Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung Masterprojekt 2

Jens Ellenberg

Entwicklung des Wecker 2.0

Fakultät Technik und Informatik
Department Informatik

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

1.1. Motivation

Mark Weiser hat in seiner Vision des 21. Jahrhundert einen fiktiven Tag aus dem Leben von Sal beschrieben [13]. In dieser Vision gibt es verschiedene Computer, beziehungsweise Programme, die Sal das Leben erleichtern. Seine Vision stammt aus den 80er Jahren. Die Funktionen der Programme sind in der Wohnung allgegenwärtig, aber werden nicht als störend wahrgenommen. Vielmehr unterstützen sie den Bewohner und wirken wie kleine Helfer, auf die man sich verlassen kann. Die Programme sind Teil der Wohnung und Teil des Lebens in dieser Vision. Seit dieser Zeit hat die Technik und auch die Informatik große Fortschritte gemacht. Die Computer sind schneller und kleiner geworden. Dadurch können sie mit mehr Daten umgehen und auch eine Vielzahl von Sensoren besser auswerten und nutzen. In der Informatik wurden neue Konzepte und auch Sprachen entwickelt. Damit haben sich auch neue Möglichkeiten erschlossen, größere und komplexere Anwendungen zu entwickeln. Aus diesem Grund stellt sich heutzutage die Frage, ob es nicht möglich ist, die Programme aus der Vision von Mark Weiser umzusetzen. In diesem Projekt soll so ein Programm entstehen. Es soll sich der Umgebung bewusst sein und intelligent auf diese reagieren und seine Funktionen dementsprechend anpassen.

1.2. Ziel des Projekts

In diesem Projekt soll der „Wecker 2.0“ umgesetzt werden. Die ersten Ideen zum Wecker wurden in „Ein Wecker in einem ubicom Haus“ (vgl. [4]) beschrieben. Der Wecker ist ein Programm, dass sich dem Kontext in der Wohnung bewusst sein soll. Anhand des Kontexts passt er seine Funktionen an. Das bedeutet, dass je nach Situation in der Wohnung und je nach Verhalten des Bewohners die „Alarm-Zeit“ neu berechnet werden soll. Dabei soll der Wecker den Kontext in der Wohnung nicht selbst erkennen. Der Kontext soll von anderen Einheiten der Wohnung erkannt werden und über ein zentrales Nachrichtensystem an den Wecker weitergegeben werden. Über dieses Nachrichtensystem soll der Wecker diese und andere Informationen mitgeteilt bekommen. Außerdem soll der Wecker den Alarm nicht selbst anzeigen, sondern nur der Wohnung mitteilen.

2. Wecker 2.0

2.1. Szenarien

Der Wecker soll nichts Neues können, sondern die Funktionen eines handelsüblichen Weckers[1] an die Bedürfnisse des Bewohners anpassen. Dabei orientieren sich die Szenarien an einem typischen Arbeitstag. Für so einen Tag sind die folgenden Aktionen beispielhaft für den Bewohner der intelligenten Wohnung gegeben:

- Schlafen im Bett
- Aufwachen
- Im Bad waschen
- Sich anziehen
- Etwas essen
- Die Wohnung verlassen

Der „Wecker 2.0“ soll für jede dieser Aktionen dem Bewohner ausreichend Zeit einplanen. So soll der Bewohner nach dem Aufstehen Zeit haben, sich zu waschen, sich anzuziehen und etwas zu essen. Außerdem muss die Zeit für den Weg zum Ort des Termins beachtet werden. Für den Bewohner soll die Reihenfolge der Aktionen nicht festgelegt sein. Der Wecker soll sich auf den Bewohner einstellen und nicht umgekehrt. Wenn der Bewohner schon aufgestanden ist, soll sich der Zeitpunkt, an dem der Wecker aktiv werden muss, ändern. Genauso ändert sich der Alarm-Zeitpunkt, wenn der Bewohner sich gewaschen und angezogen hat. In diesem Fall müsste nur noch die Zeit für das Frühstück und die Zeit für den Weg zum Termin beachtet werden.

[1] „handelsüblicher Wecker“ meint hier so einen Wecker, wie man ihn in einem Kaufhaus heutzutage findet

Aus diesem Szenario ergeben sich die folgenden Zustände für den Wecker, in denen er jeweils einen anderen Alarm-Zeitpunkt ermitteln muss:

1. Schlafender Bewohner
2. Hungriger Bewohner
3. Ungewaschener Bewohner
4. Falsch oder nicht angezogener Bewohner
5. eine beliebige Kombination aus den Punkten 2 - 4

Alle möglichen Zustände sind in einem Zustandsdiagramm zusammengefasst (Anhang: A, Abbildung: A.1). In dem Diagramm sind auch die verschiedenen Übergänge zwischen den Zuständen zu sehen. Die Anfangssituation ist der schlafende Bewohner in der Nacht. Dieser führt in der Regel nach dem Aufwachen die Aktionen: „Waschen, Anziehen und Essen“ in beliebiger Reihenfolge durch. Dabei werden in dem Diagramm auch Aktionen beachtet, die nicht der Regel entsprechen. Eine Person, die sich noch nicht angezogen hat, wird normalerweise nicht die Wohnung verlassen. Sollte aber die Kontext-Erkennung diesen Kontext-Wechsel (Anziehen) nicht erkannt haben, wäre diese Aktion vorstellbar. Genauso könnte nach einem Essens-Kontext erneut etwas gegessen werden. Daher muss das Programm in der Lage sein, auf jeden möglichen Kontext zu reagieren. Aus jedem der Zustände ist es durch den Kontext „Schlafen“ auch möglich, in den Anfangszustand zu wechseln. Diese Übergänge wurden in diesem Diagramm weggelassen, damit das Diagramm noch lesbar bleibt.

2.2. Architektur

Der Wecker ist ein Programm von vielen im Living Place Hamburg [9]. Es arbeitet nicht autonom, sondern ist abhängig von verschiedenen anderen Programmen. Der Wecker benötigt den aktuellen Kontext aus einer Kontext-Erkennung. Außerdem hat der Wecker keine Ausgabeeinheit. Wenn der Alarm-Zeitpunkt gekommen ist, wird der Alarm dem Living Place bekanntgegeben und der Living Place muss sich dann um eine entsprechende Umsetzung kümmern.

Aus den oben genannten Gründen kann man den „Wecker 2.0“ auch als Agent in einer Multi-Agenten-Umgebung sehen. Er ist über ein Nachrichtensystem mit verschiedenen anderen Agenten in der intelligenten Wohnung verbunden.

Um für die Inhalte der Kommunikation zwischen den verschiedenen Programmen eine einheitliche Basis zu haben, bietet sich eine Ontologie an. Die Ontologie wurde schon in den

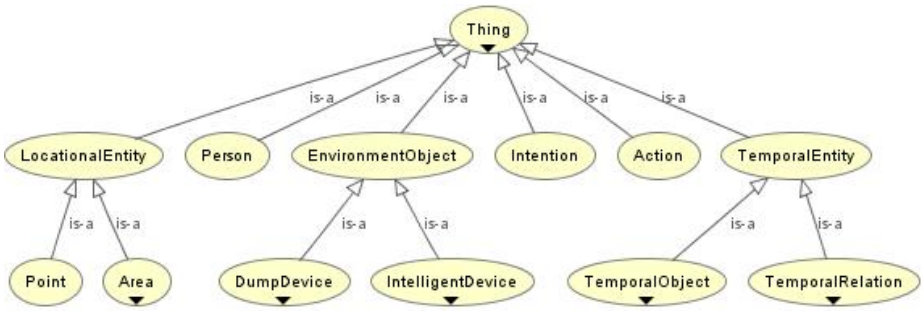

Abbildung 2.1.: Kontext Ontologie

Vorarbeiten zum „Wecker 2.0“ entwickelt und findet hier nun ihre Anwendung [3]. In der Ontologie werden auch die verschiedenen Dimensionen für die Darstellung des Kontexts festgehalten. Dabei wird der Kontext nach dem 5w1h Modell dargestellt [7]. Die Interpretation des Kontexts ist dabei aktionsorientiert (vgl. [1]).

Die Dimensionen in Abbildung 2.1 entsprechen dabei:

LocationEntity: Where-Kontext (Wo findet die Aktion statt?)

Person: Who-Kontext (Wer führt die Aktion aus?)

EnvironmentObject: What-Kontext (Was ist Gegenstand der Aktion?)

Intetion: Why-Kontext (Warum wird die Aktion ausgeführt?)

Action: How-Kontext (Was ist die Aktion?)

TemporalEntity: Whan-Kontext (Wann wird die Aktion ausgeführt?)

Der Wecker soll sein Verhalten den verschiedenen Situationen in der Wohnung anpassen. Das bedeutet, dass er ein bestimmtes Verhalten in jeder Situation hat. Diese wird mit dem Entwurfsmuster des „Zustandsautomaten“ (vgl.: Designe Pattern [6]) umgesetzt. Nach diesem Muster wird für jeden Zustand eine Klasse modelliert, die für den aktuellen Zustand die Zeiten richtig berechnet und für die Zustandsübergänge verantwortlich ist.

Der Wecker ändert sein Verhalten, wenn neue Situationen in der Wohnung erkannt werden. Diese Kontext-Erkennung soll in den kommenden Monaten entstehen. Daher ist es für die Überprüfung der Funktionalität des Weckers notwendig, die Erkennung zu simulieren. Genauso werden die Möglichkeiten für das Anzeigen des Alarms simuliert, bis der Wecker für die intelligente Wohnung einsatzbereit ist.

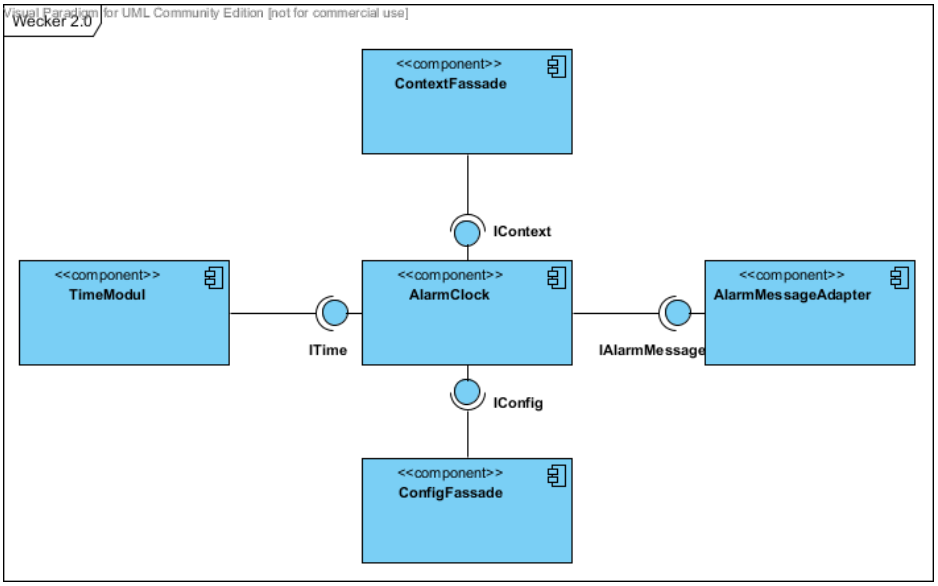

Abbildung 2.2.: Komponenten Diagramm des Wecker 2.0

2.3. Komponenten

Im Folgenden wird der Wecker genauer beschrieben. Es wird näher auf die Einzelheiten der Architektur eingegangen und die Komponenten die den Wecker ausmachen werden beschrieben. Zum Schluss wird kurz auf die Visualisierung der externen Komponenten eingegangen.

2.3.1. Allgemein

Neben den Funktionen des Weckers, die in der Komponente „AlarmClock“ zusammengefasst sind, gibt es noch vier weitere Komponenten im „Wecker 2.0“. Die Zeit des Weckers muss der realen Zeit entsprechen. Das „TimeModul“ ist für diese Überwachung der Zeit zuständig. Diese Komponente sendet in regelmäßigen Abständen die aktuelle Zeit an die Wecker-Komponente. Die Komponenten „ContextFassade“ und „AlarmMessageAdapter“ sind für die Kommunikation mit dem Nachrichtensystem zuständig. Diese Komponenten wurden von den Grundfunktionen des Weckers getrennt, um Änderungen in der Kommunikation später leichter anzupassen. Über die „ConfigFassade“ können Einstellungen am Wecker vorgenommen werden. Aus Gründen der Wartbarkeit liegt auch diese Komponente getrennt von den Grundfunktionen des Weckers. Die Abbildung 2.2 gibt einen Überblick über die verschiedenen Komponenten.

Im Abschnitt 2.2 wurde ein Nachrichtensystem genannt, mit dem der Wecker interagiert. Im Living Place wird dieses Nachrichtensystem durch den ActiveMQ von der Apache Software Foundation realisiert [10]. Dieses Nachrichtensystem funktioniert als Blackboard, auf dem die verschiedenen Komponenten ihre Nachrichten hinterlassen können und jeder Interessierte diese Nachrichten empfangen kann (vgl.: [5]).

Für den Living Place existiert auch ein Wrapper für den ActiveMQ, durch den alle Nachrichten gespeichert werden [10]. Dadurch lässt sich der Kontext aus der Vergangenheit abfragen und die Alarm-Nachrichten sind auch für spätere Abfragen gespeichert. Dieser Wrapper wurde in diesem Projekt eingesetzt. So kann der Wecker auch für den Kontext aus der Vergangenheit weiterentwickelt werden und andere Programme können die gespeicherten Alarmsignale nutzen.

Der Wecker muss mit verschiedenen anderen Programmen zusammenarbeiten. Diese Programme funktionieren unabhängig von dem Wecker. Daher ist es notwendig, die Kommunikation zwischen den Programmen klar zu regeln. Im Living Place wurde in [11] für die Blackboard-Kommunikation das Format „JSON“ gewählt. Daher wird der Kontext mit diesen Vorgaben auf dem Blackboard verteilt.

2.3.2. Komponente: AlarmClock

Der Wecker räumt dem Bewohner für die folgenden Aktionen Zeit ein: Aufstehen, Waschen, Essen, Anziehen und Terminwahrnehmung. Je nach dem, was der Bewohner von diesen Aktionen schon durchgeführt hat, wird die Zeit neu berechnet. Eine Alarm-Nachricht wird immer dann ausgegeben, wenn der Zeitpunkt erreicht ist, an dem mit einer dieser Aktionen angefangen werden muss, um alle Aktionen zu schaffen.

Die Meldungen des Weckers unterteilen sich in Alarm- und Warnmeldungen. Alarmmeldungen sind die wichtigen Meldungen, für die eine Reaktion des Bewohners für notwendig gehalten wird. Dies ist beispielsweise beim Aufstehen der Fall. Aber eine Reaktion ist auch notwendig, wenn der Bewohner sich noch nicht richtig gekleidet hat oder wenn der letztmögliche Zeitpunkt gekommen ist, die Wohnung zu verlassen. Warnmeldungen sind alle anderen Meldungen, wenn der Bewohner beispielsweise darauf hingewiesen wird, dass er nur noch Zeit zu frühstücken ODER waschen hat, aber nicht mehr für beides.

Neben den Zuständen des laufenden Betriebs, die in den Szenarien schon beschrieben wurden, hat der Wecker noch weitere Zustände. Diese Zustände sind in Abbildung 2.3 zu sehen. Nachdem der Wecker gestartet wurde, geht er mit einem Start-Befehl in seinen normalen Betrieb über. Nun kann es sein, dass der Wecker seine Aktionen einstellen soll. Dafür gibt es einen Sleep-Befehl. Aus diesem Zustand kann der Wecker jederzeit wieder aktiviert werden. Diese Zustandsänderung ist denkbar, wenn der Bewohner von dem Wecker nicht gestört

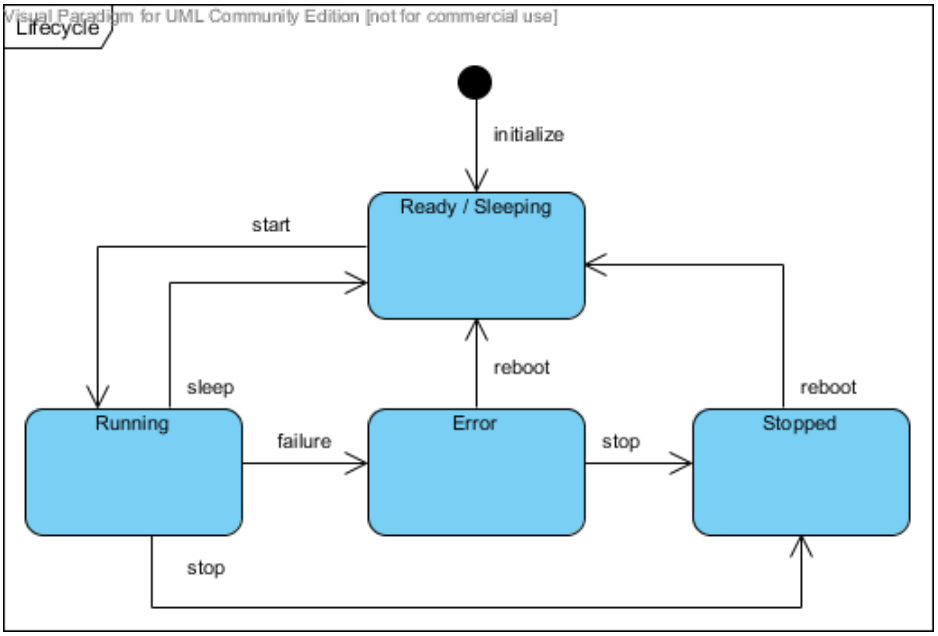

Abbildung 2.3.: Lebenszyklus des Weckers

werden will, auch wenn er Termine in seinem Kalender aktiv hat. Der Error-Zustand ist für einen Fehler im Programm vorgesehen. Wenn während des Betriebs ein Fehler auftritt, der den weiteren Betrieb verhindert oder ein unvorhergesehener Fehler auftritt, dann geht der Wecker in diesen kontrollierten Warte-Zustand. Natürlich muss es auch möglich sein, den Wecker zu beenden. Nach dem Beenden sollte nur ein vollständiger Neustart möglich sein. Der Neustart geschieht durch einen Reboot-Befehl.

2.3.3. Komponente: TimeModul

Die Zeit ist ein wichtiges Element des Weckers. Nach [8] ist für den Anwender das Gefühl der Sicherheit eines der wichtigsten bei der Nutzung eines Weckers. Daher muss für den Wecker sichergestellt werden, dass er für die Termine immer die richtige Zeit für seine Berechnungen nutzt. Das bedeutet, dass er die reale Zeit kennen muss, um zum entsprechenden Zeitpunkt den Alarm zu erzeugen, der in der Realität genau diesem Zeitpunkt entspricht. Diese Synchronisation mit der Realität wird in dem TimeModul sichergestellt. Die Abfrage der Realzeit wird über das Internet mit dem Network Time Protocol[2] realisiert.

Bei der Reaktion auf die verschiedenen Termine kommt es nicht auf eine Genauigkeit im Sekundenbereich an. Viele Termine werden nicht mit Sekunden, aber auf die Minute genau

[2] vgl.: RFC 5905 http://www.ietf.org/rfc/rfc5905.txt

vermerkt. Daher wird aus dem TimeModul zu jeder vollen Minute die aktuelle Zeit an den Wecker gesendet. Diese Zeit-Signale nutzt der Wecker, um die Alarm- und Warnmeldungen zu erzeugen.

2.3.4. Komponente: ContextFassade

Die Kontext-Erkennung befindet sich derzeit noch in der Entwicklung. Daher ist davon auszugehen, dass sich bei der Darstellung des Kontexts demnächst noch etwas ändert. Während des laufenden Betriebes könnte die Kontext-Erkennung und die Darstellung des Kontexts weiterentwickelt werden. Daher sollte das Empfangen des Kontexts außerhalb des Weckers realisiert werden. In der ContextFassade wird der Kontext vom Blackboard empfangen und in eine interne Darstellung für den Wecker umgewandelt. Des Weiteren muss der Wecker jederzeit bereit sein, den Kontext vom Blackboard zu empfangen, da man sonst mit einem Datenverlust rechnen muss. Um diese Eigenschaft zu gewährleisten, wird die Empfangseinheit als ein eigener Thread realisiert, der den Kontext vom Blackboard empfängt.

2.3.5. Komponente: AlarmMessageAdapter

Das Ausgeben des Alarms liegt nicht im Zuständigkeitsbereich des Weckers. Der Wecker gibt nur die Alarm-Nachricht an das Blackboard weiter. In dieser Komponente wird die interne Alarm-Nachricht des Weckers an das Nachrichtenformat des Blackboard angepasst. Wie mit diesen Meldungen umgegangen werden soll, wird zurzeit in anderen Projekten erarbeitet. In [12] wird an einem zentralen Ansatz gearbeitet. In dieser Arbeit löst die Alarmmeldung ein Ziel aus, dass durch eine zentrale Planung versucht wird zu erreichen. Es ist aber auch ein dezentraler Ansatz denkbar. Jedes Gerät, das sich in der Nähe des Bewohners befindet, reagiert selbstständig auf den Alarm, bis der Bewohner eine „Alarm-Aus“ Nachricht an das Blackboard schickt. Als Alarmgeber sind in der Wohnung die Lautsprecher, die Bildschirme oder die Lichtanlage denkbar.

2.3.6. Komponente: ConfigFassade

Der Wecker ist keine Uhr mit Weckfunktion, die in der Wohnung steht. Er ist ein intelligentes Programm, das auf einem Rechner irgendwo in der Wohnung läuft. Daher muss eine Möglichkeit geschaffen werden, mit der der Bewohner den Wecker bedienen kann.

Die Zeiten für die einzelnen Aktionen (Aufstehen, Waschen, Anziehen und Frühstücken) sind feste Werte, die über diese Komponente abgefragt und geändert werden können. Dem Benutzer einer Software sollte immer die Möglichkeit gegeben werden, diese auf die eigenen

Bedürfnisse anzupassen. Ansonsten entsteht schnell ein Gefühl der Bevormundung und die Software wird nicht weiter genutzt. Daher ist es möglich, die aktuellen Termine und Alarmzeiten von dem Wecker über diese Komponente zu erfahren und eventuell diese Alarm-Signale zu deaktivieren. Genauso kann man die Funktionen des Weckers deaktivieren (Sleeping-Zustand) oder ganz beenden (Stop-Zustand).

Eine offene Frage ist, wie ein ausgelöster Alarm beendet werden sollte. Da der Living Place für das Alarmieren des Bewohners zuständig ist, könnte die Alarm-Stop-Nachricht auch unabhängig vom Wecker erfolgen. Für dieses Projekt wurde entschieden, dass die Stop-Nachricht über die Config-Fassade an den Wecker gerichtet wird und dieser wiederum das Alarmende dem Blackboard mitteilt. Der Wecker sollte als Auslöser des Alarms auch die Verantwortung für dessen Beendigung tragen. Daher sollte er auch für das Ende sorgen. Hieraus ergeben sich noch weitere Vorteile. Der Wecker kann auch ohne eine externe Stop-Nachricht den Alarm nach einer gewissen Zeit wieder beenden oder den Alarm erneut aktivieren, wenn der Bewohner nach dem Alarmende noch nicht entsprechend reagiert hat.

2.4. Visualisierung

Um den Wecker zu testen und seine Funktion zu überprüfen, sind verschiedene Bedienelemente und Anzeigen notwendig. Hier werden verschiedene Einheiten simuliert, die später im Living Place durch die reale Umgebung ersetzt werden sollen.

Für die Festlegung des Kontexts wurde eine einfache Oberfläche entwickelt, auf der man die verschiedenen Kontexte ein- und ausschalten kann. Dieses Programm sendet den Kontext dann an das Blackboard. Im Living Place soll das später die Kontext-Erkennung übernehmen, die die Situation in der Wohnung automatisch erkennt.

Für die Einstellungen am Wecker wurde eine Visualisierung entwickelt, auf der der nächste Alarm, die aktuelle Zeit des Weckers und die anderen Einstellungen des Weckers zu sehen sind. Über diese Schaltfläche kann man auch einen Alarm beenden, den Wecker deaktivieren oder ganz stoppen. So eine Oberfläche sollte es genauso im Living Place geben. Es wäre denkbar, dass sie über die Küchentheke [2] oder auf einem anderen Bediengerät abrufbar ist.

Für den Alarm wurde ein Programm entworfen, das auf dem Blackboard lauscht und die Aktivierung oder Beendigung eines Alarms anzeigt. Dieser wird im Living Place von anderen Programmen übernommen, die dann die Signale an die Lautsprecher, Bildschirme oder an das Licht senden. Dieses könnte dahingehend eingeschränkt werden, dass diese Signale nur dann gesendet werden, wenn der Bewohner auch in der Nähe ist.

3. Zusammenfassung und Ausblick

3.1. Zusammenfassung

In diesem Projekt ist ein Programm aus der Vision von Mark Weiser entstanden. Der Wecker der Zukunft passt sein Verhalten den verschiedenen Situationen in einer intelligenten Wohnung an. Beispielhafte Kontexte sind in diesem Projekt: Schlafen, Aufwachen, Waschen, Anziehen und Frühstücken. Der Wecker hat verschiedene Zustände für jede relevante Situation in der Wohnung. Der Wechsel von einem Zustand in den anderen wird durch einen neuen Kontext ausgelöst, der in der Wohnung erkannt wird. Das Erkennen des Kontexts innerhalb der Wohnung ist nicht Teil dieses Projekts. Der Wecker erhält die Kontext-Informationen über eine Blackboard Architektur und teilt den Alarm über dieses Blackboard der Wohnung mit. Die Wohnung zeigt den Alarm dann über verschiedene Geräte selbstständig an.

3.2. Ausblick

Der Wecker reagiert auf den aktuellen Kontext in der Wohnung. Er beachtet Kontext in unerwarteten Situationen. Kontext wird aber in der Regel nicht mit absoluter Sicherheit erkannt. Es könnte sein, dass eine Situation nicht erkannt wird oder dass das System einen Kontext erkennt, der mit der Realität nicht übereinstimmt. Hier könnte der Wecker die Wahrscheinlichkeiten beachten, mit der ein erkannter Kontext tatsächlich existiert.
Der Kontext, auf den der Wecker zurzeit reagiert, ist einfach gehalten, weil die möglichen Zustände bei weiteren Kontexten exponentiell ansteigen würden. Es könnte über eine Verfeinerung des Kontexts nachgedacht werden, wenn dabei die möglichen Zustände des Weckers nicht jede Permutation der verschiedenen Kontexte repräsentieren.
Nicht jeder Termin sollte einen Alarm erzeugen. Hier könnte man den Wecker so modifizieren, dass er die wichtigen von den unwichtigen Terminen unterscheidet. Dies könnte durch einen Lernalgorithmus erfolgen. Außerdem haben die meisten Menschen verschiedene Gewohnheiten. Vielleicht sind für den einen TV-Sendungen wichtig oder man möchte am Wochenende den Morgen genießen, ohne dass ein konkreter Termin vorliegt. Hier könnte der Wecker versuchen, die Gewohnheiten zu lernen und so den Bewohner noch weiter zu unterstützen.

Literaturverzeichnis

[1] ABOWD, G. D. ; DEY, A. K. ; BROWN, P. J. ; DAVIES, N. ; SMITH, M. ; STEGGLES, P. : Towards a Better Understanding of Context and Context-Awareness. In: *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*. Springer-Verlag (HUC '99). – ISBN 3–540–66550–1, 304–307

[2] BARNKOW, L. : Eine Multitouch-fähige Küchentheke: Im Kontext des Living Place Hamburg / HAW Hamburg. Version: 2010. http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master09-10-aw1/barnkow/bericht.pdf. – Forschungsbericht

[3] ELLENBERG, J. : Vorarbeiten für den Wecker 2.0 / HAW Hamburg. Version: 2010. http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2010-proj1/ellenberg.pdf. – Forschungsbericht

[4] ELLENBERG, J. : Ein Wecker in einem ubicom Haus / HAW Hamburg. Version: 2010. http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master09-10-aw1/Ellenberg/bericht.pdf. HAW Hamburg. – Forschungsbericht

[5] ERMAN, L. D. ; HAYES-ROTH, F. ; LESSER, V. R. ; REDDY, D. R.: The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. In: *ACM Comput. Surv.* 12 (1980), June, 213–253. http://dx.doi.org/http://doi.acm.org/10.1145/356810.356816. – DOI http://doi.acm.org/10.1145/356810.356816. – ISSN 0360–0300

[6] In: GAMMA, E. ; HELM, R. ; JOHNSON, R. ; VLISSIDES, J. : *Design patterns: abstraction and reuse of object-oriented design*. Springer-Verlag New York, Inc.. – ISBN 3–540–43081–4, 701–717

[7] JANG, S. ; WOO, W. : 5W1H: Unified user-centric context. In: *Proceedings of the 7th International Conference on Ubiquitous Computing* Citeseer, 2005

[8] KLAUSER, K. ; WALKER, V. : It's about time: an affective and desirable alarm clock. In: *DPPI '07: Proceedings of the 2007 conference on Designing pleasurable products and interfaces*. New York, NY, USA : ACM, 2007. – ISBN 978–1–59593–942–5, S. 407–420

[9] LUCK, P. D. K. ; KLEMKE, P. D. G. ; GREGOR, S. ; RAHIMI, M. A. ; VOGT, M. : Living Place Hamburg – A place for concepts of IT based modern living / Hamburg University of Applied Sciences. Version: Mai 2010. http://livingplace.informatik.haw-hamburg.de/content/LivingPlaceHamburg_en.pdf. – Forschungsbericht

[10] OTTO, K. ; VOSKUHL, S. : Weiterentwicklung der Architektur des Living Place Hamburg / HAW Hamburg. Version: 2011. http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master10-11-proj2/otto-voskuhl.pdf. – Forschungsbericht

[11] OTTO, K. ; VOSKUHL, S. : Entwicklung einer Architektur für das Living Place Hamburg / HAW Hamburg. Version: 2010. http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2010-proj/otto-voskuhl.pdf. HAW Hamburg. – Forschungsbericht

[12] TENNSTEDT, S. : Agentenbasierte Workflow Engine fÃ¼r eine intelligente Wohnung / HAW Hamburg. Version: 2010. http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master09-10-seminar/tennstedt/bericht.pdf. HAW Hamburg. – Forschungsbericht

[13] WEISER, M. : The computer for the 21st century. In: *SIGMOBILE Mob. Comput. Commun. Rev.* 3 (1999), Nr. 3, S. 3–11. http://dx.doi.org/10.1145/329124.329126. – DOI 10.1145/329124.329126. – ISSN 1559–1662

A. State Machine

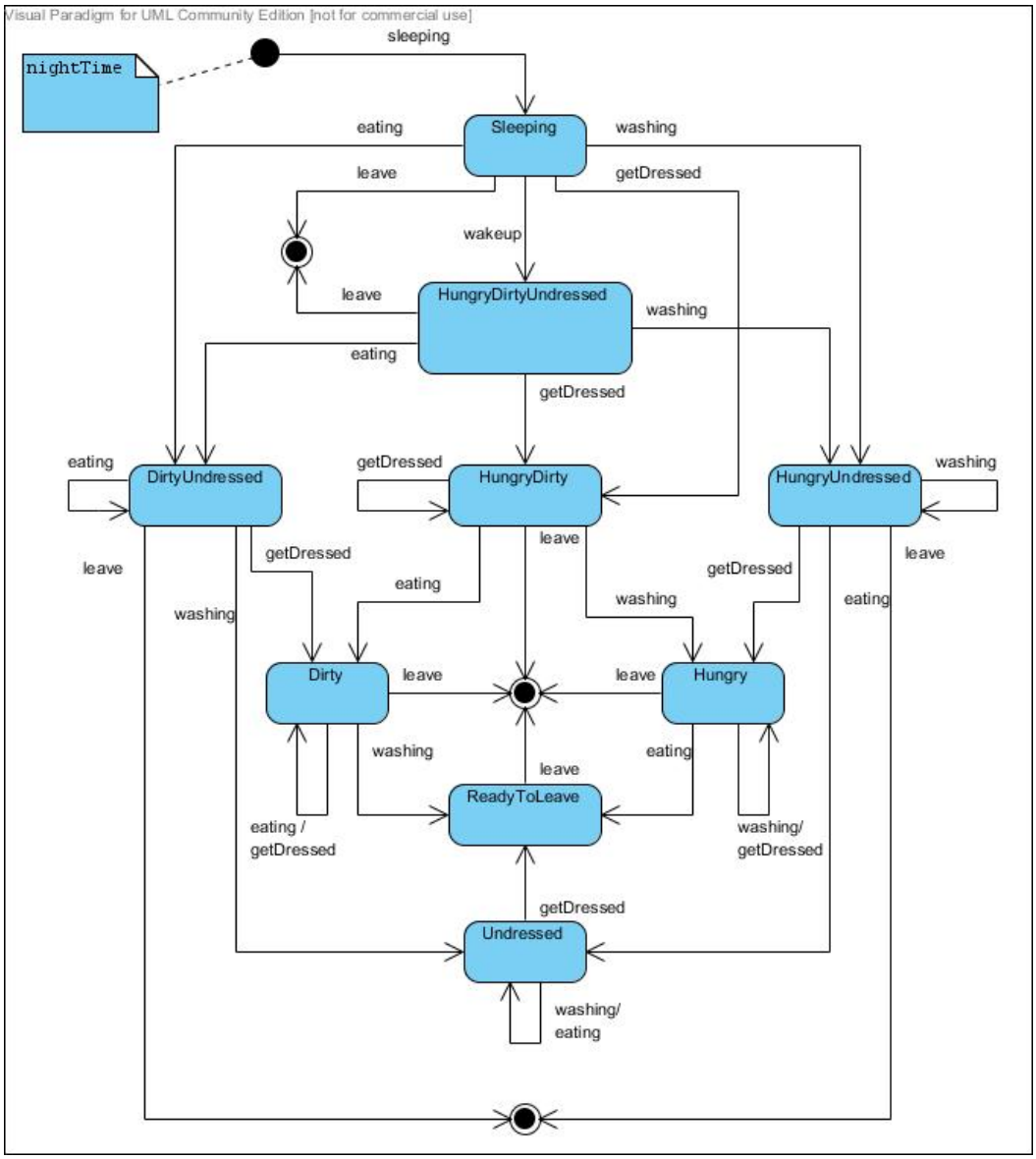

Abbildung A.1.: Zustände des Weckers

B. Bildquellen

Die Quellen der Bilder sind:

- Bild in Abschnitt: 2.1 von www.aboutpixel.de Fotograf: Frank Zunker
- Abbildung 2.1 wurde mit Protégé 4.1 Beta erstellt.
- Abbildung 2.2 wurde mit dem Tool Visual Paradigm for UML 8.0 erstellt.
- Abbildung 2.3 wurde mit dem Tool Visual Paradigm for UML 8.0 erstellt.
- Abbildung A.1 wurde mit dem Tool Visual Paradigm for UML 8.0 erstellt.