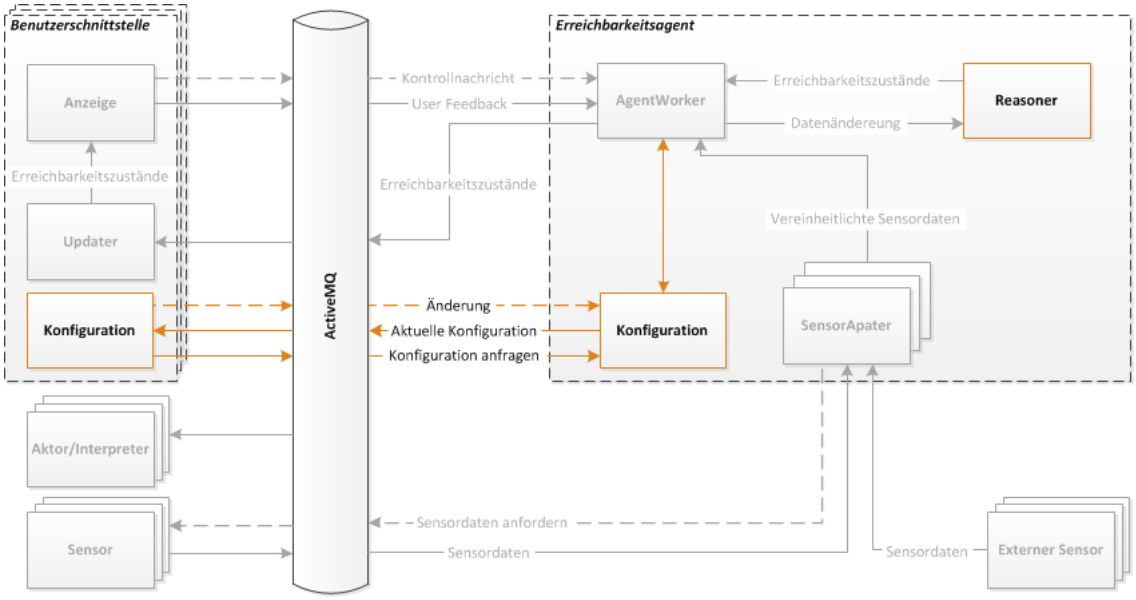

Abbildung 3: Neue Kernkomponenten

Es wurde ein Konfigurationssystem implementiert, welches es dem Bewohner ermöglicht seine Vorlieben bezüglich des Erreichbarkeitsbestimmung und wichtige Systemeigenschaften zu konfigurieren. Zusätzlich wurde die *Reasoner* Komponente, welche nach dem ersten Projektteil nur zu Zwecken des internen Nachrichtenaustauschs implementiert war, durch eine auf *JBoss Drools*[6] basierende Regelmaschine ersetzt.

3.1.1. Sensorsimulation

Ein häufig auftretendes Problem bei der Entwicklung von sensorbasierten Systemen ist, dass nicht unbedingt klar ist, welche Sensoren tatsächlich erforderlich oder zielführend sind. Um hier unnötig hohe Kosten bei der Anschaffung oder Zeitaufwände bei der Entwicklung und Integration von Sensoren zu vermeiden, welche sich im Nachhinein als nicht relevant erweisen, wurde zu Beginn dieses Projektteils eine Möglichkeit geschaffen Sensorevents von beliebigen Sensoren zu simulieren. Hierzu werden sogenannte „Stories“ angelegt, welche aus Sensorevents bestehen. Diese sind jeweils mit einem bestimmten Zeitpunkt auf einer Zeitleiste verknüpft und können dadurch geordnet und mit entsprechenden zeitlichen Abständen dem Erreichbarkeitsagenten zur Verfügung gestellt werden. Die Entwicklerschnittstelle zur Erstellung und Betrachtung dieser Stories ist in Abbildung 4 dargestellt.

Über die Optionen im oberen Bereich können neue Stories angelegt, gespeicherte Stories geladen oder die aktuelle Story gespeichert werden. Mithilfe des Formulars auf der linken Seite können neue Sensorevents zu der Story hinzugefügt werden, welche als Liste von Sensorevents auf der rechten Seite dargestellt ist. Diese Liste wird automatisch anhand der Zeitpunkte sortiert. Angezeigt wird jeweils der Zeitpunkt, der Sensortyp und dahinter in Klammern der jeweils vom Sensor gemessene Wert. Über die Zeitleiste im unteren Bereich können bestimmte Zeitpunkte auf der zu simulierenden Zeitleiste selektiert werden. Findet zu dem selektierten Zeitpunkt ein Sensorevent statt, so wird dieses in der Story grün hervor gehoben. Liegt der selektierte Zeitpunkt zwischen zwei Events, so werden beide in der Story orange markiert. Erstellte Events können per Drag & Drop von der Story auf den „Trash“ Bereich verschoben und dadurch gelöscht werden.

Die Daten werden in einem simplen Textformat als **.sgs* Datei[7] gespeichert. Um eine solche Story zu simulieren wurde ein Java-Programm geschrieben, welches die Dateien in JSON Nachrichten (vgl. Listing 4 (*JsonSensorDataFeederItem*)) konvertiert und diese dann kontrolliert auf der ActiveMQ publiziert. Um diese JSON Nachrichten verarbeiten zu können, muss der jeweilige Sensoradapter im Erreichbarkeitsagent implementiert sein.

[6] Regelbasiertes Reasoningframework für Java – `http://www.jboss.org/drools/` – Zuletzt besucht: 19.02.2013

[7] **S**tory **G**enerator **S**imulation – Ein Dateiformat, welches kodierte Sensorevents beinhaltet und im Zusammenhang mit dem Erreichbarkeitsagenten zu Simulationszwecken verwendet werden kann.

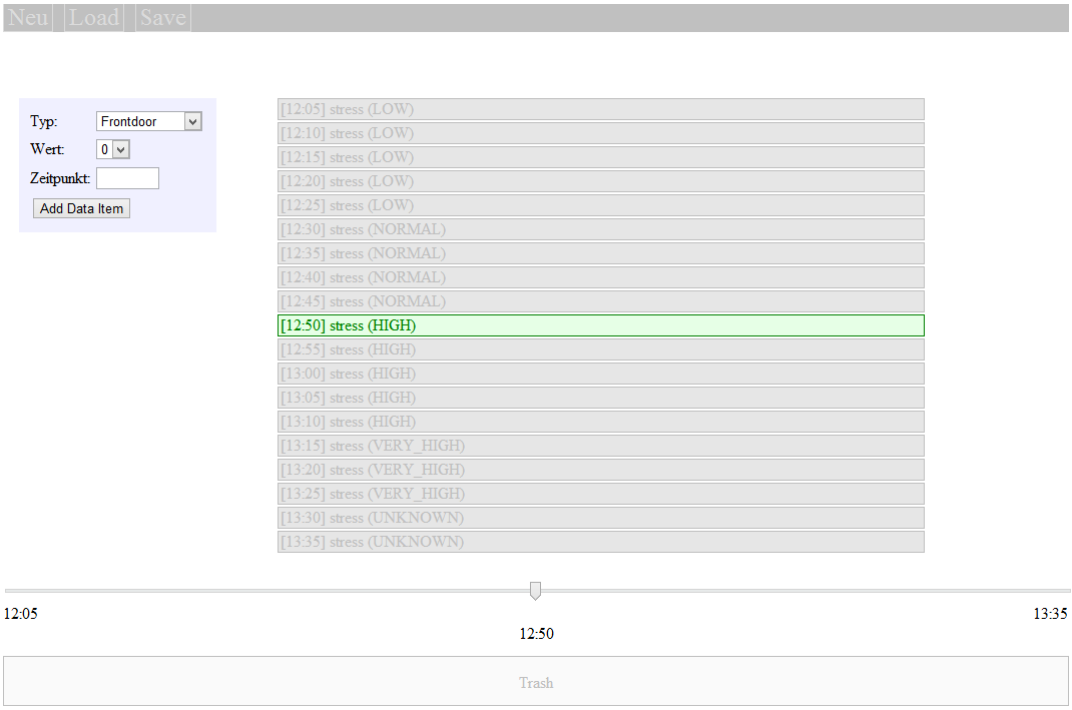

Abbildung 4: Sensorsimulation nach „Wizard of Oz“

3.1.2. Zusätzliche Komponenten

Wie in Abbildung 3 dargestellt, gibt es zusätzlich zu der Implementierung der Sensorsimulation zwei wichtige Erweiterungen am Framework. Zum Einen wurde ein Konfigurationssystem ergänzt, welches es dem Benutzer ermöglicht von beliebigen Orten auf die Funktionsweise des Erreichbarkeitsagenten einwirken zu können, zum Anderen wurde das Framework durch einen regelbasierten Reasoner erweitert, welcher eine Manipulation der Regelbasis zur Laufzeit ermöglicht.

Konfiguration

Das Konfigurationssystem wurde als eigenständige Komponente in das Framework integriert. Beim Start des Erreichbarkeitsagenten wird die Komponente *Konfiguration* als Thread gestartet. Dieser lauscht daraufhin auf Anfragen auf der ActiveMQ. Eine solche Anfrage kann auf zwei verschiedene Arten gestellt werden, entweder als Abfrage der aktuellen Konfiguration oder als Änderungsanfrage zu bestimmten Konfigurationswerten.

Empfängt die Komponente eine Anfrage der aktuellen Konfiguration, welche dem JSON Format aus Listing 1 (*JsonRAPropertiesRequest*) genügt, so werden die aktuell definierten Konfigurationswerte ermittelt und anschließend als JSON Nachricht der Form Listing 2 (*JsonRAPropertiesResult*) auf der ActiveMQ bereitgestellt. Hierbei wird in der Anfrage definiert,

welcher Konfigurationsbereich ermittelt werden soll (*Type*). Zur Verfügung stehen hier aktuell die Systemkonfiguration (*SYSTEM*) und die Einstellungen des Reasoners (*REASONER*). Die Anfrage kann durch die Angabe einer bestimmten Variable (*Property*) auf den Wert dieser beschränkt werden. Sollen alle Werte aus diesem Bereich ermittelt werden, so wird hier der Platzhalter „**ALL*“ verwendet. Wichtig ist zudem, dass die Anfragenachricht eine eindeutige Kennung (*Id*) aufweist, da auf diese in der Antwortnachricht Bezug genommen wird. Die Antwortnachricht enthält neben der Kennung der Anfragenachricht (*RequestId*) wieder den Konfigurationsbereich und eine eigene Kennung. Darüber hinaus enthält die Antwortnachricht die zu ermittelnden Konfigurationsvariablen inklusive ihrer aktuellen Werte (*Properties*). Die Anfrage von Konfigurationswerten und die entsprechende Rückmeldung wird hauptsächlich im Zusammenhang mit dem Aufbau der Benutzerschnittstelle zur Konfigurationsmanipulation (siehe Abschnitt 3.1.3) verwendet. Ändert der Benutzer einen Konfigurationswert aus dieser Benutzerschnittstelle heraus, so wird eine Änderungsanfrage, wie in Listing 3 (*JsonRAPropertiesChange*) beschrieben, über die ActiveMQ an den Erreichbarkeitsagenten übermittelt. Diese Nachricht enthält im Wesentlichen wieder den Konfigurationsbereich (*Type*), den Namen der Variablen (*Property*) und den neuen Wert (*NewValue*), welche von der Konfigurationskomponente im Erreichbarkeitsagenten empfangen und umgesetzt wird.

Regelbasierter Reasoner

Um die Funktionalität der Frameworks testen zu können, war es notwendig einen lauffähigen Reasoner zu implementieren, welcher die Kommunikation möglichst dicht am Echtbetrieb realisiert. Ein Ziel der Masterarbeit wird es sein verschiedene Reasonertechnologien vor dem Hintergrund der Erreichbarkeitserkennung zu vergleichen. Es wurde auf Basis verschiedener Vorüberlegungen beschlossen, ein lernendes System und einen regelbasierten Reasoner zu verwenden und diese jeweils getrennt und kombiniert miteinander zu vergleichen. Ein regelbasierter Reasoner ist, bedingt durch die vorgegebene Regelbasis, weniger adaptiv als ein lernendes System, da die Basisregeln im Vorwege definiert werden. Die Adaptivität des Erreichbarkeitsagenten ist allerdings entscheidend für die Akzeptanz beim Bewohner. Um den regelbasierten Reasoner möglichst konfigurierbar zu gestalten, musste zum Einen eine Konfigurationsmöglichkeit in den Erreichbarkeitsagenten integriert werden (vgl. Abschnitt 3.1.2 *Konfiguration*) und zum Anderen eine Möglichkeit geschaffen werden die Regelbasis des Reasoners zur Laufzeit zu modifizieren. Als regelbasierten Reasoner wird in dieser Arbeit *JBoss Drools* verwendet. Die ermittelten Sensordaten werden als *SensorData* Objekte dem sogenannten *Working Memory*[8] zur Verfügung gestellt, welches auf Basis der bekannten Regeln einen Erreichbarkeitszustand pro definierter Personengruppe ermittelt. Die Regelbasis wird zur Ermittlung der Erreichbarkeit hierarchisch aufgebaut, sodass für jede Personengruppe eine Hauptregel definiert ist, in welcher benötigte Sensorobjekte geladen werden. Zusätzlich zu diesen Hauptregeln gibt es pro Personengruppe jeweils eine von der Hauptregel erbende Kindregel je möglichem Erreichbarkeitszustand, welche die Be-

[8] dt. Arbeitsgedächtnis – Enthält Sensordaten und definierte Regeln zur Laufzeit

dingungen für das Erreichen dieses Zustands enthält. Diese Regeln sind je Personengruppe disjunkt, sodass jeweils nur ein Erreichbarkeitszustand pro Gruppe festgestellt werden kann. Da *JBoss Drools* an dieser Stelle keine Möglichkeiten zur Modifikation der Regelbasis zur Laufzeit anbietet, muss diese vollständig neu generiert, das aktuell laufende *Working Memory* deaktiviert und ein neues auf Basis der geänderten Regeln gestartet werden. Ein Beispiel für eine sehr einfache Regelbasis ist in Listing 5 dargestellt.

3.1.3. Benutzerschnittstelle

Damit der Benutzer die dynamische Konfiguration vornehmen kann, wurde die Benutzerschnittstelle erweitert, sodass in der Taskleiste nun der Menüpunkt „Configuration“ ergänzt wurde. Unter diesem Menüpunkt hat der Benutzer die Auswahl zwischen „System“ und „Reasoner“. Durch anklicken der jeweiligen Option wird dem Benutzer ein weiteres Fenster mit den angebotenen Konfigurationsmöglichkeiten geöffnet. In Abbildung 5 ist exemplarisch die Konfigurationsschnittstelle für die Systemoptionen dargestellt.

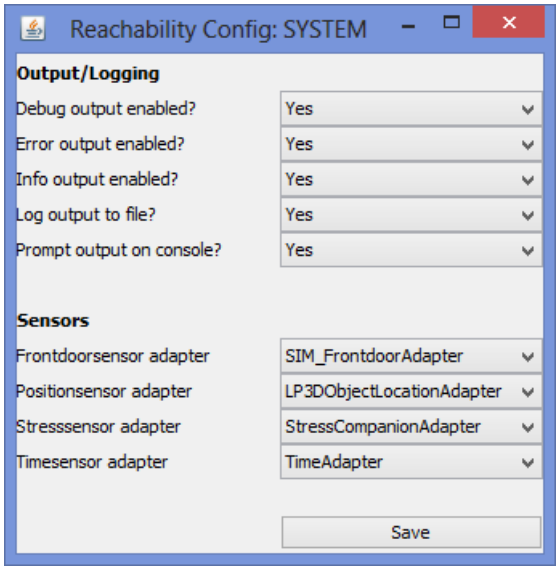

Abbildung 5: Konfigurationsschnittstelle

Wie im Beispiel kann die Anzeige thematisch gegliedert sein. Im oberen Bereich sind Konfigurationsoptionen bezüglich der Ausgabe und Aufzeichnung zusammengefasst, im unteren Bereich hat der Benutzer die Möglichkeit die jeweils zu verwendenden Sensoradapter auszuwählen. Dies kann relevant sein, wenn es verschiedene Sensoren für einen bestimmten Wert gibt, wie zum Beispiel für die Position des Bewohners (z. B. UWB- (Ubisense[9]) [EKV+11] oder Videotracking).

[9] Ein Positionierungssystem für den Innenbereich – http://www.ubisense.net – Zuletzt: 27.02.2013

Ergänzend zu der Änderung an der vorhandenen Benutzerschnittstelle wurde prototypisch eine weitere Repräsentationsmöglichkeit für die Anzeige der ermittelten Erreichbarkeitszustände realisiert. Ziel dieser neuen Schnittstelle ist es, dem Bewohner möglichst einfach und schnell einen Überblick über seine aktuelle Erreichbarkeit zu geben. Diese Überlegung resultierte in einer farbbasierten Darstellungsform, welche in Abbildung 6 dargestellt ist.

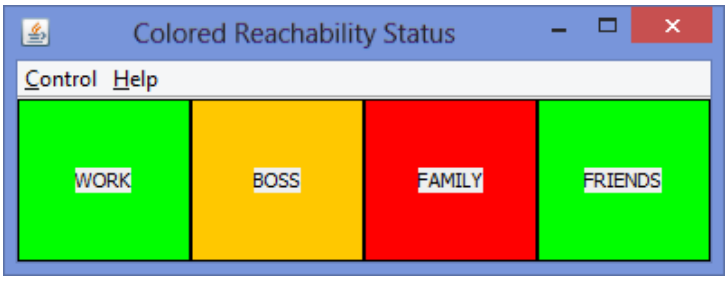

Abbildung 6: Farbbasierte Erreichbarkeitsanzeige

Wie man leicht erkennen kann, wird der Erreichbarkeitszustand jeweils durch eine Farbe repräsentiert. Dabei sind folgende Farben möglich: Grau (UNAVAILABLE), Rot (DO_NOT_DISTURB), Gelb-orange (BUSY), Grün (AVAILABLE) und Weiß (NOT_CALCULATED). Die jeweiligen Personengruppen werden nebeneinander als rechteckige Fläche dargestellt, wobei die Bezeichnung der jeweiligen Gruppe im Zentrum der Fläche steht. Die Farbe der Fläche wird durch den jeweils ermittelten Erreichbarkeitszustand determiniert. Da diese Benutzerschnittstelle vorrangig der Information des Benutzers dienen soll, wurde hier auf Interaktivität in Form von aktivem Feedback verzichtet.

3.2. Implementierungsstand

Nach Abschluss des zweiten Teils des Masterprojektes wurde der Erreichbarkeitsagent soweit implementiert, dass die Voraussetzungen für das effektive Arbeiten an der Masterarbeit erfüllt sind. Die Kommunikation wurde sowohl intern als auch extern vollständig realisiert. Es wurde die Möglichkeit geschaffen Sensoren und Reasoner sehr einfach zu ergänzen bzw. auszutauschen. Auf Basis der vorliegenden Implementierung kann der Fokus im Rahmen der Masterarbeit klar auf die Identifizierung relevanter Faktoren für die Erreichbarkeitserkennung und den Vergleich verschiedener Reasonertechnologien gelegt werden. Informationen zum weiteren Vorgehen können der Seminararbeit [Kan13] entnommen werden.

4. Zusammenfassung

Im Rahmen der vorliegenden Arbeit konnten die Vorbedingungen für ein Framework zur Entwicklung eines Erreichbarkeitsagenten geschaffen werden. Das Framework wurde soweit vorbereitet, dass beliebige Sensoren (sowohl über die ActiveMQ als auch über andere Kommunikationsschnittstellen) angeschlossen und verwendet werden können. Dazu ist es lediglich notwendig einen entsprechenden Sensoradapter zu implementieren, welcher die Daten in die interne Repräsentation konvertiert. Anschließend muss nur noch die jeweilige Dimension im System registriert werden. Es wurden mehrere Möglichkeiten geschaffen, die Vorgänge und Resultate der Erreichbarkeitsbestimmung für den Bewohner und den Entwickler sichtbar zu machen und zu überwachen. Es wurde ein Konfigurationssystem installiert, welches es dem Benutzer erlaubt sowohl auf die System- als auch auf die Reasonereigenschaften einzuwirken, um den Erreichbarkeitsagenten dadurch an seine persönlichen Vorlieben anzupassen. Die Bestimmung der Erreichbarkeitszustände erfolgt in einer lose gekoppelten, leicht austauschbaren Komponente, wodurch Testläufe mit verschiedenen Reasonertechnologien möglich sind. Ein regelbasierter Reasoner wurde exemplarisch mithilfe von *JBoss Drools* realisiert, sodass auch hier Anpassungen an der Regelbasis zur Laufzeit erfolgen können.

4.1. Ausblick

Das in dieser und der vorangegangenen Arbeit [Kan12c] entwickelte Framework liefert eine lauffähige Testumgebung für die anschließende Masterarbeit. Um deren Fragestellungen [Kan13] untersuchen zu können, wird es im weiteren Verlauf dennoch notwendig sein zusätzliche Anpassungen am Quellcode vorzunehmen. So wird es für die Identifizierung der relevanten Faktoren der Erreichbarkeitsermittlung erforderlich sein, verschiedene *SensorAdapter* für die in Frage kommenden Sensoren zu implementieren, damit deren Messresultate während der Simulation in die Erreichbarkeitsermittlung einfließen können. Die Realisierung dieser Adapter wird aber, dank der losen Kopplung der Komponenten, sehr einfach möglich sein. Es werden wahrscheinlich weitere Feinjustierungen bei der Generierung der Regelbasis nach abschließender Sensoranalyse notwendig werden. Alternativ zu dem implementierten regelbasierten Reasoner wird ein lernendes System als Reasoner ergänzt werden müssen, um die beiden Technologien vergleichen zu können. Hierfür wird voraussichtlich, je nach lernendem System, entweder *SNIPE*[10] oder das *Encog Machine Learning Framework* [11] verwendet werden.

[10] **S**calable And Generalized **N**eural **I**nformation **P**rocessing **E**ngine – Ein Java-Framework für künstliche neuronale Netze – http://www.dkriesel.com/tech/snipe – Zuletzt besucht: 19.02.2013

[11] Ein Java-Framework für verschiedene lernende Systeme – http://www.heatonresearch.com/encog – Zuletzt besucht: 19.02.2013

Literatur

[Bor12] BORNEMANN, Sven B.: *Mobile Türklingel für Smart Homes*. Hamburg, Germany, HAW Hamburg, AW1 Ausarbeitung, 2012. – http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master11-12-aw1/bornemann/bericht.pdf – Zugriff: Februar, 2013

[EKV+11] ELLENBERG, Jens ; KARSTAEDT, Bastian ; VOSKUHL, Sören ; LUCK, Kai von ; WENDHOLT, Birgit: An environment for context-aware applications in smart homes. In: *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. Guimarães, Portugal, Sept. 21–23, 2011. – http://ipin2011.dsi.uminho.pt/PDFs/Poster/50_Poster.pdf – Zugriff: Februar, 2013

[FHA+05] FOGARTY, James ; HUDSON, Scott E. ; ATKESON, Christopher G. ; AVRAHAMI, Daniel ; FORLIZZI, Jodi ; KIESLER, Sara ; LEE, Johnny C. ; YANG, Jie: Predicting human interruptibility with sensors. In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 12 (2005), March, Nr. 1, S. 119–146. – ISSN 1073–0516. – http://www.interruptions.net/literature/Fogarty-TOCHI05.pdf – Zugriff: Februar, 2013

[Kan12a] KANTAK, Malte: *Erreichbarkeit in Smart-Homes*. Hamburg, Germany, HAW Hamburg, Ausarbeitung Anwendungen 1, 2012. – http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master11-12-aw1/kantak/bericht.pdf – Zugriff: Februar, 2013

[Kan12b] KANTAK, Malte: *Erreichbarkeit in Smart-Homes – Related Work*. Hamburg, Germany, HAW Hamburg, Ausarbeitung Anwendungen 2, 2012. – http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2012-aw2/kantak/bericht.pdf – Zugriff: Februar, 2013

[Kan12c] KANTAK, Malte: *Vorarbeit für einen Erreichbarkeitsagenten*. Hamburg, Germany, HAW Hamburg, Projektbericht Masterprojekt 1, 2012. – http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2012-proj1/kantak.pdf – Zugriff: Februar, 2013

[Kan13] KANTAK, Malte: *Erreichbarkeit in Smart-Homes*. Hamburg, Germany, HAW Hamburg, Ausarbeitung Masterseminar, 2013

[Lin12] LINDEMANN, Benjamin: *Stress am IT-Arbeitsplatz – Projektbericht*. Hamburg, Germany, HAW Hamburg, Masterprojekt 1, 2012. – http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2012-proj1/lindemann.pdf – Zugriff: Februar, 2013

[LKG+10] LUCK, P. D. K. ; KLEMKE, P. D. G. ; GREGOR, S. ; RAHIMI, Mohammad A. ; VOGT, Matthias: Living Place Hamburg - A place for concepts of IT based modern living / HAW Hamburg. Hamburg, Germany, Mai 2010. – Forschungsbericht. – http://livingplace.informatik.haw-hamburg.de/content/LivingPlaceHamburg_en.pdf – Zugriff: Februar, 2013

A. Nachrichtenformate

Im Folgenden werden Beispiele für JSON Nachrichten aufgelistet, welche bei der Kommunikation des Erreichbarkeitsagenten verwendet werden. Hierbei handelt es sich um eine Auswahl der Nachrichten, welche im Rahmen dieser Arbeit behandelt werden. Für weitere JSON Nachrichten, welche im Zusammenhang mit dem Erreichbarkeitsagenten verwendet werden, wird an dieser Stelle auf den Anhang A der Arbeit [Kan12c] verwiesen. Es werden jeweils nur die im Zusammenhang mit dem Erreichbarkeitsagenten relevanten Attribute dargestellt. Die Nachrichten sind in JSON[12] codiert.

Listing 1: JsonRAPropertiesRequest

```
{
  "Type"       : "SYSTEM",
  "Property"   : "*ALL",
  "Version"    : "2012.2",
  "Id"         : "ReachabilityUI_Config_SYSTEM_454284552",
  [. . .]
}
```

Listing 2: JsonRAPropertiesResult

```
{
  "RequestId" : "ReachabilityUI_Config_SYSTEM_454284552",
  "Type"      : "SYSTEM",
  "Properties": {
    "SENSOR_TIME"       : "TimeAdapter"
    "SENSOR_STRESS"     : "SIM_StressCompanionAdapter"
    "PRINT_OUTPUT"      : "true"
    "SENSOR_POSITION"   : "LP3DObjectLocationAdapter"
    "ERROR"             : "true"
    "LOG_OUTPUT"        : "true"
    "INFO"              : "true"
    "DEBUG"             : "true"
    "SENSOR_FRONTDOOR"  : "SIM_FrontdoorAdapter"
  },
  "Version"   : "2012.4",
  "Id"        : "
     ReachabilityAgentWorker_PropertyService_1333803003",
  [. . .]
}
```

[12] **J**ava**S**cript **O**bject **N**otation

Listing 3: JsonRAPropertiesChange

```
{
  "Type"      : "SYSTEM"
  "Property"  : "DEBUG"
  "NewValue"  : "false"
  "Version"   : "2012.2"
  "Id"        : "ReachabilityUI_Config_SYSTEM_1723597492"
  [. . .]
}
```

Listing 4: JsonSensorDataFeederItem

```
{
  "SensorType"  : "stress"
  "SensorValue" : "VERY_HIGH"
  [. . .]
}
```

B. JBoss Drools Regelbasis

Im Folgenden ist beispielhaft eine einfache Realisierung einer Regelbasis in *JBoss Drools* für den Erreichbarkeitsagenten aufgeführt. Es werden hierbei nur die Regeln für die Personengruppe *WORK* aufgeführt und auf die Regeln der Personengruppen *BOSS*, *FAMILY* und *FRIENDS* verzichtet, da diese simultan zu den aufgeführten Regeln definiert sind. Zur Erreichbarkeitsbestimmung wurde hier lediglich die Dimension *Stress* verwendet (vgl. [Lin12]).

Listing 5: JBoss Drools Regelbasis (Beispiel)

```
1  // ------------------------------------------------------------
2  // Autogenerated JBoss Drools rule base
3  //   Purpose:      Determining the current reachability status
4  //   Generated by: ReachabilityAgent
5  //   Version:      2012.4
6  //   Author:       Malte Kantak <switchback@hotmail.de>
7  //   Institute:    HAW Hamburg
8  //   Generated at: Tue Jan 08 10:30:04 CET 2013
9  // ------------------------------------------------------------
10
11 package rule.base.calcResult
12
13 import de.hawhamburg.reachability.Config
14 import de.hawhamburg.reachability.sensor.SensorData
15 import de.hawhamburg.reachability.reasoner.engine.
      DroolsBasedEngine
16 import de.hawhamburg.reachability.enumeration.ReachabilityGroup
17 import de.hawhamburg.reachability.enumeration.ReachabilityStatus
18 import de.hawhamburg.stresscompanion.output.
      StressCompanionStressLevel
19
20 global DroolsBasedEngine REASONER;
21 global java.lang.Long ACCESS_KEY;
22
23 // ********** DEFAULT LOADING RULE ************************
24 rule "Default"
25     enabled false
26     when
27         $stress: SensorData( name == Config.SENSOR_STRESS )
28     then
29 end
30
31 // ********** DETERMINE REACHABILITY FOR WORK GROUP ********
```

```
32  rule "Default WORK" extends "Default"
33      enabled false
34      when
35      then
36  end
37
38  rule "WORK AVAILABLE" extends "Default WORK"
39      when
40          SensorData(this == $stress, value == (double)
               StressCompanionStressLevel.LOW.ordinal())
41      then
42          REASONER.setReachabilityStatus(ReachabilityGroup.WORK,
               ReachabilityStatus.AVAILABLE, ACCESS_KEY);
43  end
44
45  rule "WORK BUSY" extends "Default WORK"
46      when
47          SensorData(this == $stress, value == (double)
               StressCompanionStressLevel.NORMAL.ordinal())
48      then
49          REASONER.setReachabilityStatus(ReachabilityGroup.WORK,
               ReachabilityStatus.BUSY, ACCESS_KEY);
50  end
51
52  rule "WORK DO_NOT_DISTURB" extends "Default WORK"
53      when
54          SensorData(this == $stress, value == (double)
               StressCompanionStressLevel.HIGH.ordinal())
55      then
56          REASONER.setReachabilityStatus(ReachabilityGroup.WORK,
               ReachabilityStatus.DO_NOT_DISTURB, ACCESS_KEY);
57  end
58
59  rule "WORK UNAVAILABLE" extends "Default WORK"
60      when
61          SensorData(this == $stress, value == (double)
               StressCompanionStressLevel.VERY_HIGH.ordinal())
62      then
63          REASONER.setReachabilityStatus(ReachabilityGroup.WORK,
               ReachabilityStatus.UNAVAILABLE, ACCESS_KEY);
64  end
65
66  rule "WORK NOT_CALCULATED" extends "Default WORK"
67      when
```

```
68          SensorData(this == $stress, value == (double)
               StressCcompanionStressLevel.UNKNOWN.ordinal())
69      then
70          REASONER.setReachabilityStatus(ReachabilityGroup.WORK,
               ReachabilityStatus.NOT_CALCULATED, ACCESS_KEY);
71  end
72
73  [. . .]
```