Usability-Evaluation in intelligenten Umgebungen

Dissertation

zur

Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

der Fakultät für Informatik und Elektrotechnik

der Universität Rostock

vorgelegt von

Stefan Propp, geb. am 10.02.1982 in Schwerin
aus 19059 Schwerin, Bertolt-Brecht-Straße 26

urn:nbn:de:gbv:28-diss2010-0116-1

Rostock, 28.02.2010

Begutachtet von

Prof. Dr.-Ing. Peter Forbrig als betreuender Hochschullehrer,
Universität Rostock, Fakultät für Informatik und Elektrotechnik,
Institut für Informatik,
Albert-Einstein-Straße 21, 18059 Rostock

Prof. Dr.-Ing. Thomas Kirste,
Universität Rostock, Fakultät für Informatik und Elektrotechnik,
Institut für Informatik,
Albert-Einstein-Straße 21, 18059 Rostock

Prof. Dr. rer. nat. Michael Herczeg,
Universität zu Lübeck,
Institut für Multimediale und Interaktive Systeme,
Ratzeburger Allee 160, 23538 Lübeck

Verteidigung: 02.07.2010

Zusammenfassung

Mit fortschreitender Evolution der Computertechnik hat sich die Interaktion zwischen Mensch und Computer stark verändert. Während bei klassischen Arbeitsplatzrechnern Eingaben meist explizit, bspw. über Maus und Tastatur erfolgen, findet in intelligenten Umgebungen, sogenannten „Smart Environments“, auch verstärkt implizite Interaktion statt. So können Menschen in einer physischen Umgebung interagieren, die auf ihre Aktivitäten reagiert und Assistenz bietet, um sie bei der Erreichung ihrer Ziele zu unterstützen. Interaktionen sind dabei gekennzeichnet durch Kooperation mehrerer Nutzer beim Durchführen einer Aufgabe, den Einbezug mehrerer Geräte, Multimodalität und Kontextabhängigkeit. Nutzer erwarten darüber hinaus, dass sie das System im Alltag direkt ohne Bedienungsanleitung nutzen können. Durch die gebotene Assistenz können Nutzer Zeit sparen und mental entlastet werden. Um allerdings die Akzeptanz der neuen Interaktionsmöglichkeiten bei Nutzern sicherzustellen, ist eine Evaluation der Usability notwenig. Die Evaluation von Smart Environments ist durch besondere Herausforderungen gekennzeichnet, die sich von der Evaluation klassischer Applikationen unterscheidet. Bspw. ist der komplette Aufbau eines Smart Environments für einen Test ressourcenintensiv, kooperative Interaktionen können nicht losgelöst evaluiert werden, umfangreiche Kontextvariablen sind zu berücksichtigen und bei der Auswertung von Daten aus Feldversuchen ist der Umgang mit privaten Daten kritisch. Daher müssen bestehende Evaluationsmethoden angepasst und neue Evaluationsmethoden entwickelt werden, die diese Probleme aufgreifen. Ein besonderer Schwerpunkt dieser Arbeit liegt dabei auf der Evaluation während früher Phasen der Entwicklung. Entsprechende Methoden gehören bei traditionellen Applikationen zu den meistverwendeten [VMS+02], sind aber für Smart Environments kaum verfügbar.

In der vorliegenden Arbeit wurde ein Konzept entwickelt, das Usability-Evaluation in den modellbasierten Prozess der Entwicklung integriert, um jederzeit entwicklungsbegleitend Evaluationen durchführen zu können. Das Ziel ist, Probleme in möglichst frühen Phasen zu finden, in denen die Behebung erheblich einfacher und preiswerter ist (1:10:100-Regel). Dazu wurde das Konzept eines virtuellen Smart Environment entwickelt und implementiert, das die physische Laborumgebung ergänzt und vier Anwendungsszenarien unterstützt: (1) Während der Anforderungsanalyse werden mit zukünftigen Nutzern die Anforderungen erarbeitet und in ersten Aufgabenmodellen abgebildet. Um diese frühzeitig zu validieren,

werden die Modelle animiert, um „Cognitive Walkthroughs“ nun auch für Smart Environments zu ermöglichen. (2) Da viele Menschen keine Erfahrungen mit der Interaktion in Smart Environments haben, fällt es ihnen schwer, vorausschauend Anforderungen zu artikulieren. Das virtuelle Smart Environment unterstützt die Durchführung von „Wizard of Oz“-Experimenten, um zukünftigen Nutzern eine Vorschau zu geben, wie sich die gewünschte Assistenz „anfühlt“. (3) Nachdem erste Komponenten des Smart Environments implementiert sind, wie bspw. die Intentionserkennung, sollen diese auch getestet werden. Hierfür werden allerdings Kontextinformationen benötigt. Das virtuelle Smart Environment erlaubt es, diese Kontextdaten künstlich zu erstellen, um in der Anforderungsanalyse ermittelte Szenarien nachzustellen und gezielt definierte Auslöser für die Assistenz zu testen. Fehler auf Modellebene können dabei behoben und das verbesserte Modell erneut getestet werden. (4) Wenn das komplette physische Smart Environment aufgebaut ist, werden Nutzertests durchgeführt. Die dabei aufgezeichneten Daten können später zur Analyse visualisiert werden. Interaktionen werden dabei in anonymisierter Form abgespielt, um den Schutz privater Daten der beteiligten Personen sicherzustellen.

Die vorliegende Arbeit stellt ein Vorgehensmodell für die iterative Evaluation von Smart Environments vor, die den modellbasierten Entwicklungsprozess begleitet und damit Lösungsansätze liefert auf dem Weg zur Einführung des Paradigmas der Benutzerorientierten Gestaltung („user-centred design“, UCD) [DIN99] für Smart Environments.

Schlüsselwörter: Usability-Evaluation, Smart Environments, Aufgabenmodellierung

Computing-Reviews-Klassifizierung: D2.2 Tools and Techniques, H1.2 User / Machine Systems, H 5.1 Multimedia Information Systems

Abstract

The evolution of computing technology caused changes for human computer interaction. While traditional desktop computers are mainly based on explicit interaction with keyboard and mouse, smart environments often shift the focus on implicit interaction. Hence people may interact naturally in a physical environment, which reacts on their activities and provides assistance, to support users to accomplish their goals. Interactions are characterized by several users performing a task cooperatively, the involvement of several devices, multimodality and context dependence. Furthermore, humans expect to use the system directly without reading any manuals. Assistance systems try to save users' time and decrease the mental workload. However, to ensure the acceptance of new interaction possibilities, usability evaluations are necessary. The evaluation of smart environments introduces new challenges, which differ from the evaluation of traditional applications. For instance, building a completely functional physical smart environment for testing purposes is resource consuming, cooperative interactions cannot be evaluated in isolation, various context parameters have to be taken into account and when analyzing data from field studies handling private data is critical. Hence existing evaluation methods have to be revised and additionally new methods have to be invented, to cope with the challenges. A particular focus of this work is the evaluation during early development stages. Methods dedicated to those stages rank among the most applied methods for evaluating traditional applications, but are lacking in the field of smart environments.

This work presents a concept, which integrates usability evaluation into the model-based development process, to facilitate evaluating developed artefacts at any stage of development. The concept aims at identifying issues at early development stages, because according to the 1:10:100-rule rectifying artefacts early is easier and cheaper. The concept of a virtual smart environment was developed and implemented to complement the physical laboratory environment und to support four scenarios of use: (1) During requirements analysis requirements are elicited together with prospected users and specified as task models. To validate them, models are animated to facilitate cognitive walk-troughs for smart environments. (2) Due to the fact, that many people don't have any experience with interactions in smart environments, it is difficult for them to express requirements in advance. The virtual smart environment provides support for "wizard of oz"-experiments, to give future users a

first glimpse at the envisioned assistance and how it would “feel”. (3) After having implemented first components of the smart environment, like the intention recognition, they should be tested subsequently. However, context data is needed as a prerequisite. The virtual smart environment allows creating artificial sensor data to evaluate whether the scenarios elicited during requirements analysis are supported, particularly evaluating the specified triggers for assistance functionality. Issues at the model level can be rectified and rapidly re-evaluated. (4) When the physical smart environment is completely set up, user tests are conducted. The captured data can be visualized for analysis purposes. Interactions are replayed in an anonymized form, to ensure the privacy of involved users.

This work presents a process model for the iterative evaluation of smart environments, accompanying the model-based development process and paving the way to establish the paradigm of user-centred design (UCD) [DIN99] for smart environments.

Key words: Usability-Evaluation, Smart Environments, Task Modelling

Computing-Reviews-Classification: D2.2 Tools and Techniques, H1.2 User / Machine Systems, H 5.1 Multimedia Information Systems

Danksagung

Mein Dank gilt meinem Betreuer, Prof. Dr. Peter Forbrig, der die Arbeit in diesem interessanten Forschungsfeld ermöglicht hat und in dessen Arbeitsgruppe für Softwaretechnik ich integriert war. Ich danke auch allen Kollegen der Arbeitsgruppe für die vielfältige Unterstützung während meiner Forschungstätigkeit.

Den Gutachtern Prof. Dr. Thomas Kirste und Prof. Dr. Michael Herczeg danke ich für konstruktive Hinweise zum Erstellen der vorliegenden Arbeit.

Für anregende fachliche Diskussionen bedanke ich mich bei Gregor Buchholz, Maik Wurdel und Christoph Burghardt. Gregor Buchholz danke ich darüber hinaus auch für das Korrekturlesen der Arbeit.

Daniel Reichart danke ich für technische Hinweise, die es mir erleichtert haben, die von mir entwickelten Werkzeuge nahtlos in die bestehende Infrastruktur einzufügen.

Bedanken möchte ich mich auch bei Studenten, die im Rahmen von Projekten und akademischen Arbeiten zu der vorliegenden Arbeit beigetragen haben.

Den Stipendiaten des Graduiertenkollegs MuSAMA danke ich für die produktive und gute Arbeitsatmosphäre.

Allen Probanden danke ich für die Teilnahme an der Nutzerstudie und ihre vielfältigen Kommentare zu dem entwickelten Werkzeug.

Für die finanzielle Unterstützung danke ich der Deutschen Forschungsgemeinschaft, von der ich im Rahmen des Graduiertenkollegs 1424 „Multimodal Smart Appliance Ensembles for Mobile Applications“ durch ein Stipendium gefördert wurde.

Ein sehr herzlicher Dank gilt meiner Familie und meinen Freunden, die alle auf ihre Art und Weise zum Gelingen dieser Arbeit beigetragen haben.

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Motivation

Mit zunehmender Komplexität technischer Produkte steigt die Bedeutung der Usability. In einer Umfrage der Zeitschrift „Technology Review“ [Tec05] wurden 1190 Mitarbeiter von Firmen verschiedener Branchen befragt. Dabei „attestierten 70% der Befragten dem Faktor Usability einen hohen bis sehr hohen Einfluss auf den Markterfolg eines Produktes“. Ein Grund für diese Ansicht liegt darin, dass mehr als die Hälfte der Befragten Erfahrungen mit herausgebrachten Produkten mangelnder Gebrauchstauglichkeit hat, die in vielen Fällen überarbeitet werden mussten oder ganz eingestellt wurden. Somit ergeben sich durch den Einsatz von Usability-Methoden neben einer erhöhten Nutzerzufriedenheit auch ökonomische Vorteile, wie ein erhöhter „return on investment“ [PRD07 (S.242)].

Im Bereich der Evaluation traditioneller Arbeitsplatzrechner stehen zahlreiche Methoden zur Evaluation der Usability bereit [Nie93, IH01], die über die Jahre hinweg entwickelt und verbessert wurden. Die Art und Weise der Nutzerinteraktion in Smart Environments[1] unterscheidet sich allerdings von der in traditionellen Applikationen [CD04, PRD07]. So können Smart Environments multimodale, in die Umgebung eingebettete Benutzungsschnittstellen enthalten. Entsprechend des aktuellen Kontextes sagt das System die Intentionen der Nutzer vorher und bietet entsprechende Assistenz an. Nutzer erwarten zudem, dass sie das System direkt und ohne Bedienungsanleitung nutzen können [Hol05].

Durch diese Veränderungen in der Mensch-Maschine-Interaktion ergeben sich auch neue Herausforderungen im Bereich der Usability und entsprechender Evaluationsmethoden. Der Bau von Smart Environments ist in der Regel teurer als der von traditionellen Applikationen und meist handelt es sich um Forschungsprototypen mit Spezialhardware. Baut man zunächst eine komplette physische Umgebung, um dann summativ Nutzerstudien durchzuführen, muss man nachträgliche Änderungen mit einem hohen Aufwand erkaufen (1:10:100-Regel). Eine Lösung bietet die iterative Entwicklung, bei der sich Entwicklung und Evaluation abwechseln, um kontinuierlich Rückmeldungen der Nutzer in den Entwicklungsprozess einfließen zu lassen. Es wird ein Vorgehensmodell vorgestellt, das Prinzipien der

[1] Smart Environments sind physische Umgebungen, die auf Aktivitäten der Nutzer reagieren, um ihnen Assistenz bei der Erreichung ihrer Ziele zu bieten. (vgl. [AE06] und Kapitel 2.3)

benutzerorientierten Gestaltung („user-centred design“) [DIN99] auf Smart Environments überträgt. Dafür benötigte Evaluationsmethoden für frühe Phasen der Entwicklung sind für Smart Environments kaum verfügbar [CM04], obwohl diese für die Evaluation traditioneller Applikationen sehr beliebt sind.

Darüber hinaus spielen Kontextinformationen für die Interpretation von Interaktionen in Smart Environments eine entscheidende Rolle. Dies muss auch bei Evaluationsmethoden berücksichtigt werden. Cook und Das [CD04] stellen fest, dass häufig der Ort eine besondere Berücksichtigung findet, während der Aufgabenkontext vernachlässigt wird. Die Einbeziehung von Aufgabenmodellen in den Entwicklungs- und Evaluationsprozess gibt der vorliegenden Arbeit eine aufgabenorientierte Sicht.

Ein anderes häufig auftretendes Problem bei der Evaluation [NSK+08] ist die Frage nach dem Umgang mit persönlichen Daten. Besonders bei Nutzertests müssen umfangreiche Kontextdaten protokolliert werden. Ein Ansatz zur anonymisierten Wiedergabe von aufgezeichneten Nutzerinteraktionen wird dazu vorgestellt.

1.2 Zielstellung der Arbeit

Das Ziel der vorliegenden Arbeit besteht darin, ein Vorgehensmodell für die aufgabenmodellbasierte Evaluation von Smart Environments zu entwickeln. Dabei wird folgendes angestrebt:

Spezifikation von Anforderungen an die Usability-Evaluation von Smart Environments

Im Rahmen einer Analyse der Besonderheiten von Smart Environments sollen Usability-Probleme aufgezeigt und Herausforderungen bei der Evaluation ermittelt werden. Nach einer Untersuchung vorhandener Evaluationsmethoden werden Anforderungen für das zu entwickelnde Vorgehensmodell abgeleitet.

Entwicklung eines Vorgehensmodells für die Usability-Evaluation von Smart Environments

Basierend auf den aufgestellten Anforderungen soll ein Vorgehensmodell für die Usability-Evaluation von Smart Environments entwickelt werden, dass eine Evaluation begleitend zum Entwicklungsprozess gewährleistet.

Steigerung der Effizienz der Evaluation durch Werkzeugunterstützung

Entsprechend der festgelegten Anforderungen soll eine Werkzeugunterstützung entwickelt werden, die die Evaluation innerhalb des iterativen benutzerorientierten Gestaltungsprozesses von Smart Environments unterstützt. Dabei wird Wert gelegt auf eine Integration der Werkzeuge für Entwicklung und Evaluation, um kürzere Iterationszyklen zu ermöglichen.

1.3 Wissenschaftlicher Beitrag

Die Ergebnisse der vorliegenden Arbeit umfassen:

- Eine umfassende Literaturrecherche gibt einen Überblick über den aktuellen Stand der Forschung im Bereich der Evaluation von Smart Environments. Auf dieser Basis werden Herausforderungen bestehender Evaluationsmethoden aufgedeckt, um daraus Anforderungen für ein zu entwickelndes Vorgehensmodell abzuleiten.
- Ein Vorgehensmodell zur Evaluation der Usability von Smart Environments wird entwickelt, das die Evaluation von Artefakten beschreibt, die während der Anforderungsanalyse, des Designs und der Implementation entstehen. Die Integration von Entwicklung und Evaluation erlaubt es Entwicklern und Usability-Experten, auf den gleichen Artefakten zu arbeiten und dadurch sowohl schnell Modelle für eine Evaluation aufzubereiten, als auch Ergebnisse nahtlos in deren Verbesserung zurückfließen zu lassen.
- Als Werkzeugunterstützung wird ein virtuelles Smart Environment entwickelt, das das reale ergänzt und in den verschiedenen Entwicklungsphasen für die Evaluation verwendet wird. Dadurch wird sowohl eine Umgebung für die Durchführung der Evaluation geboten, als auch für die Auswertung der Ergebnisse mit zahlreichen Filter-, Aggregations-, Normalisierungs- und Visualisierungsmöglichkeiten. Das virtuelle Smart Environment unterstützt folgende Evaluationen

  1. Animation von kooperativen Aufgabenmodellen mit Kontextinformationen für Expertenevaluation durch interaktives Durchlaufen („Walkthrough“)
  2. „Wizard of Oz“-Experimente, um Nutzern eine Vorschau auf die gewünschte Assistenz zu geben
  3. Generierung von künstlichen Sensordaten zur Evaluation einzelner implementierter Komponenten
  4. Anonymisierte Visualisierung von aufgezeichneten Daten aus Nutzertests des komplett eingerichteten physischen Smart Environments

- Da die manuelle Auswertung der aufgezeichneten Daten vieler Probanden aufwendig ist, wird die Spezifikation von erwartetem Nutzerverhalten in Smart Environments eingeführt. Usability-Experten können auf diese Weise implizites Wissen über ein zielführendes Nutzerverhalten für eine bestimmte Testaufgabe explizit beschreiben. Während der Evaluation kennzeichnet das System Interaktionen, die von der Erwartung abweichen. Eine manuelle Detailanalyse kann auf diese Vorinformationen zurückgreifen, um Probleme schneller zu beheben.
- Eine Nutzerstudie berichtet abschließend über Erfahrungen beim Einsatz des virtuellen Smart Environments.

1.4 Aufbau der Arbeit

Abbildung 1.1 zeigt die Struktur der vorliegenden Arbeit, die sich wie folgt gliedert:

Kapitel 2 führt den Begriff „Usability“ ein und gibt einen Überblick über Evaluationsmethoden für traditionelle interaktive Systeme. Basierend auf einer Charakterisierung von Smart Environments werden auch hierfür Evaluationsmethoden vorgestellt und ihre Anwendung in konkreten Projekten.

Kapitel 3 nennt Anforderungen an die Evaluation von Smart Environments und leitet daraus ein Vorgehensmodell zu deren Evaluation ab, das mit den Phasen (1) Planung, (2) Entwicklung, (3) Evaluation und (4) Verbesserung, zugleich die innere Struktur von Kapitel 4-6 vorgibt. Darüber hinaus wird die Softwarearchitektur des entwickelten virtuellen Smart Environments skizziert.

Die Kapitel 4-6 sind entsprechend des Vorgehensmodells jeweils strukturiert in die Unterkapitel „Entwicklungsmethode“, „Usability-Methode“ und „Verbesserung der Usability“.

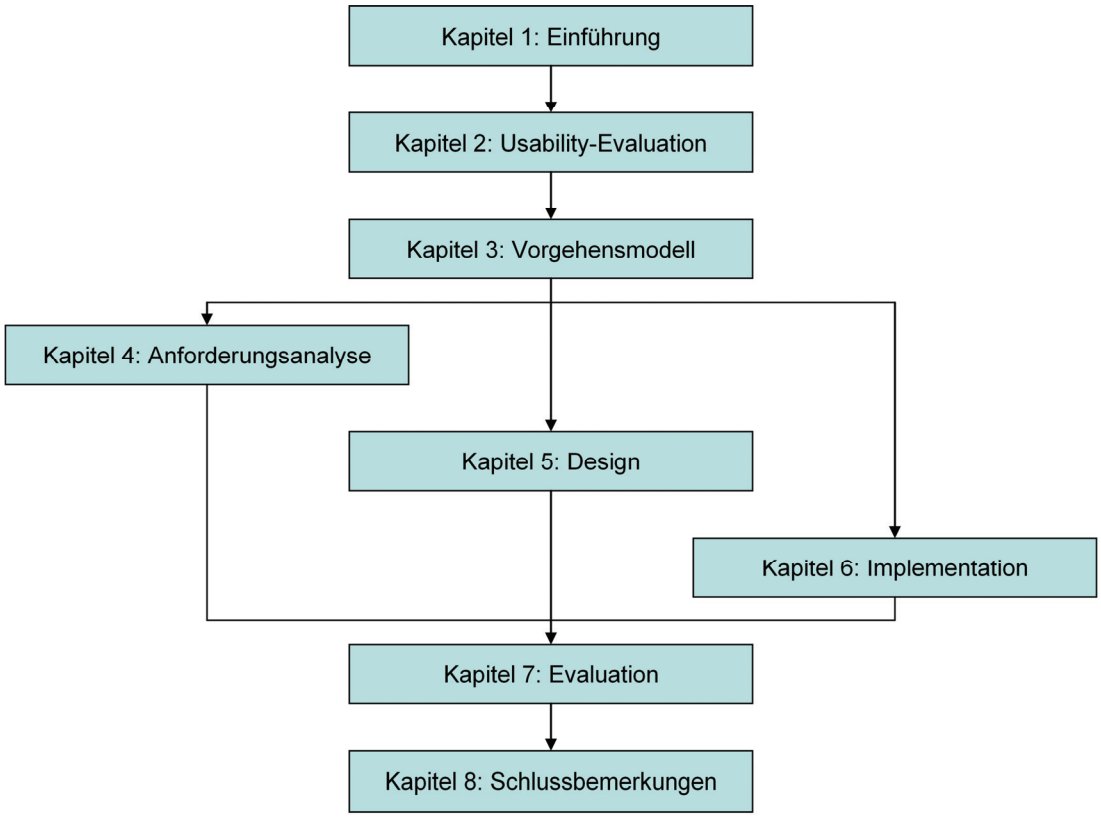

Abbildung 1.1: Abhängigkeitsgraph der Kapitel der vorliegenden Arbeit

Kapitel 4 erläutert das Vorgehen während der Phase der Anforderungsanalyse. Erstellte Aufgabenmodelle werden animiert und Anforderungen werden als „Wizard of Oz“-Experiment evaluiert.

Kapitel 5 erläutert das Vorgehen während der Designphase. Verfeinerte Aufgabenmodelle können mit Kontextbezug und kooperativen Aspekten animiert und evaluiert werden.

Kapitel 6 erläutert das Vorgehen während der Implementationsphase. Implementierte Komponenten des Smart Environments können separat mit Hilfe künstlicher Sensordaten evaluiert werden. Informationen aus Nutzerstudien werden im virtuellen Smart Envionment visualisiert.

Kapitel 7 berichtet über erste Erfahrungen bei der Anwendung des Vorgehensmodells. Eine Nutzerstudie evaluiert die Arbeit mit dem virtuellen Smart Environment in der Designphase. Eine anschließende Expertenevaluation liefert Erfahrungen bei der Erstellung künstlicher Sensordaten während der frühen Implementationsphase.

Kapitel 8 fasst die vorliegende Arbeit zusammen und gibt einen Ausblick.

Derzeit vorhandene Ansätze zur Evaluation von Smart Environments decken jeweils nur bestimmte Teilaspekte ab. Bezüge zum aktuellen Stand der Forschung sind daher in jedem Kapitel individuell erläutert.

Kapitel 2

Usability und Usability-Evaluation

Dieses Kapitel erläutert die grundlegenden Begriffe aus den Bereichen Usability und Smart Environments. Es geht auf die Besonderheiten der Interaktion in Smart Environments ein und leitet daraus Herausforderungen für deren Evaluation ab. Zur Durchführung der Evaluation werden einerseits Methoden aus dem Bereich klassischer Arbeitsplatzrechner vorgestellt, die über die Jahre hinweg entwickelt wurden und in teils adaptierter Form für Smart Environments verwendet werden, und andererseits Methoden aus dem Bereich ubiquitärer Systeme, die momentan in der Entstehung sind. Ein Überblick von Projekten zum Bau von Smart Environments und deren Evaluation illustriert, welche Evaluationsmethoden praktisch eingesetzt wurden und welche Phasen von Entwicklung und Evaluation durchlaufen wurden.

2.1 Usability

Der englische Begriff „Usability“ wird mit dem Begriff „Gebrauchstauglichkeit“ übersetzt und ist definiert als

> *„das Ausmaß, in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen“ [DIN98]*

Die Gebrauchstauglichkeit eines Produktes kann damit nicht absolut bestimmt werden, sondern ist abhängig vom Nutzungskontext: konkreten Bedingungen, unter denen man mit dem Werkzeug ein bestimmtes Ziel erfüllen kann. Gleichzeitig ist Zufriedenheit subjektiv und kann von verschiedenen Probanden anders wahrgenommen werden.

DIN EN ISO 9241-110 [Din06] beschreibt 7 Kriterien für Entwurf und Bewertung von gebrauchstauglichen Benutzungsschnittstellen:

- Aufgabenangemessenheit: der Nutzer wird bei der Erledigung seiner Arbeitsaufgabe unterstützt.
- Selbstbeschreibungsfähigkeit: der Nutzer erkennt jederzeit, an welcher Stelle im Dialog er sich befindet und welche Interaktionsmöglichkeiten bestehen.
- Erwartungskonformität: der Dialog entspricht den Belangen der Nutzer im Nutzungskontext, sowie allgemein anerkannten Konventionen.
- Lernförderlichkeit: der Nutzer wird beim Erlernen des Systems unterstützt.
- Steuerbarkeit: der Nutzer kann den Dialog in Richtung und Geschwindigkeit selbst steuern.
- Fehlertoleranz: bei falschen Eingaben kann der Nutzer das Ziel mit keinem oder geringem Korrekturaufwand dennoch erreichen.
- Individualisierbarkeit: Nutzer können die Mensch-System-Interaktion ihren individuellen Fähigkeiten und Bedürfnissen anpassen.

Die Berücksichtigung der Aspekte soll zu einer guten Usability von Produkten führen. Die Aspekte sind aber nicht überschneidungsfrei, sondern bergen in sich Zielkonflikte. Ein gut erlernbares Produkt kann bspw. vielfältige Erklärungstexte und ausführliche Dialoge enthalten, was aber bei Experten zu einer verminderten Effizienz führt. Experten hingegen bevorzugen meist präzise Fachtermini für eine effiziente Arbeit. Ein sicherheitskritisches System mit zusätzlichen Sicherheitsabfragen erhöht die Robustheit gegen Nutzerfehler, vermindert aber ebenfalls die Effizienz.

Je nach Art des Produktes wird sich daher die Gewichtung der Aspekte unterscheiden. Beispielsweise spielen die Zufriedenheit bzw. der Spaß bei der Nutzung von Spielesoftware eine übergeordnete Rolle, während die Fehlertoleranz bei sicherheitskritischen Systemen wie Flugzeugsteuerungen außerordentlich wichtig ist und die einfache Erlernbarkeit bei Fahrkartenautomaten im Mittelpunkt steht.

2.2 Methoden im Bereich interaktiver Systeme

Für die Evaluation neuartiger Nutzerschnittstellen werden häufig klassische Verfahren adaptiert und kombiniert, daher sollen diese zunächst kurz im Überblick dargestellt werden. Die Methoden lassen sich in die 5 Klassen Testen, Inspektion, Befragung, analytische Modellierung und Simulation gliedern [IH01 (S.476)], die jeweils mehrere Methodentypen beinhalten.

2.2.1 Testen

Zur Durchführung eines Usability-Tests werden Nutzer eingeladen, die entsprechend eines Testplans [Rub94] Aufgaben an dem zu testenden Produkt durchführen. Durch Beobachtung und Auswertung des Nutzerverhaltens sollen Usability-Probleme identifiziert werden.

Von Vorteil bei Tests in realer Umgebung ist der hohe Grad an Authentizität, da reale Aufgaben durchgeführt werden. Nutzerverhalten wird an Hand tatsächlicher Beobachtungen und Messungen protokolliert. Bei reiner Analyse von Beobachtungsdaten ist allerdings teilweise das Finden der Ursachen schwierig. Dies erleichtern Methoden, bei denen der Nutzer nebenbei Gedanken und Gefühle äußert. Von Nachteil ist allgemein, dass ein entwickelter Prototyp notwendig ist und dass Einladen von Nutzern tendenziell aufwendig ist.

Methodentypen: „Thinking-Aloud Protocol“, „Question-Asking Protocol“, „Shadowing Method“, „Coaching Method”, „Teaching Method“, „Codiscovery Learning“, „Performance Measurement“, „Log File Analysis“, „Retrospective Testing“, „Remote Testing“

2.2.2 Inspektion

Bei Inspektionsmethoden vergleicht der Evaluierende das Produkt mit Richtlinien, um die Konformität zu analysieren. Richtlinien reichen dabei von abstrakten Prinzipien bis hin zu konkreten Designrichtlinien (wie bspw. Schriftgrößen oder Farben).

Von Vorteil ist die Durchführbarkeit bereits in frühen Phasen der Entwicklung, wenn bspw. nur Oberflächenentwürfe verfügbar sind. Von Nachteil ist die Abhängigkeit vom Urteilsvermögen des Evaluierenden. So hat bspw. eine Studie [Sea95] gezeigt, dass teils visuell ansprechende Oberflächenentwürfe den Vorzug gegenüber effizienteren erhalten.

Methodentypen: „Guideline Review“, „Cognitive Walkthrough“, „Pluralistic Walkthrough“, „Heuristic Evaluation“, „Perspective-Based Inspection“, „Feature Inspection“, „Formal Usability Inspection“, „Consistency Inspection“, „Standards Inspection“

2.2.3 Befragung

Nutzer werden zum Produkt befragt und äußern ihre Meinung. Es können wahlweise einzelne oder mehrere Nutzer einbezogen werden, ein persönliches Interview stattfinden oder ein Papierfragebogen ausgefüllt werden. Zur visuellen Unterstützung können Bildschirmausschnitte verwendet werden.

Von Vorteil ist, dass subjektive Eindrücke der Nutzer erfasst werden, die im Gegensatz zu Ergebnissen anderer Methoden wie bspw. Logdaten häufig einfacher zu interpretieren sind. Von Nachteil ist, dass eine Beeinflussung durch die Art der Fragestellung möglich ist.

Methodentypen: „Contextual Inquiry“, „Field Observation“, „Focus Groups“, „Interviews“, „Surveys“, „Questionnaires“, „Self-Reporting Logs“, „Screen Snapshots“, „User Feedback“

2.2.4 Analytische Modellierung und Simulation

Der Evaluierende verwendet Modelle von Nutzern und Benutzungsschnittstellen um Vorhersagen der Usability zu treffen. Dabei kann bei manchen Methoden das Verhalten des simulierten Nutzers direkt als Eingabe für das zu testende Produkt verwendet werden, um die Ausgaben zu analysieren.

Von Vorteil ist die gute Automatisierbarkeit, die den Test mehrerer Entwürfe im Vergleich und mit wenig Aufwand erlaubt. Von Nachteil ist die Abhängigkeit von der Qualität des Modells. Erkannte Probleme des Produktes können auch auf eine Schwäche im Modell zurückzuführen sein.

Methodentypen: „GOMS Ananlysis“, „UIDE Analysis“, „Cognitive Task Analysis“, „Task-Environment Analysis“, „Knowledge Analysis“, „Design Analysis“, „Programmable User Models“, „Information Process Modeling“, „Petri Net Modeling“, „Genetic Algorithm Modeling“, „Information Scent Modeling“

2.3 Methoden im Bereich Ubiquitous Computing

Nachfolgend wird das Paradigma des „Ubiquitous Computing“ eingeführt und das darin verwendete Konzept des „Smart Environments“ definiert. Es wird erläutert, welche besonderen Probleme der Usability und deren Evaluation bestehen. Anschließend wird der aktuelle Stand der Wissenschaft für die Usability-Evaluation dargestellt. Dieser umfasst generelle Rahmenwerke der Usability-Evaluation, konkrete Methoden und einen Überblick über konkrete Projekte zum Bau von Smart Environments mit einer Darstellung der verwendeten Methoden.

2.3.1 Ubiquitous Computing

2.3.1.1 Evolution der Computertechnik

Die Bedeutung der Computertechnik hat sich seit der Entwicklung der ersten Computer stark verändert [HRR+08]. Betrachtet aus der Perspektive der verteilten Nutzung von Computern unterscheidet Waldner [Wal06] 5 Phasen der Evolution:

1. „Geburt der Informatik“: Erste Computer wurden als eigenständige Rechenmaschinen konstruiert, die stationär betrieben wurden.
2. „kommunizierende Computer“: Durch die Einführung von Kommunikationstechnologien wurde es möglich Computer miteinander zu verbinden und Ressourcen gemeinsam zu nutzen.
3. „Einführung der Mobilität“: Die zunehmende Verkleinerung der Komponenten und die Einführung von Akkumulatoren zur Energieversorgung ermöglichte die Konstruktion von mobilen Computern beginnend ab den 1980er Jahren.
4. „ubiquitäre Systeme“: Eine zunehmende Anzahl von Geräten mit Computertechnik ist im Alltag vorhanden.
5. „ambiente Intelligenz“: Eine große Zahl kommunizierender Geräte bevölkert unseren Alltag und versucht diesen intelligent zu bereichern.

Betrachtet aus der quantitativen Perspektive der Anzahl der Nutzer im Vergleich zur verfügbaren Anzahl von Computern lässt sich diese Entwicklung ebenfalls nachzeichnen [WB96]: Erste Computer während der „Mainframe-Ära“ waren wegen ihrer Größe und hohen Kosten selten, so dass mehrere Nutzer einen Computer verwendeten. Die nachfolgende Ära der „Personal Computer“ machte jedem Nutzer einen eigenen Computer zugänglich, während seit dem Einzug des „Ubiquitous Computing“ jeder Nutzer mehrere Computer nutzen kann.

2.3.1.2 Paradigma „Ubiquitous Computing“

Anfang der 1990er Jahre wurde der Begriff „Ubiquitous Computing“ durch Mark Weiser als neues Paradigma geprägt:

> *„The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.“ [Wei91]*

Während bisher Geräte im Alltag bewusst und als eigenständige Einheiten wahrgenommen wurden, sollen einzelne Geräte nach diesem Paradigma in den Hintergrund treten. Menschen, die eine Aufgabe in diesem Umfeld durchführen, sollen möglichst natürlich handeln, ohne sich auf die Bedienung spezifischer Geräte konzentrieren zu müssen. Weiser nennt 3 technologische Voraussetzungen: Notwendig sind preiswerte energiesparende Computer, ausreichend ubiquitäre Applikationen und ein umfassendes Netzwerk, um alles zu verbinden. Dix et al. charakterisieren „Ubiquitous Computing“ aus der Perspektive menschlicher Interaktionen als: „Any computing activity that permits human interaction away from a single workstation“ [DFA+04].

Neben dem Begriff des „Ubiquitous Computing“ finden sich weitere Begriffe, wie bspw. „Pervasive Computing“ [SM03], „Ambient Intelligence“ [Aar04] und „Everyware“[Gre06]. Diese haben große Überschneidungen miteinander und werden teils sogar synonym verwendet. Bspw. arbeiten Niemelä et al. [NL04] feine Unterschiede in den Sichtweisen von „Ubiquitous Computing“ und „Pervasive Computing“ heraus, entscheiden sich aber für den Konferenzbeitrag beide Begiffe synonym zu verwenden. Waldner [Wal06] hingegen sieht die Begriffe „Ambient Intelligence“ und „Ubiquitous Computing“ als verschieden. Die Verwendung der Begriffe im wissenschaftlichen Diskurs hat sich noch nicht abschließend konsolidiert. In der vorliegenden Arbeit wird der Begriff des „Ubiquitous Computing“ durchgängig verwendet.

2.3.1.3 Smart Environments

Ein zentrales Konzept der genannten Begriffe ist das „Smart Environment“, welches definiert wird durch:

> *„Smart Environments are physical spaces that are able to react to the activities of users, in a way that assists the users in achieving their objectives in this environment." (Kirste in: [AE06])*

Das Adjektiv „smart" kennzeichnet die Fähigkeit der Umgebung auf Nutzerinteraktionen zu reagieren. Dafür werden über die reinen Sensordaten hinaus Nutzerziele abgeleitet, um entsprechend die passenden Aktionen durch vorhandene Geräte auszuführen. Dafür ist es notwenig, den Inhalt der Nutzerinteraktionen so gut wie möglich zu verstehen.

Die so gebotene Assistenz soll den Nutzer in möglichst unauffälliger Weise bei den gewohnten Arbeitsabläufen unterstützen, bspw. die nächsten Aufgaben vorbereiten und automatisierbare Tätigkeiten selbständig ausführen. Für den Nutzer soll damit ein neues Nutzungserlebnis („user experience") entstehen, das die Arbeit mit dem System angenehm gestaltet und Freude bei der Benutzung bereitet. Weiser beschreibt die Vision als:

> *„Machines that fit the human environment instead of forcing humans to enter theirs will make using a computer as refreshing as a walk in the woods." [Wei91]*

Der Begriff „Smart Environment" umfasst ein ganzes Spektrum von Anwendungen, insbesondere als „smart home", „smart office", „smart classroom", „smart car" und „smart cloth" [CD04]. Der Fokus dieser Arbeit liegt auf der Betrachtung der Interaktion von Personen in Räumen bzw. Gebäuden. Dazu werden Besprechungsräume als konkrete Beispiele betrachtet. Rienks et al. [RNB08] geben einen Überblick darüber, wie vorausschauende, proaktive Assistenz vor, während und nach einer Besprechung ausgestaltet werden kann und verweisen auf die entsprechenden Projekte. Maes [Mae94] gibt allerdings zu bedenken, dass ein System den Umfang der Assistenz von der Sicherheit bei der Einschätzung der Situation machen sollte.

Smart Environments lassen sich durch spezifische Eigenschaften charakterisieren, die sie von klassischen Desktop-Computern unterscheiden. [CD04]:

- Einbettung: Rechenleistung ist eingebettet in die natürliche Alltagsumgebung der Nutzer.
- Mobilität: Nutzer können sich frei bewegen und dabei beliebige Geräte mitführen. Die Konkurrenz um den begrenzten Platz auf dem Schreibtisch hat sich nun verlagert auf den begrenzten Platz an der „Gürtelschnalle" der Nutzer.
- Permanenz: Ubiquitäre Systeme laufen permanent ohne Unterbrechung, im Gegensatz zu Desktop-Systemen, die für eine Aufgabe gestartet werden und anschließend abgeschaltet werden können. Dies erschwert die Wartung und stellt Anforderungen an die Evolution der Komponenten.
- Sensoren/Aktoren: Die einzelnen Knoten, die die Rechenleistung bereitstellen, besitzen zugleich Sensoren und Aktoren, so dass sie Änderungen der Umwelt wahrneh-

men und diese gleichzeitig auch beeinflussen können. Personen werden dabei häufig durch das Tragen von Etiketten („Tags“) oder durch Biometrie erkannt [MJG08].

- Interaktion: Charakteristisch ist die permanente Mensch-Maschine-Interaktion. Da viele Tätigkeiten automatisiert werden ist eine nahtlose Verzahnung zwischen menschlichen und maschinellen Tätigkeiten entscheidend, um einen kontinuierlichen Arbeitsablauf zu gewährleisten.

2.3.1.4 Projekte zum Bau von Smart Environments

Im Rahmen vielfältiger Projekte sowohl aus der Wissenschaft als auch der Industrie wurden Smart Environments realisiert. Cook und Das geben in [CD04 (S.391f)] einen Überblick über eine Reihe von Projekten. Da der Schwerpunkt dieser Arbeit auf den Aspekten der Evaluation liegt, sollen nachfolgend nur einige Projekte genannt werden:

Adaptive House, Agent-based Intelligent Reactive Environments (AIRE), Ambiente Roomware, AVIARY, Aware Home, Changing Places/House_n, Cisco Internet Home, Edinvar Assisted Interactive Dwelling HOUSE, Ericsson IT Apartments, Essex Intelligent Inhabited Environments Group (IIEG), Gloucester Smart House, IBM wired home, Icepick Technologies, Intel Proactive Health, Intelligent Building Group (EIBG), Intelligent Home Project, Intelligent Space Project, Internet Home Alliance, MavHome, Microsoft Easy Living, Philips smart home, PRIMA, Smart Homes Foundation, Stanford Interactive Workspaces und Sun Dot Com Home.

Eckl et al. [EM09] stellen fest, dass viele der bestehenden Smart Environments sich nicht flächendeckend bei Nutzern durchgesetzt haben. Um Ursachen dafür zu finden, ordnen sie die Projekte in eine Matrix mit den beiden Dimensionen „Anzahl der Geräte / Funktionen“ und „Grad der Automatisierung“ ein. In den Randbereichen unterscheiden sie drei Problembereiche:

- (1) Zu einfach. Smart Environments mit wenig Automatisierung und wenig Funktionalität sind zwar für Nutzer gut bedienbar, aber decken meist ein zu kleines Anwendungsfeld ab, so dass teils auch relevante Funktionen unberücksichtigt bleiben. In diesen Bereich fallen viele Industrieprojekte.
- (2) Zu komplex. Smart Environments mit viel Funktionalität versuchen möglichst viele Anwendungen abzudecken, führen aber häufig zu komplexer Konfiguration. In diesen Bereich fallen bspw. manuell zu konfigurierende UPnP-Netzwerke.
- (3) Zu intelligent. Smart Environments mit einem hohen Grad der Automatisierung nehmen den Nutzern fast alle eigenen Entscheidungen ab, machen dabei aber mit zunehmendem Automatisierungsgrad auch Fehler.

Eckl et al. [EM09] werfen die Frage nach der richtigen Balance zwischen zu wenig und zu viel Assistenz auf, bzw. zwischen einem zu kleinen, überschaubaren und einem zu umfangreichen Anwendungsfeld.

Intille [Int02] beschäftigte sich ebenfalls mit dem Grad der Assistenz und erkannte, dass die Gefahr besteht, dass Menschen durch zu viel Assistenz geistig und körperlich abbauen.

Rodin et al. [RL77] konnten in einer Studie mit älteren Menschen zeigen, dass diese bei einem gewissen Maß an Eigenverantwortung zufriedener und glücklicher waren.

Im Rahmen des Embassi-Projektes [HK02] stellten Nitschke et al. [NW02] fest, dass Entwickler die Frage nach der angemessenen Assistenz häufig „ad hoc oder rein technologiegeleitet“ beantworten. Um dies zu vermeiden leitet das Vorgehensmodell GUIDEAS [NW02] Entwickler durch den Entwicklungsprozess. Nachdem Fragen zu Zielen der Assistenz, Merkmalen der Nutzer und zu unterstützenden Aufgaben beantwortet sind, macht das GUIDEAS-Werkzeug Vorschläge, wie die Assistenz ausgestaltet werden sollte. Grundlage für diese Entscheidungen sind experimentelle und empirische Erkenntnisse aus der Psychologie.

2.3.1.5 Kontext

Der Begriff Kontext ist zentral für Smart Environments, da jede Nutzerinteraktion in ihrem jeweiligen Kontext zu betrachten ist, um die vorliegende Situation zu erkennen und daraus die notwendige Assistenz abzuleiten. Eine weit verbreitete Definition stammt von Dey et al.:

> *„Any information that can be used to characterize the situation of entities (i.e. whether a person, place or an object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. Context is typically the location, identity and state of people, groups and computational and physical objects“ [DA04]*

In der Literatur werden je nach Autor und Applikation sehr verschiedene Aspekte betrachtet. Schmidt [Sch00] nennt bspw. die Einflussfaktoren Gerät, Zeit, Nutzer, physische Umgebung (wie Ort, Temperatur), soziale Situation und Aufgabe. Abowd et al. [AM00] stellen fest, dass eine komplette Definition von Kontext illusorisch ist und nennen stattdessen die 5 „W“s als eine minimale Menge des notwendigen Kontexts: wer, was, wo, wann und warum. Poslad [Pos09] unterscheidet die drei Dimensionen: physischer, menschlicher und informationstechnologischer Kontext. Die Vielfalt der Sichtweisen spiegelt sich auch darin wieder, dass viele Projekte ein jeweils eigenes Kontextmodell entwerfen. [PBW+04] erläutert beispielhaft eine Reihe von Kontextontologien für intelligente Umgebungen und entwirft selbst zusätzlich eine weitere.

Wenn man diese Kontextinformationen als Eingabe ergänzend oder alternativ zu den traditionellen Eingaben über Maus und Tastatur verwendet ergeben, sich Vorteile für die Mensch-Maschine-Interaktion. Zum einen sind weniger explizite Eingaben nötig, da bereits einiges aus dem Kontext abgeleitet werden kann. Auch bei schlecht erkennbaren Eingaben, wie bspw. bei der optischen Zeichenerkennung, wird der Eingaberaum verkleinert und damit die Erkennung verbessert. Zum anderen kann die Auswahl für Nutzereingaben verkleinert werden, da bspw. nur die Auswahlmöglichkeiten angezeigt werden müssen, die im aktuellen Kontext sinnvoll sind.

2.3.1.6 Usability-Probleme in Smart Environments

Durch die besonderen Eigenschaften von ubiquitären Anwendungen, speziell Smart Environments, ergeben sich spezifische Usability-Probleme, die über die von klassischen Arbeitsplatzrechnern hinausgehen [CD04]. Ursachen hierfür sind:

Eingebettete Benutzungsschnittstellen. In der Aufgabenanalyse wird eine Unterscheidung in interne und externe Aufgaben vorgenommen [Her09 (S.21)]. Externe Aufgaben sind problemorientiert, unabhängig von der Applikation auf die Sache selbst fokussiert, bspw. das Zeichnen einer Bleistiftskizze. Interne Aufgaben dagegen sind technikorientiert, vom jeweiligen Werkzeug abhängig, bspw. das Anspitzen eines Bleistifts. Bei einem Druckbleistift würde diese Teilaufgabe wegfallen. Die in Smart Environments handelnden Menschen sollen sich möglichst gut auf ihre eigentliche externe Aufgabe konzentrieren können, während die gebotene Assistenz begleitende interne Aufgaben erledigt, bspw. Projektoren einschaltet, Leinwände herunterfährt oder Videoverbindungen zwischen Rechnern und Projektoren herstellt. Allerdings wird der gebotene Komfort gleichzeitig durch Probleme erkauft. Ein Aspekt ist das Verschwinden einer offensichtlichen Benutzungsschnittelle. Diese ist gut in die Umgebung integriert und damit teils unsichtbar (bspw. Mikrofone für Spracherkennung) oder zumindest nicht als solche zu erkennen (bspw. eine Tür, nach deren Öffnen ein Assistenzsystem das Raumlicht einschaltet). Dadurch bleibt für Nutzer unklar, worin genau ihre Eingabe besteht, was also das System wahrnimmt. Wenn das System unerwartet reagiert ist nicht immer klar, auf Basis welcher Eingaben das Verhalten beruhte. Eine instrumentierte Umgebung lässt sich kaum von einer nicht-instrumentierten unterscheiden, so dass auch gewünschte Assistenzfunktionen anwesende Nutzer überraschen können und ein Gefühl des Beobachtetseins bleiben kann. Das Smart Environment reagiert auf eine Nutzerinteraktion je nach aktuellem Kontext anders. Damit ist für Nutzer eine Systemreaktion auf ein bestimmtes Verhalten kaum vorhersagbar.

Sensordatenflut. Smart Environments nehmen das Verhalten der anwesenden Nutzer wahr und leiten daraus ab, welche Aufgaben die Nutzer gerade bearbeiten und welche sie wahrscheinlich als nächstes beginnen werden. Entsprechend werden Geräte angesteuert, um Assistenz bei diesen Handlungen zu bieten. Die Wahrnehmung von Nutzerverhalten basiert auf Sensorwerten aus der Umgebung. Durch die riesige Menge von Sensordaten und potentielle Ungenauigkeiten wird deren Interpretation erschwert, was eine ungeeignete Assistenz bezogen auf den aktuellen Kontext zur Folge haben kann.

Multimodalität. Während Nutzer klassischer Applikationen in ihren Interaktionen beschränkt sind auf eine geringe Anzahl an Modalitäten, zumeist Maus und Tastatur, ist in Smart Environments die Vielfalt fast unbegrenzt [JS07]. Anwendung finden bspw. Sprache, Ortsveränderung und die Berührung von Gegenständen. Dadurch entsteht eine riesige Menge von Kombinationsmöglichkeiten, die das Design eines Assistenzsystems und die Interpretation von Nutzerinteraktionen erschweren. Die Herausforderung für das System besteht darin, die relevanten Kommunikationskanäle auszuwählen und in ihrer Kombination zu interpretieren. Aktuelle Projekte beschränken sich daher auf die Unterstützung eines bestimmten Anwendungsszenarios, wie bspw. Besprechungen, Krankenpflege oder multimediale Erlebnisse. Im Graduiertenkolleg MuSAMA steht ein Besprechungsraum im Mittelpunkt.

Permanente Veränderungen. Ein Smart Environment ist ein ad hoc Ensemble aus Nutzern, Geräten und der räumlichen Umgebung. Mit jeder Entität, die die Umgebung betritt oder verlässt ändert sich das Smart Environment und der enstprechende Funktionsumfang. Im alten Kontext mögliche Aufgaben fallen weg, wenn das entsprechende Gerät verschwindet, und neue Aufgaben kommen hinzu. Probleme sind sowohl die technische Integration neuer Entitäten in die laufende Assistenz, als auch eine Umstellung der Nutzer auf diese Veränderungen.

Erwartungen der Nutzer. Neben den neuartigen Interaktionstechniken unterscheidet sich auch die Erwartungshaltung der Nutzer gegenüber einem Smart Environment: „Users of such devices expect to be able to run such applications with no training, no traditional packaging elements such as 'quick start' cards, and no help system" [Hol05]. Dadurch werden hohe Erwartungen an die Usability geweckt. Entsprechend der Matrix von Albert et al. [AD03], die tatsächliche und erwartete Usability in Beziehung setzt, verstärkt dies die Dringlichkeit, entsprechende Usability-Probleme zu identifizieren und zu beheben.

2.3.1.7 Ethische, rechtliche und soziale Implikationen in Smart Environments

Neben den genannten Usability-Problemen bestehen auch ethische, rechtliche und soziale Probleme für Nutzer von Smart Environments [CBM04]. Deren adäquate Lösung kann gleichzeitig zur Achillesferse für die Verbreitung ubiquitärer Technologien werden [CD04 (S.318)] Usability-Evaluationen können auch helfen, Fortschritte in diesen Gebieten zu beurteilen.

Privatheit / Vertrauen. Um Assistenz für Menschen anbieten zu können, muss die Umgebung möglichst viel über die anwesenden Menschen und Geräte wissen. Sensoren bestimmen bspw. Positionen, verwendete Geräte, mitgeführte Dokumente oder machen Videoaufzeichnungen. Diese Daten stammen aus der Privatsphäre der Beteiligten und eine Weitergabe an Dritte ist häufig unerwünscht. Einzelne Datensätze für sich genommen sind möglicherweise unbedenklich, während teils aber das Zusammenführen verschiedener Quellen nicht erwünschte Informationen preisgibt. Auch in einer vertrauenswürdigen Umgebung sind Datenverluste durch technische Defekte möglich. Selbst wenn alle Daten einem gewissen Datenschutz unterliegen, also alle Daten der Nutzerintention entsprechend verwendet werden, kann ein Gefühl des „Beobachtet-Seins" bleiben. Aus diesen Gründen kann ein Misstrauen gegenüber der umgebenden Computertechnik entstehen, das es abzubauen gilt.

Überschreitung von Grenzen. Die Trennung zwischen Menschen und der sie umgebenden Technik wird zunehmend kleiner [Aar04]. So werden künstliche Gelenke und Herzschrittmacher seit Jahren erfolgreich eingesetzt, in Kleidung werden technische Geräte direkt integriert und mobile Geräte wie Mobiltelefone, PDAs oder digitale Kameras werden von vielen ständig am Körper getragen. Diese Erweiterung der Funktionen des menschlichen Körpers mit technischen Hilfsmitteln bietet erhöhten Komfort, wirft aber gleichzeitig auch ethische Fragen auf, wie stark man Technik in den menschlichen Körper integrieren darf. Das gilt insbesondere bei Implantationen. Damit verbunden ist das Problem der wachsenden Abhängigkeit von der Technik [Wan08]. Vielfältige Gründe können dazu führen, dass ein

Gerät kurzfristig nicht verfügbar ist, bspw. weil der Akkumulator geladen werden muss oder einfach vergessen wurde, das Gerät mitzunehmen. Durch langfristige Gewöhnung an die gebotene Assistenz können beteiligte Menschen es verlernen, die Handlung manuell durchzuführen, und sind bei Störfällen möglicherweise nicht in der Lage einzugreifen („out of the loop effect“).

Verhaltensänderung. Derzeit sind Smart Environments Gegenstand der Forschung und die meisten Menschen haben keine Erfahrungen mit ihrer Benutzung. Bei ihrer Einführung in den Alltag ergibt sich die Frage, wie Menschen diese akzeptieren und neu entstehende Funktionen in ihre Handlungen integrieren. Ein Beispiel ist das Smart Environment an der Universität Rostock: es besitzt eine große Anzahl von Videoprojektoren und Leinwänden. Um Besprechungen zu unterstützen, können diese frei verwendet werden. Soll bspw. die Architektur eines Gebäudes abgestimmt werden, können mehrere konkurrierende Vorschläge im direkten Vergleich auf verschiedene Leinwände projiziert werden. Nutzer, die aus Gewohnheit nur eine Leinwand verwenden, profitieren von derartigen zusätzlichen Funktionen nicht. Eine Beeinflussung von Nutzern und Umgebung findet damit in beiden Richtungen statt: einerseits erkennt der Raum über Sensoren die Interaktion von Nutzern und passt sich im Sinne der Assistenz an. Andererseits gewöhnen sich Nutzer mittelfristig an ihre Umgebung und bauen sich ein mentales Modell [Her09 (S.51ff.), Nor02 (S.17)] der Funktionen auf. Dies führt dazu, dass Nutzer ihr Verhalten auch ein Stück weit an den Raum anpassen. Ein Beispiel hierfür sind die weit verbreiteten Bewegungsmelder zur Steuerung von Licht. Wenn man sich einem Hauseingang bei Dunkelheit aus einer bestimmten Richtung nähert, geht das Licht automatisch an. Wenn man diesen Zusammenhang als Kette von Ursache und Wirkung erlernt, und gerade Licht benötigt, kann man sich zu einer Position innerhalb der Reichweite des Bewegungsmelders bewegen. Dies kann zu unnatürlichem Handeln entsprechend des erlernten mentalen Modells führen. Fazit daraus ist, dass Nutzer während ihrer Arbeit mit einem Produkt ihre Verhaltensweisen ändern. Die Usability eines Produktes ist abhängig vom Grad der Vorkenntnisse bei der Benutzung [Nie93 (S.43ff.)] [HR98 (S.76ff.)]. Bei Smart Environments beginnen die meisten Menschen als unerfahrene Nutzer und steigern allmählich ihre Vorkenntnisse. Entsprechend der sich ändernden Bedürfnisse muss auch die Interaktion mit der Umgebung angepasst werden. Diesen Prozess der Evolution der Interaktionsmechanismen in Smart Environments sollten Usability-Evaluationen begleiten, um die Akzeptanz bei Nutzern sicherzustellen.

Überforderung. Nach dem Mooreschen Gesetz [Moo65] verdoppelt sich die Rechenleistung alle 1-2 Jahre. Damit wird es möglich, zunehmend komplexere Aufgaben maschinell abzuarbeiten. Die menschlichen Fähigkeiten erhöhen sich nicht in diesem Maße, so dass diese zum begrenzenden Faktor werden. Kritische Faktoren sind menschliche Zeit, Aufmerksamkeit und die Fähigkeit Entscheidungen zu treffen [CD04]. Früher war die Rechenleistung begrenzt und die Benutzungsschnittstelle musste eingeschränkt werden, um den Computer nicht zu stark zu belasten. Heute ist es umgekehrt. Die Funktionalität der Benutzungsschnittstelle muss eingeschränkt werden, um den menschlichen Nutzer nicht zu überfordern.

2.3.2 Evaluation von „Ubiquitous Computing“-Systemen

Aus den im vorherigen Kapitel genannten Besonderheiten von „Ubiquitous Computing“ resultieren besondere Herausforderungen für die Usability, die durch entsprechende Evaluationsmethoden gelöst werden sollen.

2.3.2.1 Herausforderungen der Evaluation von „Ubiquitous Computing“-Systemen

Die Besonderheiten von ubiquitären Applikationen führen neben Problemen für die Usability auch zu Problemen für die Evaluationsmethoden selbst. Neely et al. [NSK+08] haben Diskussionen innerhalb von 5 Usability-Workshops analysiert, die im Rahmen der Konferenzen CHI 2006, Interact 2007, Mobile HCI 2007, UbiComp 2007 und AMI 2007 stattfanden, um die aktuellen Schwerpunkte und zukünftigen Herausforderungen zu ermitteln. Aus diesen und weiteren Standpunkten (bspw. in [PRD07]) wurden die wesentlichen Probleme momentaner Evaluationsmethoden extrahiert und zu folgender Liste konsolidiert:

Teure Prototypen. Bei ubiquitären Applikationen handelt es sich meistens um teure Prototypen neuester Technologie [AMR02], die in einem Labor betrieben werden. Sie stehen nicht in ausreichender Anzahl und Ausgereiftheit zur Verfügung für die Evaluation in realen Alltagsumgebungen. Dadurch sind häufig nur Laborversuche möglich, deren Ergebnisse nicht uneingeschränkt auf den Alltag übertragen werden können.

Kleine Stichprobengröße. Bei medizinischen und militärischen Anwendungen gibt es nur wenige Probanden, während bei Alltagsanwendungen die Probanden oft nur wenig Zeit haben, wenn sie die natürlichen Abläufe nicht merklich unterbrechen wollen. Dadurch wird es schwierig ein statistisch signifikantes Ergebnis zu erhalten.

Verschiedene Domänen. Die Applikationsdomänen sind sehr vielfältig (bspw. Besprechungen, Medizin, Militär). Entsprechend verschieden sind auch die Kontexte der Anwendungen. In mehreren der ausgewerteten Workshops [NSK+08] wurde vorgeschlagen, mehrere Evaluationsmethoden zu verwenden, die verschiedene Perspektiven auf die zu testenden Applikationen erlauben. Eine Empfehlung war es, die Applikation in mehrere Komponenten zu zerlegen, die dann mit den jeweils geeigneten Methoden evaluiert werden.

Authentischer Nutzungskontext. Eine wichtige Rolle spielen Feldversuche in realer Umgebung, die unter den Stichworten „evaluation in the wild“ und „in situ“ diskutiert werden. Allerdings ist die Anzahl der Kontextvariablen sehr groß und manche sind schwer kontrollierbar. Laborversuche bieten eine bessere Kontrolle, während Ergebnisse aus Feldversuchen bessere Einblicke in die reale Benutzung geben.

Privatsphäre der Probanden. Die Erfassung der Kontextparameter erfolgt über Sensoren, die typischerweise bereits Bestandteil der Umgebung sind, da sie für den Betrieb der Anwendung benötigt werden. Beispiele sind Lokalisationssensoren und eingebaute Sensoren in Mobiltelefonen. Dies ist allerdings gleichzeitig ein Eingriff in die Privatsphäre der anwesenden Nutzer. Bspw. bei der Assistenz von Besprechungen in Firmen kann es unerwünscht sein, wenn Videoaufzeichnungen der Evaluation für Menschen außerhalb des Projektteams zugänglich werden. Zudem ist es auch rechtlich problematisch, da eine

Einwilligung der Probanden notwendig ist. Bei der Evaluation von Software auf traditionellen Einzelplatzrechnern kann dieses Problem gelöst werden, wenn nur der Bildschirminhalt gefilmt wird, da hier alle Interaktionen mit dem System stattfinden. Bei Smart Environments kann allerdings die komplette Umgebung einschließlich aller Nutzer für Interaktionen relevant sein. Formen der Anonymisierung können ein Ausweg sein.

Mangelnde Vergleichbarkeit. Die Vergleichbarkeit von Ergebnissen ist schwierig. Einerseits sind die Kontexteinflüsse sehr groß und kaum kontrollierbar. Andererseits gibt es noch keine allgemein akzeptierten Evaluationsmethoden, Datensätze und Szenarien. Derzeit werden häufig traditionelle Methoden adaptiert bzw. in differierenden Kombinationen eingesetzt und unterschiedlich dokumentiert. Whittaker et al. [WTN00] kritisieren, dass viele neu entwickelte Applikationen radikale Neuerungen darstellen und dadurch der gemeinsame Forschungsfokus fehlt. Als Folgen nennen sie Probleme, auf vorhandenen Arbeiten aufzubauen und mangelnde Vergleichbarkeit von Ergebnissen.

Metriken. Bei klassischen Applikationen stehen in quantitativen Analysen Maßzahlen, wie Grad der Aufgabenerfüllung oder benötigte Ausführungszeit, im Vordergrund. Bei ubiquitären Systemen spielen auch zunehmend „user experience“, ethische Aspekte, soziale Beziehungen und ästhetische Ansprüche eine Rolle [PRD07]. Auch hierfür sind Metriken notwendig, um Systeme miteinander vergleichen zu können.

Erlernbarkeit. Poppe et al. [PRD07] stellen fest, dass aktuelle Evaluationen vor allem eine Momentaufnahme der Usability bieten, während langfristige Nutzungsstudien vernachlässigt werden. Dadurch mangelt es an Erkenntnissen über Lerneffekte bei der Nutzung von ubiquitären Applikationen.

2.3.2.2 Generelle Rahmenwerke

Ein Rahmenwerk zur Gliederung von Evaluationstechniken für ubiquitäre Applikationen wurde von Scholtz und Consolvo [SC04] entwickelt. Das Ziel bestand darin, einen generellen Rahmen zur Einordnung bestehender Verfahren zu schaffen. Zu jedem Verfahren wird notiert, welche der vorgegebenen Aspekte es unterstützt. Dadurch soll die Vergleichbarkeit von Methoden und deren Ergebnissen erleichtert werden, so dass Wissenschaftler leichter voneinander lernen können. Zudem soll die Systematisierung es erleichtern, unberücksichtigte neue Aspekte zu entdecken und „Design-Guidelines“ für diese Aspekte zu entwickeln.

Das Rahmenwerk beschreibt 7 Evaluationsbereiche, denen jeweils Metriken zugeordnet sind. Nachfolgend werden die 7 Bereiche kurz erläutert.

- „Attention“: Aufmerksamkeit spielt bei ubiquitären Anwendungen eine wichtige Rolle, da Nutzer zwischen Geräten wechseln und teilweise auf mehrere Displays achten müssen. Durch die Einbettung von Technologie in den Alltag können Eingaben und Ausgaben potentiell überall in der Umgebung stattfinden.
- „Adoption“: Im Bezug auf die Annahme des Produktes lassen sich zwei Typen von Messgrößen unterscheiden: einerseits die zur Messung der tatsächlichen Nutzung und andererseits die zur Vorhersage des Erfolges oder Misserfolges der Produktes.

- „Trust“: Vertrauen wird auf zwei Aspekte zurückgeführt. Privatsphäre einerseits beschreibt den Umgang mit persönlichen Daten und Bewusstsein andererseits bezieht das Verständnis der Nutzer über die Aktivitäten anderer Nutzer mit ein.
- „Conceptual Models“: Experimentelle Beobachtungen haben gezeigt, dass ein gutes mentales Modell vorteilhaft ist für die richtige Einschätzung des Systemverhaltens. In Untersuchungen werden bspw. Vergleiche zwischen mentalem Modell und Systemverhalten durchgeführt.
- „Interaction“: Für die Beurteilung der Interaktionen wird die Gebrauchstauglichkeit, bestehend aus Effektivität, Effizienz und Zufriedenstellung genutzt und um kooperative Aspekte ergänzt.
- „Invisibility“: Die proaktive Assistenz der weitestgehend unsichtbaren Systemkomponenten kann Fehlinterpretationen unterliegen. Daher sind mögliche Evaluationsaspekte die Verständlichkeit der Automatismen, individuelle Nutzervoreinstellungen und die Möglichkeit des Nutzers in kritischen Situationen selbst einzugreifen.
- „Impact“: Die Einführung ubiquitärer Systeme hat Auswirkungen auf die Umgebung und auf das Verhalten der Nutzer. Zudem ist nicht jede Handlung zur Steuerung des Systems auch sozial erwünscht. Bspw. ist zu evaluieren, ob Nutzer im Alltag auf der Straße ein kleines Display über einem Auge tragen wollen.

Ein weiteres Rahmenwerk wurde von Ryu et al. [RHJ06] vorgeschlagen. Während Scholtz und Consolvo [SC04] sich auf die Evaluation von Applikationen beschränken, betrachten Ryu et al. auch physische Aspekte. Um einen Gesamtüberblick zu geben, wird zunächst der Informationsfluss beschrieben, bestehend aus fünf Komponenten: Nutzer, Eingabeartefakt, Services, Systemhardware / -software und Ausgabeartefakten. Der veröffentlichte Beitrag beschreibt auf mehr als 70 Seiten sehr detailliert, wie die einzelnen Evaluationsaspekte in Evaluationsfragen heruntergebrochen werden können und schließlich in konkrete Metriken.

Das Rahmenwerk von Kim et al. [KCJ08] bezieht ebenfalls technische Aspekte mit ein. Die Grundlage für eine Evaluation ist hierbei eine schrittweise Zerlegung der zu testenden Geräte in Komponenten. Diese so genannten Evaluationsbereiche umfassen von der Software- hin zur Hardwareebene: „Logical User Interface“, „Graphical User Interface“, „Physical User Interface“ und „Device Hardware“. Für jede dieser Ebenen werden die gleichen Evaluationsfaktoren untersucht. Diese umfassen „Adaptability“, „Controlability“, „Interconnectivity“, „Mobility“, „Predictability“, „Simplicity“ und “Transparency“.

2.3.2.3 Konkrete Methoden

Für den Bereich klassischer interaktiver Systeme gibt es eine Vielzahl von Methoden, die zu Beginn von Kapitel 2 vorgestellt wurden. Diese werden für die Evaluation ubiquitärer Systeme weiterverwendet und dazu je nach Anwendung adaptiert oder in Kombination eingesetzt. Für ubiquitäre Systeme wurden ergänzend neue spezialisierte Methoden entwickelt, von denen nachfolgend einige exemplarisch vorgestellt werden.

Experience Sampling

Die „Experience Sampling"-Methode [CW03] wird sehr früh im Entwicklungsprozess angewandt, um den Alltag der potentiellen Nutzer eines zu entwickelnden Systems besser kennenzulernen und abzuschätzen, wie sich das System voraussichtlich integriert. Noch vor dem Beginn der Entwicklung lässt sich evaluieren, ob eine ubiquitäre Applikation im gewählten Bereich sinnvoll ist, und während der Anforderungsanalyse lassen sich Anforderungen erheben. Dazu bekommen die Probanden ein mobiles Gerät, bspw. einen PDA, das sie im Alltag ständig bei sich haben. Das Gerät meldet sich regelmäßig und stellt Fragen zu der aktuellen Situation des Probanden. Je nach Funktionsumfang des mobilen Gerätes kann bspw. auch um ein Foto der aktuellen Umgebung gebeten werden. Mögliche Auslöser der Fragen können ein Zufallsmechanismus sein, ein abgelaufenes Zeitintervall seit der letzten Frage oder das Erreichen eines besonderen Ortes. Nicht in jeder Situation hat der Proband die Zeit und das Interesse die Fragen zu beantworten, daher kann bspw. eine Vergütung je beantworteter Frage diesen Anreiz steigern. Durch stichprobenartige Einblicke in den Alltag der Probanden entstehen qualitative und quantitative Daten, die das reale Nutzungsumfeld der zu entwickelnden Applikation abbilden. Während der Definition der Anforderungen kann man vergleichen, ob die Applikation sich in der realen Umgebung sinnvoll einsetzen lässt. Bspw. kann die multimodale Interaktion eingeschränkt werden durch Helligkeit, Dunkelheit, hohe Lautstärke, Unerwünschtheit von Störungen an ruhigen Orten oder häufige Ortswechsel.

Prototypen und Paratypen

Abowd et al. [AHI+05] gliedern prototypische Techniken in (1) „compound prototypes" bezeichnet als „Prototyp" und (2) „situated experience prototypes" bezeichnet als „Paratyp".

(1) Prototypen sind lauffähige Systeme, die eine vollständige Benutzungsschnittstelle besitzen, während die Funktionalität durch ein externes System simuliert werden kann. Dadurch kann man Ressourcenbeschränkungen auf kleinen Geräten zunächst übergehen und eine möglichst authentische Nutzungserfahrung vermitteln, während der Applikationskern schnell an verschiedene Geräte mit deren Oberflächen angebunden werden kann. Allerdings muss einiges implementiert werden, um evaluieren zu können. Werkzeugunterstützung für das rasche Erstellen physischer Prototypen bieten d.tools [HKB+06] und iStuff [BMR+07].

(2) Paratypen hingegen benötigen keine Implementation. Sie umfassen experimentelle Protokolle, die Nutzerinteraktionen mit dem realen System reproduzieren, und optional physische oder Papierprototypen. Den Nutzern wird zunächst das geplante System erklärt. Falls ein erster, instabiler Prototyp existiert, kann dieser gezeigt werden, um die Vorstellung zu erleichtern und die Motivation zu erhöhen. Dann bekommen die Probanden ein Notizbuch oder ein mobiles Gerät, das sie im Alltag stets bei sich haben. Immer wenn sie im Alltag das zu entwickelnde System nutzen würden, füllen sie einen kurzen Fragebogen dazu aus, in welcher Situation sie das System wie benutzen würden. Der Fragebogen sollte allerdings der Situation angemessen sein, also schnell zu beantworten, ohne den natürlichen Tagesablauf zu stark zu beeinflussen, keine zu starke mentale Belastung hervorrufen und in der physischen Umgebung ausfüllbar sein. Paratypen setzen ähnlich wie die „Experience Sampling"-

Methode keine Implementation voraus und können damit auch für die Anforderungsanalyse genutzt werden.

Technology Probes

„Technology Probes“ [HMW+03] sind ein Instrument, um eine neue Technologie in realem Umfeld zu testen. Ziel dabei ist nicht die detaillierte Verbesserung eines konkreten Prototypen, sondern das Testen einer Technologie und das Gewinnen von Ideen, wie man diese Technologie nutzen kann. Die Bandbreite möglicher Ergebnisse ist damit bewusst offen gehalten und kann auch eine völlig andere Anwendungen ergeben. Zur Durchführung wird ein lauffähiges System benötigt, dass technisch möglichst einfach und flexibel ist, so dass Nutzer damit gut klar kommen und es zugleich zum Test weiterer Anwendungen gut adaptierbar ist. Dieses System wird im realen Umfeld der Testnutzer platziert, so dass es im Alltag angewandt wird. Die Nutzung wird aufgezeichnet und später von Entwicklern und Nutzern reflektiert. Fragestellungen sind dabei: Wie verändert sich das Verhalten der Nutzer durch diese Technologie? Ist die Technologie zukunftsträchtig? Welche Anpassungen oder weiteren Anwendungen sind wünschenswert? Insgesamt besteht das Zielsystem aus drei Komponenten: (1) Sozialwissenschaften streben an, die Bedürfnisse von Nutzern im realen Umfeld zu verstehen, (2) Ingenieurwissenschaften streben an, einen Feldversuch der Technologie durchzuführen und (3) Design strebt an, Nutzer und Forscher zu inspirieren.

Wizard of Oz

Die „Wizard of Oz“-Methode [DJA93, RDW+05, CHB+07] erlaubt das frühzeitige Testen einer Applikation, die noch nicht implementiert ist. Ein Proband erhält einen Prototypen des zu erstellenden Systems, der äußerlich voll funktionsfähig wirkt, allerdings keine oder nur eingeschränkte Funktionalität besitzt. Jede Interaktion des Nutzers mit dem System wird an einen Experten weitergeleitet, der für den Probanden nicht sichtbar ist und sich bspw. im Nachbarraum aufhält. Der Experte reagiert daraufhin exakt so, wie das angestrebte System später reagieren soll und steuert entsprechend die Ausgabe an den Probanden und aktualisiert damit den Prototyp. Diese Vereinfachung ist besonders dann sinnvoll, wenn komplexe Funktionalitäten, wie Spracherkennung oder Ortsbestimmung, durchzuführen wären. Ein kritischer Punkt ist die Belastung des Experten bei zu häufiger oder zu umfangreicher Aktualisierung des Prototypen. Daher ist eine Bedienoberfläche nötig, die dem Experten diese Arbeit möglichst effizient gestaltet, wie bspw. in [LHL07] beschrieben. Die Analyse des Nutzerverhaltens gibt eine Vorschau auf das zu erstellende System und versucht mögliche Schwächen in der Gebrauchstauglichkeit frühzeitig zu erkennen. Über praktische Erfahrungen der „Wizard of Oz“-Evaluation von proaktiver Assistenz bei Besprechungen berichten [RNB08].

Multimodal Theater

Das „Multimodal Theater“ [CLS02, SL02] kombiniert die bei Interaktionsdesignern beliebten Papierprototypen mit der „Wizard of Oz“-Methode. Die Technik lässt sich frühzeitig anwenden, da weder ein kompletter Prototyp noch eine reale Bedienoberfläche benötigt werden. Es genügt eine Idee des zu erstellenden Systems, die als Papierprototyp

umgesetzt wird. Dieser wird später durch Experten so animiert, wie das geplante System später funktionieren soll. Dabei werden bspw. Sprach-, Gesten- und Ortserkennung, sowie Sprachausgabe durch den Experten simuliert. Um die Simulation zu verbessern wird zudem empfohlen, den Prototyp flexibel mit Stift, Schere und Kleber anzupassen.

Heuristische Evaluation

Heuristische Evaluationen [NM90] sind eine etablierte Methode für die Evaluation klassischer interaktiver Systeme. Eine Benutzungsschnittstelle wird von Experten mit einem Satz von Anforderungen an ein Design mit guter Gebrauchstauglichkeit verglichen, um Schwächen aufzudecken und zu beheben. Die bestehenden Heuristiken wurden vielfach erfolgreich eingesetzt zur Evaluation klassischer Benutzungsschnittstellen, müssen aber für allgegenwärtige Anzeigen („Ambient Displays“) überdacht werden. Wesentlicher Unterschied bei allgegenwärtigen Anzeigen ist, dass sie nicht aktiv zur Steuerung genutzt werden, sondern nur passiv zur peripheren Information [Fer07]. Sie integrieren sich möglichst nahtlos in den Alltag [WB96] und sollen ästhetisch ansprechend sein. Mankoff et al. [MDH+03] haben vorhandene Heuristiken analysiert, Usability-Experten befragt und einen Satz neuer angepasster Heuristiken für allgegenwärtige Anzeigen erstellt. In einer Evaluation konnten nach den Anpassungen signifikant mehr Usability-Probleme gefunden werden. Wie auch bei der klassischen heuristischen Evaluation findet ein Experte für sich genommen nur einen Teil der Probleme, so dass die Ergebnisse mehrerer Evaluatoren aggregiert werden sollten.

Ethnographische Studien

Um die Nutzung von Software im sozialen Umfeld zu betrachten, hat sich Ethnographie in den letzten 20 Jahren zu einem wichtigen Element entwickelt [CBG+06]. Um den Besonderheiten von ubiquitären Applikationen Rechnung zu tragen, haben Crabtree et al. [CBG+06] bestehende Methoden angepasst. Schwerpunkt ist die Zusammenführung der traditionell verwendeten externen Daten (bspw. Video- / Audiodokumente) mit den vom ubiquitären System bereitgestellten internen Daten (bspw. Positionsdaten). Zudem lassen sich über die traditionell aufgezeichneten Systemzustände hinaus bei ubiquitären Systemen auch Sensorinformationen über soziale Interaktion und Kooperation aufzeichnen. Die Analyse erlaubt den Entwicklern soziale Aspekte der Interaktion in realer Umgebung in das Design einzubeziehen.

Speed Dating

„Speed Dating“ [DLD+07] ist eine Designmethode zum schnellen Erforschen verschiedener Applikationskonzepte, sowie deren Interaktionen und Kontextvariablen, ohne eine Implementation vorauszusetzen. Verschiedene Konzepte werden strukturiert verglichen, um kontextbezogene Risikofaktoren zu identifizieren und Lösungsmöglichkeiten zu finden. Die Methode läuft in zwei Phasen ab: Validieren der Bedürfnisse und Nachspielen durch Nutzer. (1) Zunächst werden Nutzern aus der Zielgruppe verschiedene Papier-Storyboards vorgeführt, um Designlösungen der Forscher und Bedürfnisse von Nutzern abzugleichen. Dadurch soll der Lösungsraum für die zukünftige Applikation eingeschränkt werden. (2) Anschließend wird eine Matrix von Designproblemen erstellt. Zu jeder Permutation wird in Team-

Arbeit eine kurze dramatisierte Geschichte erarbeitet, die die Kombination der Designprobleme deutlich macht. Bspw. kann ein Ablauf mit geringer, mittlerer und hoher proaktiver Assistenz verglichen werden. Die Geschichten werden von Teilnehmern der Studie nachgespielt, wobei jeder seine typische Rolle aus dem Alltag darin wahrnimmt.

In der Literatur finden sich weitere Methoden, wie „Cultural Probes“ [GDP99], „Video Sketches“ [Zim05], „Contextual Inquiry“ [Hol05] und „Co-Design“ [HHF+06].

Die vorgestellten Methoden haben unterschiedliche Stärken und Schwächen. Die Wahl der Methode hängt von den Zielen und Rahmenbedingungen der Evaluation ab [DFA+04] von der aktuellen Entwicklungsphase, den vorhandenen Ressourcen, den zu ermittelnden Kennzahlen, dem gewünschten Grad der Objektivität/Subjektivität der Evaluation und der Art der Evaluation (Labor- oder Feldtest). Nach Carter et al. [CMK+07] ist die Wahl der Methode ein Abwägen im Bezug auf Realitätsnähe, Grad des Eingriffs in natürliche Handlungsweisen, Umgang mit Mehrdeutigkeiten, Datenknappheit, Kosten und Zeit.

So können Papierprototypen und „Wizard of Oz“-Experimente frühzeitig eingesetzt werden und helfen, Mehrdeutigkeiten aufzulösen. Während Papierprototypen nur einen geringen Ressourceneinsatz erfordern, erreicht „Wizard of Oz“ eine größere Realitätsnähe. „Experience Sampling“ erlaubt ebenfalls große Realitätsnähe, setzt aber früher an und liefert Beobachtungsdaten aus dem Alltag zukünftiger Nutzer.

2.3.2.4 Konkrete Evaluationen von Systemen

Der nachfolgende Überblick listet eine Reihe von „Smart Environment“-Systemen auf und beschreibt die durchgeführte Evaluation der Usability. Dabei wird jeweils zuerst das Projekt zum Bau des Systems kurz dargestellt, gefolgt von einem Überblick der angewandten Evaluationsmethoden.

Tivoli

Tivoli [MPH+97] ist ein System zur Aufzeichnung und Unterstützung von Besprechungen. Ergänzend zu herkömmlichen schriftlichen Protokollen werden Bild und Ton aufgezeichnet, um später auch Tonfall und nichtverbale Äußerungen beurteilen zu können. Dadurch wird neben dem Ergebnis auch der Prozess der Meinungsfindung erfasst.

Zur Evaluation wurden über 2 Jahre hinweg 60 authentische Besprechungen im Xerox Parc begleitet. Diese wurden dokumentiert durch Videoaufzeichnungen, gesammelten Dokumenten, Interviews mit Teilnehmern und Logs der Interaktion mit dem System. Die Analyse zeigte, wie sich die Nutzung mit wachsender Erfahrung änderte: von einer anfänglich punktuellen Nutzung der Aufzeichnungsfunktionen, hin zu einer dauerhaften Nutzung mit punktueller Deaktivierung bei sehr brisanten Diskussionen. Dabei wurde der Zwiespalt zwischen Vorteilen der Nutzung und Einschränkungen der Privatsphäre deutlich.

Classroom 2000

Der Classroom 2000 [Abo99] ist ein vollinstrumentierter Klassenraum, der Vorlesungen aufzeichnet, um sie später wieder als Dokumentation verfügbar zu machen. Die anwesenden

Studenten sollen damit in die Lage versetzt werden, ihre Strategien zum Notizenmachen und Studieren frei zu wählen, ohne von der beim Vortrag genutzten Technologie eingeschränkt zu werden. Das Projekt wurde in zwei Phasen evaluiert [SAK+02]:

(1) Zunächst wurde für einen frühen Prototyp eine formative Evaluation ohne Kontrollgruppe durchgeführt, um die tatsächliche Benutzung durch die Studenten besser zu verstehen und die weitere Entwicklung daran auszurichten. Der Prototyp bestand aus zwei Komponenten: ein Whiteboard zum Aufzeichnen der Aktivitäten des Lehrenden und ein Laptop zum Aufzeichnen der Notizen der Studenten. Während einer Nutzerstudie wurden Studenten bei der Verwendung des Systems beobachtet und eine erste Einschätzung war, dass die Notizfunktion für Studenten kaum Vorteile brachte. Eine ergänzende Umfrage zeigte, dass die Nutzer die Bedienoberfläche für Notizen als zu kompliziert empfanden. Daraufhin wurde diese Funktionalität entfernt und nur das Whiteboard beibehalten.

(2) Nach der anschließenden Weiterentwicklung hin zum fertigen System wurde eine summative Evaluation mit Kontrollgruppe durchgeführt. Innerhalb von 18 Monaten wurden 60 Studenten an verschiedenen Universitäten zu Tests eingeladen. Dabei entstanden umfangreiche quantitative statistische Daten und qualitative Daten aus Interviews und Umfragen. Die Auswertung der Daten konnte ein umfangreiches Bild zeichnen von der Änderung des Lernprozesses innerhalb der instrumentierten Umgebung und den Auswirkungen außerhalb des Klassenraumes.

Die Evaluationen wurden über mehrere Jahre hinweg durchgeführt und gehören damit zu den umfangreichsten im Bereich ubiquitärer Applikationen. Verbesserungen wurden jeweils iterativ vorgenommen und erneut evaluiert. Abowd et al. haben auf diese Weise das Konzept des „living laboratory“ entwickelt, bei dem reale Nutzer in einer Laborumgebung reale Aufgaben durchführen, begleitet durch iterative Weiterentwicklung..

Doorman

Das System Doorman [MST+01] öffnet die Tür für Besucher der Universität von Tampere und leitet sie durch das Gebäude. Es ist behilflich dabei, den Weg zu bestimmte Räume oder Personen zu weisen. Die Interaktion erfolgt über natürlich sprachliche Eingaben und multimodale Sprachausgaben, also Sprache in Kombination mit Zeigegesten.

Die Evaluation wurde begleitend zur Entwicklung als „Wizard of Oz“-Experiment durchgeführt. Das Spracherkennungsmodul wurde durch einen menschlichen Interpreter ersetzt, der entsprechend der Nutzereingaben das System manuell gesteuert hat, um das fertige System zu simulieren. Basierend auf ihren Erfahrungen empfehlen Mäkelä et al. [MST+01] das Verfahren für die iterative Entwicklung von ubiquitären Systemen.

Campus Aware

Das ursprünglich unter dem Namen „E-Graffitti“ begonnene Projekt „Campus Aware“ [BGK+02] unterstützt das kontextsensitive Erstellen von Informationen. Nutzer können Nachrichten austauschen und mit Orten verknüpfen. Neben Fakten zum Ort können auch Meinungen und Fragen annotiert werden, die zu späteren Antworten anderer anregen. Wenn sich Nutzer an diese Orte bewegen, erscheinen die entsprechenden, hinterlegten Nachrichten, um die Navigation auf dem Campusgelände zu erleichtern.

Die Evaluation erfolgte durch eine Kombination aus Umfragen, Interviews und qualitative Auswertungen der Nutzungsdaten und annotierten Inhalte. Als Nutzer dienten sowohl neue Studenten, die das Gelände nicht kannten, als auch langjährige Studenten mit entsprechender Ortskenntnis. Die Entwicklung und Evaluation erfolgte in zwei Iterationen, mit dem Fazit, dass auf Basis der Nutzerstudien weitere Iterationen sinnvoll erscheinen.

Labscape

Labscape [AB02, CAF02] ist ein Smart Environment, das Zellbiologen bei ihrer täglichen Arbeit im Labor unterstützt. Während eines Experiments sammelt und organisiert das System Daten als Dokumentation und bringt diese in eine formale Repräsentation, die auch für nicht beteiligte Biologen lesbar sein soll.

Das Ziel der Evaluation bestand darin, zu untersuchen wie gut sich die gebotene Assistenz in die Laborabläufe integriert und die Prozesse verbessert. Innerhalb einer Feldstudie führten Biologen reale Experimente mit Labscape durch. Währenddessen wurden quantitative Daten, wie bspw. die Häufigkeit der Zugriffe auf die automatisch erstellte Dokumentation, erfasst. Da die Experimente allerdings deutlich voneinander abwichen, waren keine statistischen Auswertungen möglich. Die Daten wurden daher eher im deskriptiven als im statistischen Sinne verwendet. Die Probanden bekamen bewusst keine künstlichen, besser vergleichbaren Aufgaben, da die natürlichen Abläufe nicht verändert werden sollten und die Laboranten keine Zeit hatten für zusätzliche unproduktive Experimente. Für die Evaluation wurden Videobeobachtungen, Interaktionslogs und Interviews durchgeführt.

Exploratorium

Das Forschungslabor von Hewlett-Packard betreibt ubiquitäre Applikationen im „Exploratorium“ [FKO+02], einem interaktiven Wissenschaftsmuseum in San Francisco. Verschiedene Applikationen wurden erprobt. Bspw. gibt „Informer“ Informationen zu Exponaten, an denen Besucher verweilen. Dazu bekommen Besucher PDAs, die per Infrarot die passierten Exponate erkennen. Ergänzend dazu hilft die Applikation „Rememberer“, interessante Exponate zu merken, Informationen dazu zu speichern und aufgenommene Photos dem jeweiligen Aufnahmeort zuzuordnen. Dafür werden RFID-Elemente an den Exponaten genutzt. „Suggester“ macht Vorschläge dazu, was man am aktuellen Exponat ausprobieren kann.

Die Evaluation erfolgte über Beobachtungen, Interviews, Logs der Internetzugriffe zum Besuch der Internetseiten mit Exponatinformationen und Email-Umfragen nach dem Museumsbesuch. Ergänzend dazu wurden Beobachtungen und Umfragen von Kontrollgruppen genutzt. Im Ergebnis stellte sich bei „Rememberer“ als positiv heraus, dass Museumsbesucher interessante Exponate merken und interaktive Inhalte später zu Hause in Ruhe anschauen konnten. Allerdings war allen Applikationen gemein, dass viele Nutzer begeistert waren von den Funktionen der PDAs und diese teils mehr Aufmerksamkeit bekamen, als die Exponate des Museums selbst.

Homelab

Das Homelab [WRA05] wurde von Philips eingerichtet. Alle Räume sind ausgestattet wie in einer normalen Wohnung, um möglichst realitätsnahe Studien durchführen zu können. Probanden haben die Möglichkeit längere Zeit dort zu leben, um das Verhalten 24 Stunden am Tag zu beobachten. Dafür sind Mikrophone und 34 Videokameras für Probanden unsichtbar eingebaut. Angrenzend an die Wohnung befindet sich ein Beobachtungsraum. Beobachtungen werden annotiert und statistisch (z.B. LSA, Zeitreihenanalysen) ausgewertet, um bspw. wiederkehrende Verhaltenmuster zu erkennen. Ergänzend dazu wurden Interviews und Fragebögen verwendet. Für Bestandteile der Umgebung, die noch nicht implementiert waren, wurden „Wizard of Oz"-Experimente durchgeführt.

Placelab

Das Placelab [ILB+05] ist eine reale Wohnung, die Wohn-, Schlaf-, Büro-, Küchen- und Badezimmer umfasst. Alle Räume sind mit Sensoren ausgestattet, bspw. für Temperatur, Luftfeuchtigkeit, Licht, Wasser-, Gas-, Stromverbrauch, Tür- und Fensteröffnung.

Im Sinne eines „living laboratory" erfolgt die Evaluation langfristig und rund um die Uhr. In einer Evaluation [ILB+05] lebten drei Probanden dort für jeweils 10 Tage. Währenddessen wurden Interaktionen durch drei Bewegungssensoren am Körper aufgezeichnet, sowie Interviews und Fragebögen genutzt.

Smart Home Controller

Ein „Smart Home Controller" [YLL+08] soll die Steuerung von Geräten (bspw. Licht, Fernseher, Klimaanlage, Rollläden) in der eigenen Wohnung erleichtern. Eine grafische Benutzungsoberfläche zeigt den Zustand der Umgebung an und erlaubt die Steuerung der Geräte. Für jede Nutzergruppe soll eine maßgeschneiderte Oberfläche bereitstehen.

Vor Beginn der Entwicklung wurden Personas entwickelt. Dies geschah auf Basis von Interviews mit späteren Nutzern, Fokusgruppen und statistischen Daten über chinesische Haushalte. Zwei davon, der ältere Mensch und das Kindermädchen, wurden näher untersucht. Ausgehend von erarbeiteten Anwendungsszenarien wurde für beide Personas je ein Prototyp entwickelt. In einer Nutzerstudie hatten je vier bzw. fünf Testnutzer aus der Zielgruppe in einer realen Umgebung Aufgaben zu erfüllen. Entsprechend der auftretenden Probleme wurden die Prototypen verbessert.

DICE

Das „Distributed Intelligent Conferencing Environment" (DICE) [GQM+09] verfolgt das Ziel, Konferenzraumtechnologie für jeden einfach verfügbar zu machen, ohne dass die Anwesenheit eines Technikers erforderlich ist. DICE umfasst einen Konferenzraum für 50 Teilnehmer mit Podium und drei großen Bildschirmen. Alternativ zur Assistenz kann der Raum auch über eine grafische Benutzungsoberfläche konfiguriert werden.

Die Entwicklung erfolgte in zwei Phasen. In der ersten Phase wurden nur die Kernfunktionen implementiert und anschließend evaluiert. Quantitative Aspekte umfassten dabei die Analyse von aufgezeichneten Nutzungsdaten und qualitative Aspekte Interviews der Nutzer. In der zweiten Ausbaustufe wurde der Raum auf Basis der Ergebnisse der Nutzerstudien

iterativ erweitert und verändert. So wurde bspw. die grafische Benutzungsoberfläche zur Konfiguration verbessert.

In der Literatur finden sich weitere ubiquitäre Applikationen mit einer Dokumentation der Usability-Evaluation, die hier nur genannt werden sollen: Guide [CDM+00], Sotto Voce [WSH01], Future Home [IR02], Rasa [MCW+02] und Tampere Univerity eHome [KV04].

Die vorgestellten „Smart Environment"-Projekte geben einen Überblick über Evaluationen der letzten 10 Jahre in Hochschulen und der Industrie. Insbesondere in frühen Projekten mangelt es an Erfahrung, wie Nutzer sich in Smart Environments verhalten, um daraus Designvorschläge abzuleiten. Das Konzept des „living laboratory" (eingeführt in Classroom 2000) setzt daher auf langfristige Beobachtungen menschlichen Verhaltens in realen Situationen (auch in Homelab, Placelab). In mehreren Projekten (Classroom 2000, Campus Aware, DICE, Smart Home Controller) erfolgt die Entwicklung in Phasen, bei denen ein Prototyp iterativ evaluiert und weiterentwickelt wird. Allerdings wird bei diesen Projekten nur über zwei Iterationsschritte berichtet. Im Bereich klassischer Systeme werden häufig viele Iterationen mit geringem Aufwand verwendet [Nie93]. Sogenannte „discount"-Evaluationsmethoden[2] [Nie90, Nie93], wie Heuristiken oder kognitives Durchwandern, wurden nicht eingesetzt. Am häufigsten wurden Nutzertests, Interviews und Fragebögen verwendet. Eines der aktuelleren Projekte (Smart Home Controller) orientiert sich zur Anforderungsanalyse an Methoden der Benutzer-orientierten Gestaltung und definiert Personas und Szenarien. Bei der Durchführung von Nutzertests unterscheiden sich die Evaluationen nach solchen mit vorgegebenen Aufgaben (bspw. Smart Home Controller) und solchen mit frei wählbaren Aufgaben (bspw. Labscape). Vorgegebene sind gut kontrollierbar und statistisch auswertbar, allerdings können frei wählbare ein authentischeres Bild liefern.

2.4 Zusammenfassung

Usability ist eine wichtige Produkteigenschaft, die eine effektive, effiziente und zufriedenstellende Arbeit sicherstellen soll [DIN98]. Für traditionelle Arbeitsplatzanwendungen stehen zahlreiche Methoden aus den Bereichen Testen, Inspektion, Befragung, sowie analytischer Modellierung und Simulation zur Verfügung [IH01, Nie93].

Smart Environments sind physische Orte, die auf das Verhalten von Nutzern reagieren und Assistenz für die Erreichung ihrer Ziele anbieten [AE06]. Interaktionen sind gekennzeichnet durch eingebettete, unsichtbare Benutzungsschnittstellen, multimodale Interaktionen, eine starke Kontextabhängigkeit und die Erwartung der Nutzer, ohne Bedienungsanleitung direkt arbeiten zu können. Daraus resultieren besondere Herausforderungen bei der Evaluation [PRD07, NSK+08]. Bspw. ist der komplette Aufbau eines Smart Environments für einen Test ressourcenintensiv, kooperative Interaktionen können nicht

[2] „Discount Usability-Methoden versuchen mit geringem Ressourceneinsatz eine große Menge an Usability-Problemen zu identifizieren. [Nie90]

losgelöst evaluiert werden, umfangreiche Kontextvariablen sind zu berücksichtigen und bei der Auswertung von Daten aus Feldversuchen ist der Umgang mit privaten Daten kritisch. Daher müssen bestehende Evaluationsmethoden aus dem Bereich traditioneller Arbeitplatzanwendungen in Bezug auf ihre Übertragbarkeit auf Smart Environments überprüft werden. Evaluationsmethoden für Smart Environments lassen sich auf Grund ihrer Beziehung zu traditionellen Methoden in drei Kategorien einteilen:

1. traditionelle Methode direkt anwendbar (bspw. Fragebögen, Interviews)
2. traditionelle Methode angepasst (bspw. Heuristiken [MDH+03], Adaption von Papierprototypen zum „Multimodal Theater“ [CLS02])
3. neue Methode entwickelt (bspw. „Experience Sampling“ [CW03], Paratypen [AHI+05])

Eine Untersuchung von Projekten zum Bau von Smart Environments ergab, dass diese Methoden in verschiedenen Kombinationen genutzt werden, wobei die Intergration von Entwicklungs- und Usability-Prozessen noch am Anfang steht. Die Untersuchung der Usability früher Prototypen ist darüber hinaus noch wenig ausgeprägt [CM04].

Kapitel 3

Vorgehensmodell für Usability-Evaluation in Smart Environments

In diesem Kapitel werden zunächst die Anforderungen an das zu entwickelnde Vorgehensmodell formuliert. Darauf aufbauend wird das Vorgehensmodell vorgestellt und die Softwarearchitektur für die Werkzeugunterstützung erläutert. Anschließend wird ein Beispiel eingeführt, das die Methoden im weiteren Verlauf der Arbeit illustrieren soll.

3.1 Anforderungen

3.1.1 Unterstützung früher Phasen des Entwicklungsprozesses

Die Entwicklung von Software soll einem wohl definierten Prozess folgen [Bal98]. Ein Prozessmodell liefert die entsprechende Beschreibung, welche Entwicklungsphasen in welcher Reihenfolge durchlaufen werden. In jeder Phase werden bestimmte Aktivitäten durchgeführt, die festgelegte Artefakte hervorbringen. Währenddessen sind die im Prozessmodel spezifizierten Rahmenbedingungen einzuhalten, wie bspw. Verantwortlichkeiten, notwendige Qualifikationen der beteiligten Mitarbeiter, anzuwendende Methoden und Standards.

Je früher während der Entwicklung ein Problem identifiziert wird, desto einfacher kann es behoben werden [Bal98 S.22]. Die „1:10:100 - Regel“ [Rav03 (S.202)] besagt, dass sich die Kosten zur Beseitigung eines Fehlers von einer Entwicklungsphase zur nächsten um den Faktor 10 steigern. Wenn bspw. ein Problem in der Anforderungsphase noch in 1 Stunde behoben wird, kann sich dieser Aufwand in der Designphase schon auf 10 Stunden erhöhen.

Carter et al. [CM04] haben den Stand der Forschung für die Evaluation ubiquitärer Systeme untersucht. Bestehende Ansätze wurden dabei nach der Phase der Entwicklung, in der sie durchgeführt werden, klassifiziert. Für die Evaluation noch vor Beginn des Designs wurden verschiedene Methoden identifiziert, die das Verhalten der Nutzer mit dem alten System untersuchen. Auch zur Evaluation nach Einrichtung des fertigen Systems stehen vielfältige aufwendige summative Evaluationsmethoden bereit, die entweder im Labor oder

in realer Umgebung den Umgang mit dem neuen System untersuchen. Für die Evaluation in frühen Phasen der Entwicklung, bspw. zur Begleitung von Design und Implementation, und für die iterative Evaluation sind jedoch kaum Methoden verfügbar. Dies steht im Gegensatz zur Evaluation mit klassischen Methoden, wo kostengünstige Methoden zur Evaluation früher Artefakte (bspw. Papierprototypen oder heuristische Evaluation) sehr beliebt sind. In einer Studie [VMS+02] wurden Teilnehmer der Konferenz CHI („Computer-Human Interaction“) und der UPA („Usability Professionals' Association“) befragt. Mehr als die Hälfte der meistverwendeten Methoden waren dabei informelle, kostengünstige Methoden. Als Gründe dafür wurden Zeit- und Kostenvorteile genannt.

Alles in allem fehlen derzeit Methoden für die Evaluation von Smart Environments in den frühen Phasen der Entwicklung. Ein zu entwickelnder Ansatz soll Methoden zur Verfügung stellen, die diese Lücke schließen.

3.1.2 Konkrete Ausgestaltung des Paradigmas „Benutzer-orientierter Gestaltung“

Um die Akzeptanz und Produktivität späterer Nutzer sicherzustellen werden bei der Benutzer-orientierten Gestaltung interaktiver Systeme nach DIN EN ISO 13407-11 [DIN99] (UCD - „user-centred design“) frühzeitig Nutzerinteressen berücksichtigt. Die Bedeutung des Benutzer-orientierten Designs wird auch von Norman [Nor02] und Shneiderman [She02] hervorgehoben. Zentrale Eigenschaften dieses Prinzips sind:

- „die aktive Beteiligung der Benutzer und ein klares Verständnis von Benutzer- und Aufgabenanforderungen;
- eine geeignete Funktionsaufteilung zwischen Benutzern und Technik;
- die Iteration von Gestaltungslösungen;
- multidisziplinäre Gestaltung.“ [DIN99]

Gulliksen et al. [GGB+03] kritisieren allerdings, dass die bisherigen Definitionen, bspw. von Norman [Nor86] und Karat [Kar97], zwar wichtige Eigenschaften von UCD-Prozessen nennen, aber zu generell und unspezifisch sind, um eine direkte Umsetzung zu ermöglichen. Karat charakterisiert UCD durch „… [UCD] captures a commitment the usability community supports - that you must involve users in system design - while leaving fairly open how this is accomplished.“ Gulliksen et al. [GGB+03] reagieren darauf, indem sie aus der Literatur und eigenen praktischen Erfahrungen Schlüsselprinzipien herausarbeiten, um den UCD-Prozess für die Entwicklung interaktiver Systeme zu konkretisieren.

Im Rahmen dieser Arbeit soll der UCD-Prozess speziell für die modellbasierte Entwicklung von Smart Environments mit Evaluationsmethoden konkretisiert werden und Werkzeuge bereitgestellt werden, die die praktische Umsetzung eines UCD-Prozesses ermöglichen.

3.1.3 Aufgabenbasierter Prozess

Die Benutzer-orientierte Gestaltung basiert auf einem „klare[n] Verständnis von Benutzer- und Aufgabenanforderungen“ [ISO99]. Während der Anforderungsanalyse sind die Aufgaben der Nutzer zu erarbeiten, um diese möglichst gut zu unterstützen. Auch später während der weiteren Entwicklung ist dieses Verständnis der Nutzerziele und -aufgaben Voraussetzung dafür, eine gute Gebrauchstauglichkeit sicherzustellen:

> *„task analysis does not stop with design. At every stage of the design and development process, task analysis is critical. It is the major input to use cases and design specifications, and it helps us understand how the emerging product affects users. It is the key to evaluating designs as scenarios for heuristic evaluations and for usability testing Task analysis must be the organizing principle for documentation and training.“ [CRW07 (S.929)]*

Cook et al. [CD04 (S.xiv)] kritisieren im Jahre 2004 noch die vereinfachte Betrachtung von Kontextinformationen, die bestimmte Facetten, wie den Ort, in den Mittelpunkt stellen, während Aufgaben übergangen werden. Beim CHI’2006-Workshop „The Many Faces of Consistency in Cross-Platform Design“ wurde diskutiert, was eine konsistente Benutzung über Plattformgrenzen hinweg bedeutet und welche Faktoren einwirken. Als Fazit wurden 5 Faktoren identifiziert, wobei der Aspekt der Aufgabe als vermutlich wichtigster identifiziert wurde [NRG06, DF06]. Dix et al. [KGC+08] beobachten ein wachsendes Interesse an einer Untersuchung der Rolle von Aufgaben speziell in ubiquitären Systemen:

> *„there is a growing interest within the research community regarding tasks in ubiquitous computing.” [KGC+08 (S.192)]*

So wird eine aufgabenbasierte Modellierung für ubiquitäre Systeme durchgeführt von Luyten et al. [LBV+06], Trapp et al. [TR06], Coutaz et al. [CBA+07], Feuerstack et al. [FBA07] [BLF+08] und Wurdel et al. [WPF08], während Evaluationsmethoden fehlen, die die frühzeitige Evaluation dieser Modelle erlauben.

Entsprechend soll ein aufgabenbasierter Prozess für die Entwicklung von Smart Environments zu Grunde gelegt werden, der das Zusatzwissen aus Aufgabenmodellen für die Evaluation nutzbar macht.

3.1.4 Integration von Entwicklung und Evaluation

Wie Dumas und Redish [HR99] bereits in den 1990er Jahren feststellten, vollzieht sich ein Wandel bei der Evaluation weg von großen, summativen Methoden am Ende der Implementation hin zu kleineren, formativen Methoden begleitend zum Entwicklungsprozess. Entsprechend spielt die Integration von Softwareentwicklung und Evaluation eine zunehmend größere Rolle [Aik06, NLB09]. Die in dieser Arbeit zu entwickelnde Methode soll

eine Integration gewährleisten: einerseits auf Ebene des Vorgehensmodells durch die Integration der Evaluation in den Entwicklungsprozess und andererseits während der operativen Prozesse durch die Integration von Evaluationswerkzeugen. Durch die Integration von Werkzeugen und gemeinsame Nutzung von Artefakten soll der Übergang zwischen Entwicklung und Evaluation in beiden Richtungen erleichtert werden:

- Übergang Entwicklung – Evaluation: Nachdem ein Artefakt (bspw. Aufgabenmodell oder Quellcode) entwickelt wurde, soll die Evaluation dessen mit möglichst geringem Aufwand vorbereitet und durchgeführt werden können.
- Übergang Evaluation – Entwicklung: Die zu entwickelnde Werkzeugunterstützung soll für ausgewählte Artefakte beispielhaft zeigen, wie Erkenntnisse aus der Evaluation in den Entwicklungsprozess zurückgeführt werden können. Ziel ist eine Verkürzung der Zyklen aus Entwicklung und Evaluation.

Durch den nahtlosen Übergang zwischen Entwicklung und Evaluation soll zudem die von Forschern und Praktikern bspw. in der Studie [VMS+02] gewünschte Zeit- und Kosteneffizienz verbessert werden.

3.1.5 Berücksichtigung des Datenschutzes

Bei Usability-Workshops in den letzten Jahren [NSK+08] wurde vermehrt das Problem des Umgangs mit vertraulichen Nutzerdaten genannt. Werden bei Feldstudien reale Nutzer in realen Situationen beobachtet, bspw. bei internen Besprechungen, entstehen Aufzeichnungen, die sorgsam verwendet werden müssen. Hierbei entstehen Konflikte zwischen Datenschutz und Spielraum für die Evaluation. So empfehlen bspw. Dumas und Redish nach Abschluss eines Nutzertests Schlüsselszenen als kurzen Videozusammenschnitt („highlight tape“) [DR99 (S.355ff.)] zu präsentieren, um Designern und Entwicklern einen Eindruck praktischer Probleme zu vermitteln. Bei Aufzeichnungen interner Besprechungen ist dies allerdings in der Regel nicht möglich. Darüber hinaus kann das Veröffentlichen von Videoaufzeichnungen von Personen auch rechtlich problematisch sein. Methoden der Anonymisierung sollen Lösungen für diese Herausforderung anbieten.

3.2 Vorgehensmodell

Beim Prozess der Softwareentwicklung werden in mehreren Entwicklungsphasen Artefakte geschaffen, beurteilt und weiterentwickelt, bis das Endprodukt in der gewünschten Qualität entsteht. Zur Unterstützung dieses Prozesses haben sich verschiedene Vorgehensmodelle in Forschung und Industrie herausgebildet. Sie beschreiben verschiedene Abstraktionsgrade und besitzen verschiedene Stärken und Schwächen [Bal98], so dass die Auswahl von der konkreten Anwendung abhängig ist.

Nachfolgend wird ein Vorgehensmodell vorgestellt, dass die in Kapitel 3.1 formulierten Anforderungen berücksichtigt und speziell auf die Benutzer-orientierte Gestaltung von Smart Environments zugeschnitten ist.

In der Softwareentwicklung werden (in Anlehnung an [Bal00]) folgende Phasen unterschieden: Planungs-, Anforderungsanalyse-, Design-, Implementations-, Einführungs- und Wartungsphase. Die vorliegende Arbeit konzentriert sich auf die Phasen von der Analyse der Anforderungen bis zur Fertigstellung eines physischen Smart Environments und diskutiert das Vorgehen in diesen Phasen wie folgt:

- Kapitel 4: In der Anforderungsanalyse werden Aufgaben- und Benutzeranalyse durchgeführt, um daraus Aufgabenmodelle zu erstellen, die animiert werden.
- Kapitel 5: In der Designphase werden die Aufgabenmodelle zu kooperativen Aufgabenmodellen verfeinert und mit Kontextbedingungen versehen. Diese Modelle werden im virtuellen Smart Environment animiert.
- Kapitel 6: In der Implementationsphase erfolgt der Übergang zu statistischen Modellen und Nutzerstudien mit der komplett eingerichteten Umgebung werden möglich.

Jede der Entwicklungsphasen läuft iterativ ab, wobei jeder Iterationsschritt aus den drei Teilschritten Planung, Entwicklung und Evaluation besteht, so wie in Abbildung 3.1 dargestellt. Tabelle 3.1 gibt ergänzend dazu einen Überblick der wichtigsten verwendeten Methoden.

(1) Während der Planung wird für den aktuellen Iterationsschritt der Umfang der zu entwickelnden Artefakte festgelegt und die Art der durchzuführenden Evaluation. Auf Basis der Evaluationsergebnisse wird der Fortschritt der Entwicklung beurteilt und bei Erreichen der gewünschten Qualität und der definierten Anforderungen das Produkt ausgeliefert.

(2) Während der Entwicklung werden die geplanten Artefakte erstellt. Je nach Stand der Entwicklung können dies textuelle Szenarien sein, erste Entwürfe von Aufgabenmodellen, verfeinerte kooperative Aufgabenmodelle [WDS08], statistische Modelle (HMM: „Hidden-Markov-Modelle“) [Rab89] oder daraus generierte Quelltexte [BPK+09]. Entsprechende Notationen und Werkzeuge werden erläutert, so weit sie für die anschließende Evaluation notwendig sind.

(3) Die Evaluation [Uhl99] umfasst Methoden zum Testen und Bewerten erstellter Artefakte. Je nach Entwicklungsstand sind dies Expertenevaluationen zum interaktiven Durchlaufen der Entwürfe von Aufgabenmodellen, „Wizard of Oz“-Experimente zur Vorschau auf die spätere Assistenz in realer Umgebung, durch Experten generierte künstliche Sensordaten zum Test einzelner Komponenten oder Nutzerstudien im funktionsfähigen physischen Smart Environment. Neben den im Rahmen dieser Arbeit entwickelten Evaluationsmethoden kommen auch klassische Verfahren, wie Fragebögen und Interviews, zum Einsatz.

Die Evaluation umfasst das Durchführen von Experimenten, das Dokumentieren der Ergebnisse und deren Analyse. Ein wichtiger Aspekt am Ende der Analyse ist die Identifikation von Verbesserungspotentialen. Daher schließt sich den Unterkapiteln zur (4.1 / 5.1 / 6.1)

Entwicklung und (4.2 / 5.2 / 6.2) Evaluation jeweils ein weiteres an, das in (4.3 / 5.3 / 6.3) die Verbesserung der Usability thematisiert.

Den hohen Stellenwert der Rückkopplung von der Evaluation zur Entwicklung würdigt Lindgaard durch: „test results must deliver something useful back into the system development process“ [Lin94]. Die in dieser Arbeit entwickelten Werkzeuge gewährleisten eine enge Integration von Entwicklung und Evaluation und erlaubt teilweise das direkte Verändern von Artefakten aus dem Evaluationswerkzeug heraus, um eine schnelle Verbesserung, erneute Evaluation und Erfolgskontrolle zu ermöglichen.

Das Vorgehensmodell baut auf der Entwicklungsmethode der kooperativen Aufgabenmodellierung [WSF08] und der anschließenden Überführung in statistische Modelle [BPK+09] auf, um Entwicklungs- und Usabilitymethoden zu integrieren.

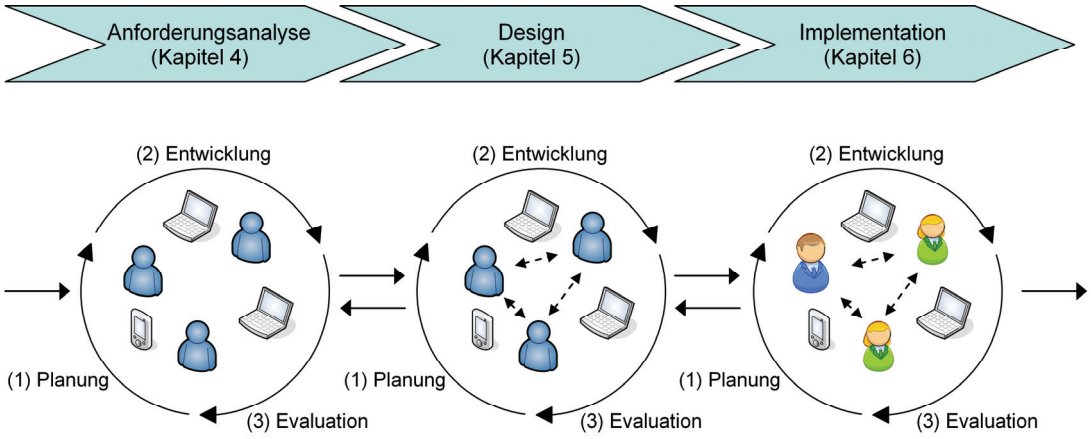

Abbildung 3.1: Vorgehensmodell

	Anforderungsanalyse	Design	Implementation
(1)	- Planung und Auswahl der Entwicklungs- und Evaluationsmethoden		
(2)	Kapitel 4.1: - Benutzeranalyse - Aufgabenanalyse - Entwurf von Szenarien	Kapitel 5.1: - Verfeinerung zu kooperativem Aufgabenmodell	Kapitel 6.1: - Transformation zu HMM und Quellcode - Einrichtung des phys. SE
(3)	Kapitel 4.2: - Durchlaufen der Aufgabenmodelle - „Wizard of Oz“ - Experimente	Kapitel 5.2: - Durchlaufen der kooperativen Aufgabenmodelle im virtuellen SE	Kapitel 6.2: - Generieren künstlicher Sensordaten - Visualisierung der Sensordaten - Datenaufbereitung zur Analyse

Tabelle 3.1: Methoden im Vorgehensmodell

3.3 Architektur des entwickelten Systems

3.3.1 Einzubeziehende Informationen

Bevor die Architektur des entwickelten Evaluationssystems vorgestellt wird, sollen kurz die Informationen genannt werden, die für die Modellierung und Evaluation von Interaktionen in Smart Environments relevant sind. Dies sind:

- Nutzer: Die Interaktionen der Nutzer lassen sich nur dann richtig beurteilen, wenn man ihre Erfahrungen bei der Benutzung solcher Systeme und ihre körperlichen Fähigkeiten kennt.
- Aufgaben: Jeder Nutzer im Smart Environment strebt bestimmte Ziele an, zu deren Erfüllung er bestimmte Aufgaben ausführt. Die reine Betrachtung physischer Ereignisse (wie Fingerbewegungen oder Ortsveränderungen) sind schwer interpretierbar. Die Abstraktion zu höherwertigen Informationen (wie Aufgaben und Zielen) ist hilfreich [HR00]. Die Betrachtung auf dem höheren Abstraktionsniveau von Aufgaben ist sowohl für die Modellierung, als auch für die Evaluation vorteilhaft. Aufgaben lassen sich Rollen zuordnen, die schließlich von Nutzern eingenommen werden können.
- Geräte: Nutzer können zur Ausführung von Aufgaben Werkzeuge zur Hilfe nehmen. Dies können persönliche Geräte sein, die Nutzer selbst mitbringen, oder Geräte, die bereits im Smart Environment vorhanden sind. Für die Ausführung von Aufgaben kann der aktuelle Zustand eines Gerätes notwendig sein, während anders herum betrachtet während der Ausführung einer Aufgabe der Zustand eines Gerätes verändert werden kann.
- Ort: Die Umgebung selbst spielt eine große Rolle bei der Ausführung von Aufgaben. So können manche Aufgaben einen bestimmten Ort voraussetzen. Zudem beeinflusst die Architektur und die Platzierung von Gegenständen die Interaktionen der Nutzer.
- Domänenobjekte: Je nach Umgebung und Aufgaben können Domänenobjekte relevant sein, wie Vortragsfolien oder eingehende Emails.

Da eine Integration zwischen Entwicklung und Evaluation, sowohl konzeptionell, als auch auf Ebene der Werkzeuge, angestrebt wird, soll die Form die Funktionsweise des Smart Environments kurz dargestellt werden. Während Nutzer im Smart Environment interagieren, soll die Umgebung den Nutzern Assistenz anbieten bei der Erreichung ihrer Ziele. Dabei spielt die zielbasierte Interaktion eine wichtige Rolle [AE06 (S.331)]. In der Umgebung installierte Sensoren nehmen den Weltzustand wahr, insbesondere die Positionen der Nutzer, aufgenommene Geräte und Gegenstände, die Temperatur, sowie Zustände von Geräten, Fenstern und Türen. Auf Basis dieser Sensordaten versucht die Intentionserkennung mögliche Nutzerziele abzuleiten, wie „Nutzer A möchte einen Vortrag halten“. Die Strategieplanung ermittelt die Fähigkeiten der vorhandenen Geräte, in diesem Falle die der Laptops, Projektoren und Leinwände, und versucht eine Konfiguration des Raumes

herzustellen, die die Nutzer unterstützt. Hier könnten bspw. die Folien vom Laptop des Vortragenden auf die Leinwand neben ihm projiziert werden.

Während der Planung und Entwicklung des Smart Environments ist noch keine physische Umgebung vorhanden, die für die Evaluation genutzt werden kann. Daher soll ein virtuelles Smart Environment entwickelt werden, dass die Evaluation begeleitend zur Entwicklung erlaubt.

3.3.2 Architektur

Im Rahmen dieser Arbeit wurde das in Abbildung 3.2 dargestellte ViSE-System („Virtuelles Smart Environment“) entwickelt. Es stellt dem realen Smart Environment ein virtuelles gegenüber, das für die Evaluation relevante Aspekte abbildet.

Der Modellierer entscheidet für die konkrete Anwendung über die einzubeziehenden Merkmale der beteiligten Nutzer, Geräte, Gegenstände und Umgebung. Für die Nutzer wird ein Handlungsvorrat mittels kooperativer Aufgabenmodelle beschrieben. Geräte und Gegenstände werden durch ihre physischen Eigenschaften, wie Größe und Farbe beschrieben. Geräte besitzen zudem einen Zustand, der sich durch die Ausführung von Aufgaben ändern kann. Die Umgebung wird durch ein Ortsmodell beschrieben, das für die Aufgabenausführung relevante Orte festlegt und bauliche Besonderheiten, wie Wände und Türen spezifiziert.

Aktiviert man im virtuellen Smart Environment die Interaktionen von Nutzern, werden die entsprechenden Interaktionsdaten (bspw. Ortsveränderung oder Aufnehmen eines Gegenstandes) an die ViSE-Engine übergeben. Diese kann wahlweise vier verschiedene Funktionalitäten erfüllen:

(1) Während früher Phasen der Entwicklung liegen erste kooperative Aufgabenmodelle vor, die animiert werden, um die Erfüllung der Anforderungen zu validieren. Soll im virtuellen Smart Environment eine Aufgabe ausgeführt werden, prüft die ViSE-Engine, ob der aktuelle Kontext die Vorbedingung der Aufgabe erfüllt. Ist dies der Fall wird die Aufgabe ausgeführt und spezifizierte Effekte ändern den Zustand der Umgebung. Diese Änderungen werden zur Visualisierung an das virtuelle Smart Environment zurückgegeben.

(2) Zur Evaluation, wie die gewünschte Assistenz in einem physischen Smart Environment subjektiv auf Nutzer wirkt können „Wizard of Oz“-Experimente durchgeführt werden. Dazu begeben sich Nutzer in die nichtinstrumentierte Umgebung, während ein Experte im Hintergrund die Interaktionen beobachtet und die Geräte entsprechend der gewünschten Assistenz so steuert, als sei die Assistenz bereits verfügbar. Dazu kann der Experte bspw. einen Projektor im virtuellen Smart Environment einschalten, den Videoausgang des Präsentierenden mit diesem Projektor verbinden und die entsprechende Leinwand herunterfahren. Die ViSE-Engine leitet diese Zustandsänderung als Steuerungsdaten an die Hardware in der physischen Umgebung weiter und steuert dort die betreffenden Geräte.

(3) Sind im physischen Smart Environment erste Softwarekomponenten (Intentionserkennung, Strategieplanung) implementiert, kann deren Funktionalität evaluiert werden. Interaktionsdaten aus dem virtuellen Smart Environment werden in künstliche Sensordaten transformiert. Bspw. können für das Aufnehmen und Ablegen von Gegenständen RFID-

Daten erzeugt werden, die künstlich mit Fehlern versehen werden, wie sie in physischer Umgebung beobachtet werden können. Diese Daten dienen als Eingabe für die Intentionserkennung, die Aufgaben der betreffenden Nutzer vorhersagt. Diese Aufgaben werden von der ViSE-Engine zur Animation des kooperativen Aufgabenmodells verwendet. Die Visualisierung des animierten Modells erfolgt im virtuellen Smart Environment.

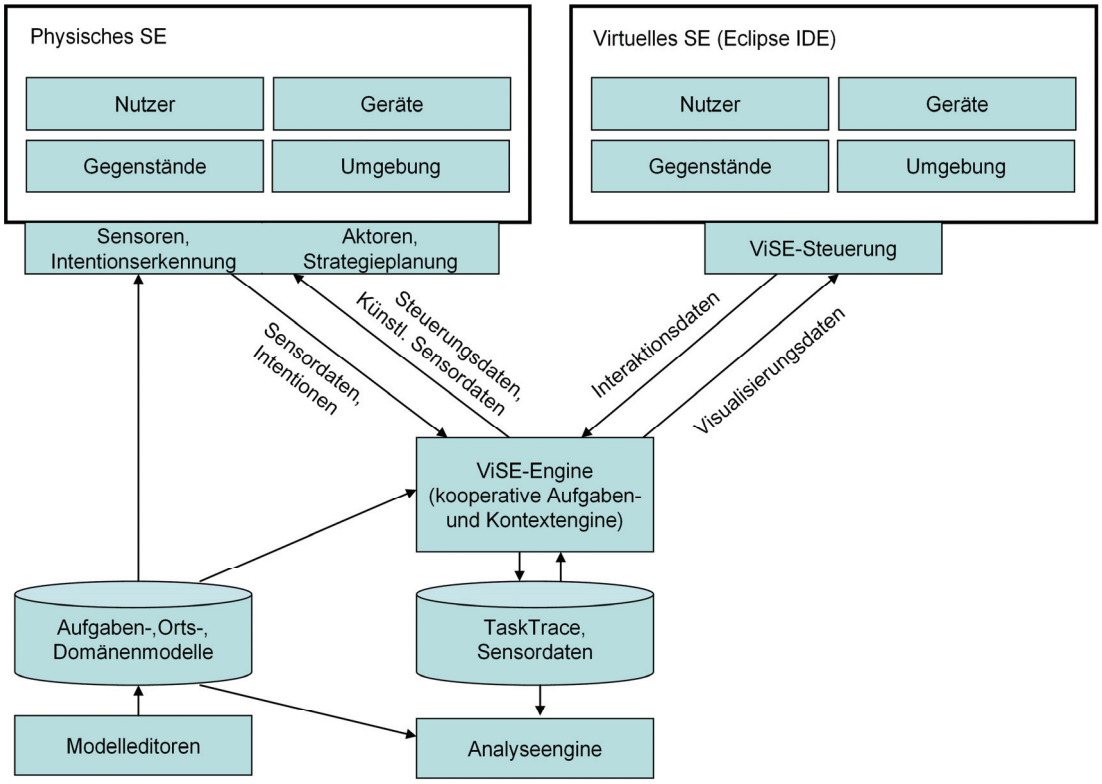

Abbildung 3.2: Architektur des implementierten ViSE-Systems

(4) Ist nach Abschluss der Implementation das komplette physische Smart Environment eingerichtet, werden darin Nutzerstudien durchgeführt. Währenddessen erkennen die eingebauten Sensoren Nutzerinteraktionen. Diese Sensordaten werden von der physischen Umgebung an die ViSE-Engine weitergeleitet, die eine Visualisierung im virtuellen Smart Environment vornimmt. Diese Darstellung der Interaktionen kann zugleich zur Anonymisierung der Testteilnehmer dienen, um die Anforderung an den Schutz persönlicher Daten zu berücksichtigen.

Während der verschiedenen Evaluationen übernimmt die ViSE-Engine die Koordination. Gleichzeitig werden durchgeführte Aufgaben und Änderungen im Kontext aufgezeichnet als TaskTrace bzw. Sensordaten. Diese lassen sich durch bereitgestellte Analysewerkzeuge aus Sicht der Aufgabenausführung auswerten. So können bspw. Nutzer oder Geräte im virtuellen Smart Environment ausgewählt werden, um die damit ausgeführten Aufgabensequenzen anzuzeigen.

Bei identifizierten Schwächen der animierten Modelle lassen sich diese über Modelleditoren verbessern und anschließend erneut animieren, um den Erfolg beurteilen zu können.

3.4 Beispiel: Besprechung in einem Smart Environment

Die entwickelte Methode soll in den nachfolgenden Kapiteln beschrieben werden. Zur Illustration wird ein Beispiel verwendet:

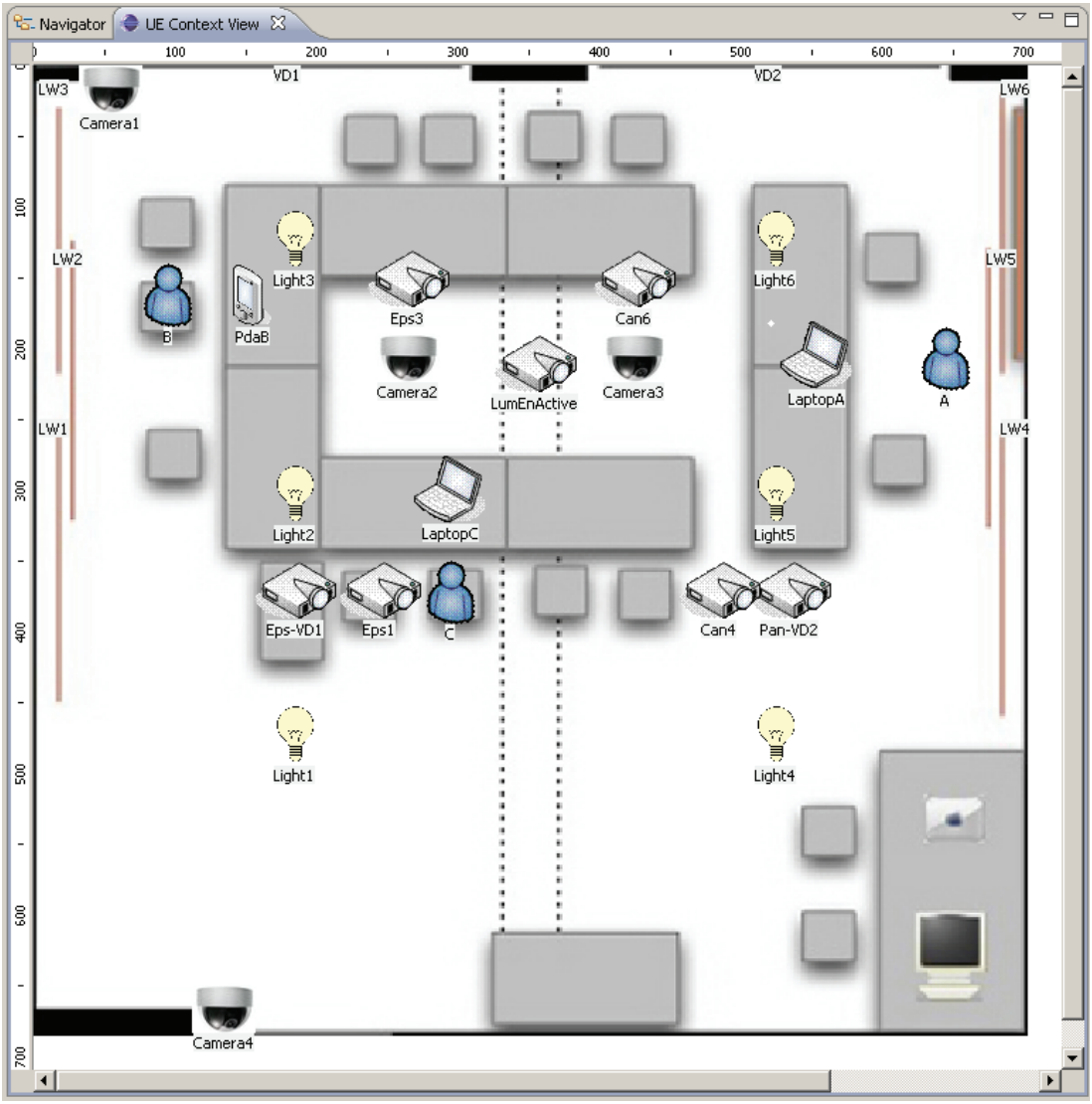

Abbildung 3.3: Schematische Draufsicht auf das „Smart Environment“-Labor

Zur Planung eines neuen Gebäudes für den Fachbereich Informatik wird eine Besprechung durchgeführt. Anwesend sind Vertreter der Universität, der Landesregierung und des Architektenbüros. Es sollen verschiedene Vorschläge für die Konzeption eines Gebäudes diskutiert werden, jeweils im Kontext von Budget und sinnvoller Nutzbarkeit. Dazu halten drei der Anwesenden jeweils einen Vortrag zu (A) dem geplanten Budget, (B) dem Bedarf an Räumen und Ausstattung, sowie (C) den baulichen Möglichkeiten und daraus resultierenden Kosten. Eine Agenda für die Besprechung sieht vor, dass alle drei ihren Standpunkt

präsentieren, wobei die Reihenfolge von den Beteiligten frei gewählt werden kann. Während der Präsentation stehen Projektoren und Leinwände zur Verfügung und Zwischenfragen sind möglich. Anschließend erfolgt eine gemeinsame Diskussion. Die Besprechung wird im „Smart Environment"-Labor der Universität Rostock durchgeführt.

Abbildung 3.3 zeigt eine schematische Draufsicht dieses Raumes. Zu erkennen sind stationäre Geräte (wie Projektoren und Leinwände), persönliche Geräte von Nutzern (wie Laptops und PDAs) und die Nutzer selbst. Die gezeigte Konfiguration zeigt 6 Leinwände und 2 Fensterverdunklungen, die ebenfalls als Leinwand genutzt werden können. Diese sind an drei Wänden im Raum verteilt, so dass die vorhandenen Projektoren anzuzeigende Inhalte gleichzeitig darstellen und im jeweiligen Blickfeld der Nutzer darstellen können.

In der dargestellten Situation befinden sich zwei Teilnehmer auf ihren Plätzen, während einer vor dem Auditorium eine Präsentation hält.

3.5 Zusammenfassung

Bestehende Ansätze zur Evaluation von Smart Environments unterstützen jeweils nur bestimmte Phasen der Entwicklung. Dabei wird die Evaluation früher Artefakte aus Anforderungsanalyse und Design kaum unterstützt. Daher werden folgende Anforderungen an ein Vorgehensmodell zur Evaluation von Smart Environments formuliert, um diese Lücke zu schließen:

- Unterstützung früher Phasen des Entwicklungsprozesses
- Konkrete Ausgestaltung des Paradigmas der „Benutzer-orientierten Gestaltung"
- Aufgabenbasierter Prozess
- Integration von Entwicklung und Evaluation
- Berücksichtigung des Datenschutzes

Auf dieser Basis wird ein Vorgehensmodell entwickelt, das einen iterativen Entwicklungs- und Evaluationsprozess für Smart Environments beschreibt. In jeder Iteration werden die jeweils entwickelten Artefakte evaluiert, um Fehler möglichst frühzeitig zu identifizieren und zu beheben. Dabei werden die Phasen Anforderungsanalyse, Design und Implementation näher betrachtet (siehe auch Propp et al. [PF09]).

Für die Unterstützung des Vorgehensmodells mit Werkzeugen wird die Softwarearchitektur eines virtuellen Smart Environments vorgestellt. Dieses unterstützt vier verschiedene Funktionalitäten:

- (1) Animation von kooperativen Aufgabenmodellen und Kontextinformationen für Expertenevaluation durch interaktives Durchlaufen („Walkthrough")
- (2) „Wizard of Oz"-Experimente, um Nutzern eine Vorschau auf die gewünschte Assistenz zu geben

- (3) Generierung von künstlichen Sensordaten zur Evaluation einzelner implementierter Komponenten
- (4) Anonymisierte Visualisierung von aufgezeichneten Daten aus Nutzertests des komplett eingerichteten physischen Smart Environments

Abschließend wird ein Besprechungsszenario eingeführt, das die Fähigkeiten des „Smart Environment“-Labors an der Universität Rostock illustriert und als durchgehendes Beispiel in der vorliegenden Arbeit dient.

Kapitel 4

Anforderungsanalyse

Zu Beginn der Entwicklung eines Smart Environments sind zusammen mit Auftraggebern und zukünftigen Nutzern die Anforderungen zu erheben. In diesem Kapitel wird der Prozess von Nutzerprofilen bis hin zu Aufgabenmodellen diskutiert. Zur Evaluation werden zwei Möglichkeiten vorgestellt. Das interaktive Durchlaufen der Modelle hilft Modellinkonsistenzen aufzudecken und Modelle schrittweise zu verfeinern. Die Durchführung von „Wizard of Oz"-Experimenten dagegen hilft, den Nutzern die gewünschte Assistenz zu demonstrieren, so dass sie die Anforderungen reflektieren können. Abschließend wird diskutiert, welche Probleme man auf diese Weise finden kann und wie man die erstellten Artefakte verbessert.

4.1 Entwicklungsmethode

Während der Anforderungsanalyse werden die Anforderungen an ein Produkt ermittelt, analysiert und anschließend für die weitere Entwicklung festgeschrieben, als Vertrag zwischen Auftraggeber und -nehmer. Bei der Abnahme wird das fertige Produkt (Ist-Zustand) mit den vereinbarten Produkteigenschaften aus der Produktdefinition (Soll-Zustand) verglichen. Balzert [Bal00] definiert Anforderungen wie folgt:

> *„Anforderungen legen die qualitativen und quantitativen Eigenschaften eines Produktes aus der Sicht des Auftraggebers fest."*

Eine besonders kritische Menge an Informationen bei der Anforderungsanalyse ist das Wissen, wie Nutzer ihre Aufgaben bearbeiten, sowohl physisch als auch kognitiv [CB05 (S.459)]. Dieses Wissen bildet die Grundlage dafür, die alltäglichen Arbeitsabläufe und Informationsflüsse bestmöglich zu unterstützen. In diesem Bereich haben sich verschiedene Methoden der Aufgabenanalyse bewährt, die auch hier angewandt werden sollen. Der folgende Abschnitt gibt einen Überblick über etablierte Methoden der Aufgabenmodellierung, auf deren Basis die hier vorgestellte Anforderungsanalyse für Smart Environments aufbaut.

4.1.1 Vergleich: Notationen zur Aufgabenmodellierung

Aufgabenanalyse und -modellierung werden in verschiedenen Anwendungsdomänen betrieben [KA92], die jeweils eigene Perspektiven besitzen und unterschiedliche Einflüsse auf die Entwicklung der Methoden ausüben. Wurzeln der Aufgabenmodellierung finden sich in den Bereichen:

- Kognitive Psychologie / Ergonomie: Aufgabenanalysen helfen zu verstehen, wie Nutzer mit einem System interagieren. Darüber hinaus lässt sich die kognitive Beanspruchung abschätzen [SY98].
- Anthropologie / Ethnographie: Bei der Analyse menschlichen Verhaltens spielen qualitative Methoden eine große Rolle, die entsprechend des Zieles des Designs eines Produktes mit begrenztem Budget angepasst werden [HR98].
- Aufgabenplanung in Organisationen: Die Analyse der Struktur und des Ressourcenbedarfs von Aufgaben hilft bei der Beurteilung der mentalen Beanspruchung der Mitarbeiter und bildet die Grundlage bei der Organisation und Verteilung von Arbeitsaufgaben [KA92].
- Softwareentwicklung: Die Aufgabenanalyse ist ein verbreiteter Ansatz für das nutzerzentrierte Design von Anwendungen [HR98]. Aufgabenmodelle können die Grundlage bilden für modellbasierte Entwicklung interaktiver Systeme [FDR+04].
- Usability: Um Usability von interaktiven Systemen zu beurteilen, können während einer Nutzerstudie die Interaktionen aufgezeichnet werden. Eine Abstraktion dieser Daten auf eine aufgabenorientierte Ebene erleichtert die Interpretation [PRS07] und erleichtert die Integration von Entwicklungs- und Usability-Methoden [PBF09].

Eine Reihe von Notationen zur Aufgabenmodellierung wurden entwickelt (siehe: [LV04, JLM+08, Gie09]), die sich im Grad der Formalisierung, ihrer Darstellung (textuell oder grafisch), der Granularität ihrer Beschreibungen und der Ausdrucksmächtigkeit unterscheiden. Dennoch ist ein gemeinsames Prinzip erkennbar:

Ein Aufgabenmodell beschreibt, wie Menschen durch die Ausführung von Aufgaben ein bestimmtes Ziel erreichen können und beinhaltet die „statische und dynamische Organisation von Arbeit“ [For07]. Aufgaben lassen sich dabei typischerweise hierarchisch in Unteraufgaben zerlegen (statische Organisation). Wie weit diese Zerlegung geht, hängt vom konkreten Anwendungsfall und gewünschten Abstraktionsniveau ab. Zudem legen bei den meisten Ansätzen temporale Abhängigkeiten zwischen den Aufgaben eine zeitliche Reihenfolge fest (dynamische Organisation), die in manchen Notationen durch zusätzliche Bedingungen eingeschränkt wird, so dass bspw. Nutzer eine bestimmte Rolle haben müssen oder ausreichende Ressourcen vorausgesetzt werden. Einige für die Entwicklung der Aufgabenmodellierung wichtige Notationen sollen nachfolgend kurz vorgestellt werden:

Hierarchical Task Analysis (HTA)

HTA [AD67] ist einer der ersten Ansätze zur Modellierung von Aufgaben und wurde ursprünglich für den Ausbildungsbereich entwickelt. Aufgaben lassen sich hierarchisch in

Teilaufgaben zerlegen und sogenannte Pläne beschreiben die zeitliche Abfolge von Unteraufgaben. Die Beschreibung der Pläne erfolgt nicht formal, wodurch die Ausdrücke sehr mächtig sein können, aber auch möglicherweise nicht eindeutig sind.

Goals, Operators, Methods, Selection Rules (GOMS)
Card et al. [CMN83] entwickelten GOMS als ein Modell zur Beschreibung menschlichen Verhaltens bei der Ausführung von Aufgaben, wobei quantitative Schätzungen der Ausführungszeit möglich sind. GOMS besteht aus einer Menge von (a) Zielen, die einen gewünschten Zustand beschreiben, (b) Operatoren, die elementare Handlungen zur Änderung der Umgebung oder der mentalen Vorstellung der Nutzer darstellen, (c) Methoden, als Sequenzen von Operatoren der Aufgabenausführung und (d) Auswahlregeln, die für eine gegebene Situation beschreiben, welche Methode verwendet wird. Aufbauend auf dem von Card et al. definierten GOMS entstanden weitere Methoden, wie GOMSL und CPM-GOMS [LV04].

Méthode Analytique de Description de tâches (MAD*)
MAD* [SPG89] unterscheidet zwei Aufgabentypen: zusammengesetzte und elementare Aufgaben. Eine zusammengesetzte Aufgabe wird in Unteraufgaben zerlegt, deren Reihenfolge durch Operatoren ausgedrückt wird. Aufgaben besitzen einen initialen Zustand, der durch notwendige und hinreichende Vorbedingungen beschrieben wird, und einen finalen Zustand, der Nachbedingungen für eine erfolgreiche Durchführung der Aufgabe beschreibt. Dabei kann ein Modell von Domänenobjekten und für diese Aufgabe verantwortliche Nutzer referenziert werden. Der Editor ALICE gibt Werkzeugunterstützung.

Task Knowledge Structures (TKS)
TKS [JJ89] erlaubt wie MAD* die Dekomposition von Aufgaben und Definition von Ausführungsfolgen. Temporale Beziehungen werden hier darüber hinaus formal durch die Prozessalgebra CSP definiert. Analog zu Aufgaben werden auch Ziele in Unterziele zerlegt. Jedem Unterziel wird eine Unteraufgabe zugeordnet, wodurch zwei sich überlappende Strukturen entstehen. Jedes Ziel wird als CSP-Prozess betrachtet und sein Verhalten durch eine Prozessgleichung beschrieben. Das Nutzermodell assoziiert Nutzer und Rollen und ordnet Aufgaben zu.

Groupware Task Analysis (GTA)
GTA [VLB96] verbindet Konzepte aus der Mensch-Computer-Interaktion (HCI) und ethnografische Methoden, wie sie im Bereich kooperativer Arbeit (CSCW) eingesetzt werden. Darauf aufbauend erlaubt GTA die Modellierung der Aufgabenausführung in kooperativen Umgebungen. Dafür wurden bei der Dekomposition von Aufgaben in Unteraufgaben kooperative Aufgaben eingeführt. Ein Rollenkonzept erlaubt es, jedem Nutzer Rollen zuzuordnen, die mit einer Menge von Aufgaben assoziiert sind. Aufgaben können entweder frei zugeordnet werden oder eine Organisation abbilden. Aufgaben werden entsprechend dieser Rollenzuordnung ausgeführt. Das Werkzeug Euterpe bietet Softwareunterstützung.

ConcurTaskTrees (CTT)
CTT [Pat99] besitzt wie die anderen Ansätze Aufgabenhierarchien und eine zeitliche Reihenfolge zwischen Unteraufgaben. Diese Reihenfolge wird aber im Unterschied zu anderen Ansätzen nicht in den Knoten der Aufgabenhierarchie für alle Kinder definiert, sondern jeweils zwischen zwei benachbarten Knoten annotiert. Wie bei TKS sind die temporalen Operatoren über eine Prozessalgebra definiert. Die Operatoren sind über LOTOS definiert, wobei CTT eine größere Menge an Operatoren bietet als TKS. Eine Besonderheit bei CTT ist die Annotation von Plattformen, wie Computer oder PDA, an Aufgaben. Dadurch wird eine Art Schablone über das Aufgabenmodell gelegt, dass die für das jeweilige Gerät nicht verfügbaren Aufgaben ausblendet. Als Softwarewerkzeug steht CTTE [MPS02] zur Verfügung.

Visual Task Model Builder (VTMB)
VTMB [BBS99] ist eine Notation und ein gleichnamiges Werkzeug zur Erstellung von hierarchischen Aufgabenmodellen. Temporale Operatoren werden in einem Aufgabenknoten definiert, um die Reihenfolge der Ausführung der Aufgaben in den Unterknoten festzulegen. Die Anzahl der Operatoren ist allerdings kleiner als bei CTT. Eine Aufgabe ist dann ausführbar, wenn der temporale Operator dies erlaubt und zusätzlich Vorbedingungen an die aktuelle Situation erfüllt werden. Dazu beschreibt ein Umgebungsmodell eine Menge von Objekten mit Attributen. Bei der Ausführung einer Aufgabe können Effekte diesen Weltzustand verändern und damit die Vorbedingungen für die nächste Aufgabe erfüllen.

Im Vergleich wird deutlich, dass die Notationen aus verschiedenen Bereichen stammen, bspw. HTA und GOMS aus der kognitiven Analyse, GTA aus CSCW, sowie TKS und CTT aus der Softwareentwicklung. In dieser weiteren Arbeit wird CTT in adaptierter Form verwendet. CTTE ist ein frei verfügbares Werkzeug für die Aufgabenmodellierung, das in Universitäten und der Wirtschaft international weit verbreitet ist. In der Internetpräsenz des W3C findet sich darüber hinaus ein Vorschlag, die CTT-Notation zu standardisieren [Pat08]. Die Operatoren sind formal auf der Basis von LOTOS definiert und bieten eine große Ausdrucksmächtigkeit, die größer ist als die vieler anderer Notationen [LV04]. Andere Notationen bieten in Teilbereichen zusätzliche Sprachkonzepte, wie GOMS, dass mentale Vorgänge modelliert, um die Abschätzung von Zeitverhalten für die Aufgabenausführung zu erlauben, was für das Anliegen dieser Arbeit allerdings nicht notwendig ist. Für CTT wurden zudem von verschiedenen Forschergruppen Erweiterungen entwickelt, insbesondere zur Modularisierung von CTT-Modellen [LM07], zum Einweben von „Error Pattern“ [BB06] und zur Einführung weiterer temporaler Operatoren [FDM03, Luy04, SWF+07].

4.1.2 Ausgewählte Modellierungsnotation

Nachdem CTT als Notation für die weitere Arbeit ausgewählt wurde, soll die verwendete Notation und die von Reichart [RF08] durchgeführten Adaptionen dargestellt werden, so wie auch in dieser Arbeit verwendet.

Die technische Dokumentation von CTT [PS01] nennt die in Tabelle 4.1 aufgeführten Operatoren. Sie sind von den LOTOS-Operatoren (Language Of Temporal Ordering Specification) [ISO88] abgeleitet. In dieser Arbeit werden alle Operatoren außer die mit Informationsaustausch verwendet. Um die ähnlichen Symbole für „choice“ und „option“ klar auseinanderhalten zu können, wird „option“ als „?“ dargestellt. Hinzugefügt wurde durch Reichart [RF08] die Instanziteration (Symbol: „#“), die das mehrfache Instanziieren einer Aufgabe erlaubt, bspw. das mehrfache Ausführen von „Vortragsfolien bearbeiten“. Jede Instanz der Aufgabe hat die gleiche Struktur, also die gleichen Unteraufgaben, aber einen anderen Zustand, der einen anderen Fortschritt in der Bearbeitung erlaubt.

Operator	Notation	Semantik
Concurrency	T1 ||| T2	ausführbar in beliebiger Reihenfolge
Concurrency with Information Exchange	T1 |[]| T2	ausführbar in beliebiger Reihenfolge und Synchronisation über Informationsaustausch
Choice	T1 [] T2	genau eine Aufgabe muss gewählt werden
Disabling	T1 [> T2	sobald T2 begonnen wird, ist T1 endgültig beendet
Enabling	T1 >> T2	nachdem T1 beendet wird, ist T2 ausführbar
Enabling with Information Passing	T1 []>> T2	nachdem T1 beendet wird, ist T2 ausführbar, zudem Informationsweitergabe
Suspend / Resume	T1 |> T2	T2 kann T1 unterbrechen und nachdem T2 beendet ist, wird T1 im alten Zustand weiter ausgeführt
Iteration	T1*	T1 kann wiederholt ausgeführt werden, bis ein „disabling“-Operator die Wiederholung abbricht
Option	[T1]	Ausführung ist optional
Recursion		Das wiederholte Auftreten einer Aufgabe in ihrem eigenen Aufgabenunterbaum erlaubt das rekursive Ausführen der Aufgabe, bis sie unterbrochen wird
Order Indpendence	T1 |=| T2	T1 und T2 werden in beliebiger Reihenfolge jeweils komplett ausgeführt

Tabelle 4.1: CTT Notation (in Anlehnung an [PS01])

CTT sieht die Annotation der binären Operatoren zwischen zwei benachbarten Aufgaben vor. Bei mehreren Geschwisterknoten mit verschiedenen Operatoren muss der Leser die Prioritäten der Operatoren beachten, um die Reihenfolge richtig zu interpretieren. In dieser Arbeit werden die Operatoren als eigene Knoten hierarchisch im Aufgabenbaum dargestellt, so dass sich die Priorität der Teilausdrücke direkt über die Ober-/ Unterknoten ablesen lässt. Die Ausdrucksmächtigkeit wird nicht beeinflusst, da es sich um eine reine Präsentationsfrage handelt.

Diese vier verschiedenen Aufgabentypen von CTT werden verwendet:

- Nutzeraufgabe: kognitive Aufgabe des Nutzers, bspw. Auswählen der geeigneten Problemlösungsstrategie

- Applikationsaufgabe: vom System eigenständig ausgeführt, bspw. Generierung von Anfrageergebnissen
- Interaktionsaufgabe: Interaktion mit dem System (Ein- / Ausgabe), bspw. Bearbeiten einer Tabelle
- Abstrakte Aufgabe: Aufgabe mit Unteraufgaben verschiedener Kategorien

Die Struktur der hier verwendeten Aufgabenmodelle wird durch ein Metamodell (UML-Notation) in Abbildung 4.1 dargestellt. Die Wurzel des Aufgabenbaumes ist als Aufgabe definiert und darunter können beliebig viele „TaskComposites“ folgen, die Aufgabe, n-ärer temporaler Operator oder unärer Iterationsoperator sein können.

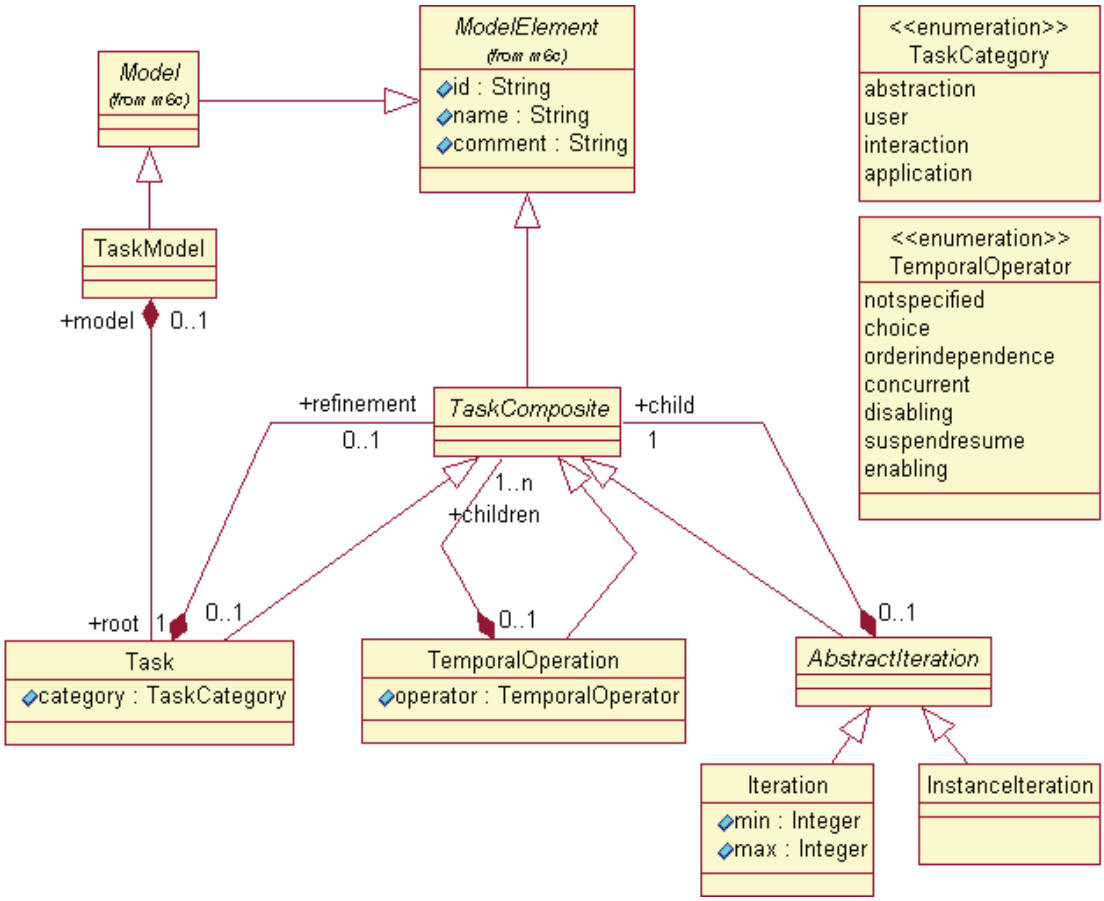

Abbildung 4.1: Metamodell des Aufgabenmodells (in Anlehnung an [RF08])

Das nachfolgende Beispiel verdeutlicht die Notation (Abbildung 4.2). Mit Hilfe des Aufgabenmodelleditors [RF08] wird das Aufgabenmodell für einen Vortragenden erstellt. Dazu wird die Wurzelaufgabe „Halten einer Präsentation“ hierarchisch in Unteraufgaben und temporale Operatoren zerlegt. Operatoren können hierarchisch aufeinander folgen, zwischen zwei Aufgabenebenen ist jeweils ein Operator nötig, der die Beziehung definiert. Dies wird durch einen OCL-Ausdruck sichergestellt: „context Task inv: not self.refinement.oclIsTypeOf(Task)“. Bei der Benennung der Aufgaben sollte man eindeutige

Namen verwenden, da die Wiederholung eines Aufgabennamens an anderer Stelle bei CTT [PS01] für die Wiederverwendung derselben Aufgabenstruktur vorgesehen ist. Außerdem sollte man bei der Verwendung der Iteration besonders aufmerksam sein. Um eine Iteration beenden zu können muss, ein „Disabling“-Operator folgen, was man hier zwischen „give_a_talk“ und „finish_talk“ sehen kann.

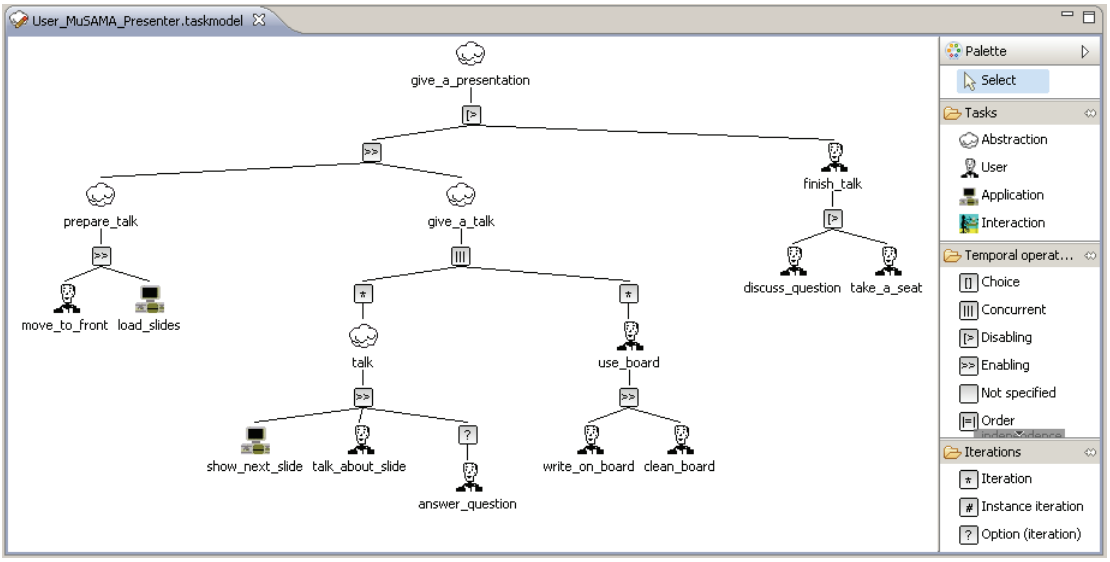

Abbildung 4.2: Beispiel Aufgabenmodell „Geben einer Präsentation“

4.1.3 Aufgabenanalyse

Nachdem die gewählte Notation zum Aufzeichnen von Aufgaben erläutert wurde, soll nun die verwendete Methode zur Aufgabenanalyse dargestellt werden. Die Aufgabenanalyse dient dem Verstehen und Konstruieren von Arbeitssystemen [Her09 (S:20ff)], hier dem Verständnis eines Smart Environments.

Um die Handlungsabläufe von Nutzern zu verstehen, gibt es verschiedene Methoden, bspw. Beobachtung des Nutzerverhaltens in Feldstudien, Befragung von Nutzern während ihrer Arbeit, Fragebögen oder Analyse von Arbeitsdokumentationen der bisherigen Abläufe. Die dabei zu sammelnden Informationen sind vielfältig [CB05] und umfassen bspw. Nutzerziele, Teilaufgaben, Repetitivität, Bedingungen für die Ausführung, besondere Ärgernisse oder Stärken der bisherigen Abläufe, Wünsche an die neue Lösung und der Umgang mit Ausnahmefällen. Dadurch soll ein möglichst vollständiges Bild der Aufgaben, Nutzer und ihrer Umgebung entstehen.

Zur Planung eines Smart Environments kann man sich zunächst die bisherige, nichtinstrumentierte Umgebung der Nutzer ansehen. Im Falle des Entwurfes eines Besprechungsraumes kann man eine Reihe von Besprechungen in der bisherigen Umgebung beobachten, um ein Verständnis zu gewinnen, wie Besprechungen ablaufen, welche Aufgabensequenzen durch Assistenz unterstützt werden können, welche persönlichen Geräte bevorzugt mitge-

bracht werden und welche Kontexteinflüsse wirken. Beobachtungen im bisherigen Besprechungsraum sollte man kritisch hinterfragen, ob dieses Verhalten aus Nutzersicht wünschenswert ist oder der aktuellen Umgebung geschuldet ist. Befragungen der Gesprächsteilnehmer ergeben ergänzend weitere Einblicke aus der Sicht der Nutzer. Im Hinblick auf die Abbildung der Abläufe als Aufgabenmodell sind zunächst die zu unterstützenden Aufgaben zu identifizieren, in Reihenfolgen anzuordnen und schließlich daraus eine Hierarchie zu bilden [HR98]. Eine Frage dabei ist das Finden der richtigen Granularität bei der Zerlegung von Aufgaben in Teilaufgaben. Bei der Auswahl der Aufgaben sollten solche gewählt werden, die auf unterer Ebene direkt mit dem Smart Environment in Beziehung stehen, bspw. direkt unterstützte Tätigkeiten wie „Licht einschalten“ oder „Inhalt der eigenen Laptopanzeige auf Leinwand darstellen“. Als Werkzeugunterstützung für die Modellierung dient der in Abbildung 4.2 vorgestellte Aufgabenmodelleditor.

Am Ende der Anforderungsanalyse sollen folgende Artefakte vorliegen: Jede Nutzerrolle im Raum, bspw. Vortragender und Teilnehmer, sind als separate Aufgabenmodelle zu modellieren. Um in dieser Phase möglichst wenig Aufwand in den Umgang mit Modellierungswerkzeugen zu stecken, sondern mehr Zeit für Gespräche mit Nutzern zu haben, genügen zunächst separate Modelle für jeden Nutzer. Kooperative Aspekte werden dabei textuell erfasst und später in der Designphase formal spezifiziert. Diese Vorgehensweise bietet zudem den Vorteil, dass Interaktionsdesigner oder Usability-Experten ohne umfassende Kenntnisse in der Softwaremodellierung einen schnellen Einstieg finden, da Aufgabenmodelle intuitiv erstellt werden können. Ergänzend können textuell beschriebene Szenarien notiert werden, die ergänzende Details nichtformal festhalten und später als Testfall zum Validieren von Prototypen und fertigem System dienen. Ergänzend zur reinen Aufgabenanalyse stehen weiter Verfahren zur Verfügung, wie die Erstellung von Personas, Umgebungsprofilen, Nutzerwunschtabellen und Kulturkapseln („culture capsules“) [CRW07 (S.940ff)].

4.2 Usability-Methode

4.2.1 Aufgabenmodellsimulator

Ziel der Evaluation ist es, sicherzustellen, dass die erstellten Aufgabenmodelle die realen Arbeitsabläufe möglichst naturgetreu abbilden. Dazu werden die Modelle animiert und gegen konkrete Beispielszenarien validiert, um qualitative Evaluationsergebnisse über die Güte des Models zu erhalten. Experten und Nutzer wählen eine Menge von vorab nichtformal beschriebenen Szenarien und durchlaufen die Aufgabenmodelle interaktiv. Der von Reichart et al. [RFD04] entwickelte Aufgabenmodellsimulator wurde dafür verwendet und um die Aufzeichnung der ausgeführten Aufgabensequenzen erweitert. Abbildung 4.3 zeigt den Simulator.

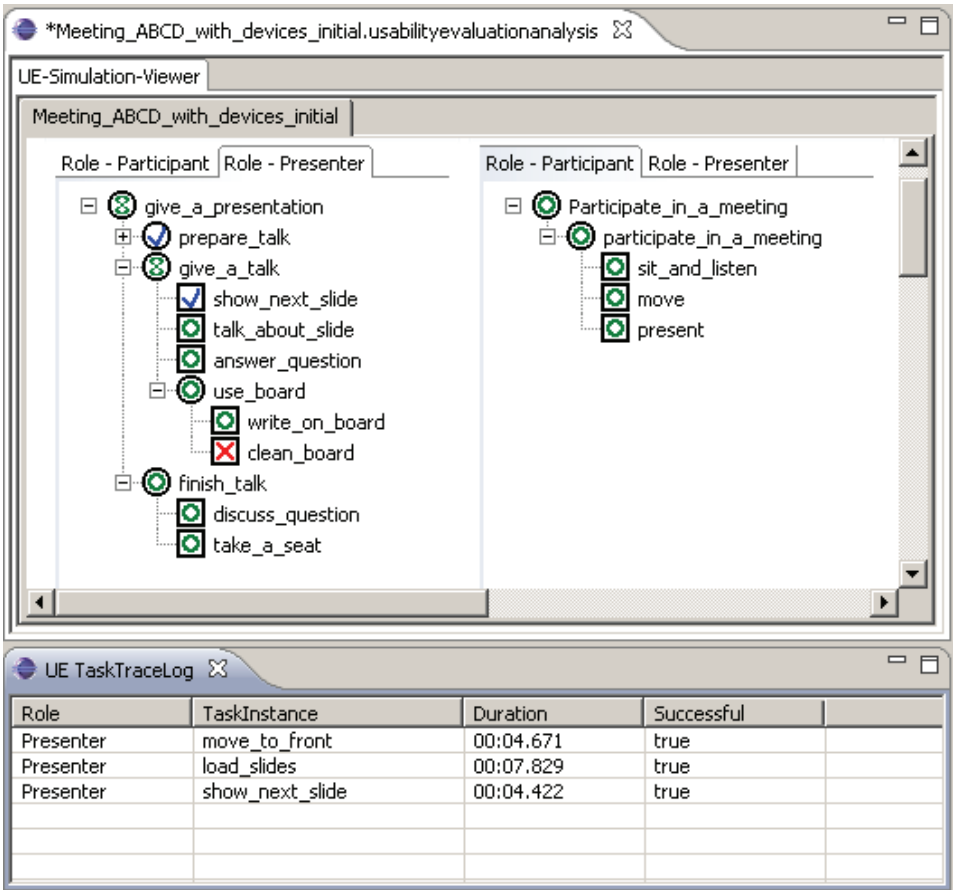

Abbildung 4.3: Beispielsimulation des Aufgabenmodells „Halten einer Präsentation“ (Instanz des Modells in Abbildung 4.2)

Der Simulator zeigt die Instanz eines Aufgabenmodells an. Dabei bleibt die hierarchische Struktur des Modells erhalten und wird um 90 Grad gegen den Uhrzeigersinn gedreht. Symbole vor dem jeweiligen Aufgabennamen repräsentieren den Zustand der Aufgabe: So stellt bspw. ein rotes Kreuz die Aufgabe als nicht aktivierbar dar, ein grüner Kreis als aktivierbar, ein grünes Kreuz als gerade laufend und ein blaues Häkchen als abgeschlossen. Ein Kontextmenü erlaubt das Animieren der weiteren Aufgabenbearbeitung.

Das Verhalten der Aufgaben wird durch den in Abbildung 4.4 dargestellten Zustandsautomaten beschrieben, der den Lebenszyklus der Aufgabe abbildet. Der zielführende Verlauf zum Ausführen einer Aufgabe beginnt im Zustand „disabled“, wechselt nachdem alle Vorbedingungen der Aufgabe erfüllt sind in „enabled“, nach dem Start der Aufgabe durch den Nutzer zu „running“ und schließlich nach erfolgreichem Beenden durch den Nutzer zu „completed“. Die Beschreibung des Verhaltens von Aufgaben durch einen endlichen Zustandsautomaten wurde zuvor bereits in [Bom07] beschrieben und hat sich von der Ausdrucksmächtigkeit her als ausreichend für praktische Anwendungen erwiesen. Der Simulator wurde so erweitert, dass er diese Ereignisse zusammen mit einem Zeitstempel und dem betreffenden Aufgabenmodell aufzeichnet.

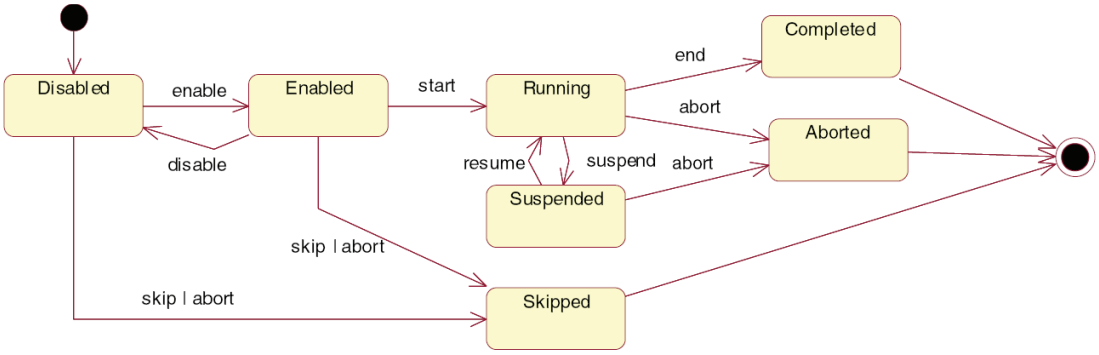

Abbildung 4.4: Zustandsautomat einer Aufgabe (aus [RF08])

Die im Zustandsautomaten verwendeten Ereignisse sind in Tabelle 4.2 aufgelistet. Anhand ihrer Quelle lassen sie sich aus Systemsicht in externe und interne Ereignisse einteilen. Externe werden vom Nutzer initiiert. Dazu stellt der Simulator ein Kontextmenü bereit. Interne werden vom System selbst initiiert. Wenn eine Aufgabeninstanz ihren Zustand wechselt, sendet sie zudem entsprechende Ereignisse an Kinder- bzw. Elternaufgaben im Aufgabenbaum, wodurch diese ebenfalls ihren Zustand ändern können.

Quelle	Ereignis	Semantik
Extern	start	Beginn der Bearbeitung einer Aufgabe
	end	Beenden der Bearbeitung einer Aufgabe
	crash	Abbruch während der Bearbeitung einer Aufgabe
Intern	enable	Versetzen einer Aufgabe in einen ausführbaren Zustand
	disable	Versetzen einer Aufgabe in einen nichtausführbaren Zustand
	suspend	Unterbrechen einer Aufgabe
	resume	Fortsetzen der Aufgabenausführung nach Unterbrechung
	abort	Abbruch vor erfolgreicher Beendigung
	skip	Überspringen einer Aufgabe durch Ausführen einer Alternativaufgabe

Tabelle 4.2: Ereignisse im Aufgabenmodellsimulator

4.2.2 Benutzeranalyse

Entscheidend für Entwicklung und Evaluation ist ein gutes Verständnis der späteren Nutzer. Ein Weg dieses Verständnis zu erlangen erfolgt in drei Schritten [CB05]: zunächst werden Nutzerprofile erstellt, daraus konkrete Personas und schließlich Szenarien festgelegt.

Informationen für die Erstellung von Nutzerprofilen stammen aus vielfältigen Quellen, wie bspw. Befragungen zukünftiger Nutzer, Marktanalysen, Kundendienstmitarbeiter oder Produktmanager. Für jede Nutzergruppe wird ein Profil angelegt, das je nach Anwendung sehr verschiedene Eigenschaften umfasst. Beispiele sind demographische Eigenschaften (Alter, Geschlecht, Wohnort), Ausbildung, technische Erfahrung, Erfahrung mit dem zu

entwerfenden System und entsprechendes Fachwissen in diesem Bereich. Für die Eigenschaften werden meist keine konkreten Werte, sondern Intervalle angegeben, da eine ganze Gruppe von Personen erfasst werden soll. Zudem können auch Prozentzahlen angegeben werden, um zu zeigen, dass die Ausprägung einer Eigenschaft nur für einen Teil der beschriebenen Personen zutrifft. Tabelle 4.3 zeigt beispielhaft das Nutzerprofil für „Informatikstudenten“.

Alter	18-25 Jahre
Geschlecht	90% männlich
Beruf	Student
Berufserfahrung	0-15 Semester (typisch: 6-8 Semester)
Ausbildung	Abitur, naturwissenschaftlich orientierte Schulfächer
Wohnort	Rostock
Erfahrung mit Technik	Sowohl als Anwender, als auch als Entwickler
Körperliche Einschränkung	Keine
Persönliche Geräte	Mobiltelefon, Laptop
Ziel mit dem System	Halten einer Präsentation im Forschungsseminar, Laden der Folien vom eigenen Laptop

Tabelle 4.3: Beispiel eines Nutzerprofils für „Informatikstudenten“

Im Gegensatz zu einem Nutzerprofil beschreibt eine Persona [Coo99 (S.123ff.)] einen konkreten Nutzer, der als Repräsentant für eine Gruppe von Nutzern dient. Ein Nutzerprofil wird dabei mit konkreten Daten gefüllt. Cooper [Coo99] legt Wert darauf, dass nicht am Ende ein durchschnittlicher Nutzer entsteht, der von allen Eigenschaften etwas besitzt und am Ende so nicht existiert, sondern ein in sich stimmiger, typischer Nutzer entsteht. Wenn bspw. das Nutzerprofil eine durchschnittliche Anzahl von 2,3 Kindern angibt, muss man sich für 2 oder 3 Kinder entscheiden. Ziel dabei ist es, einen fiktiven Nutzer zu kreieren, der genauso existieren könnte. Mögliche Eigenschaften von Personas kommen aus fünf Bereichen [Nie07]: Physiognomie, Psyche, sozialer und bildungsbedingter Hintergrund, Emotionen und persönliche Charaktereigenschaften. Der letzte Bereich soll eine flache Darstellung vermeiden und die Persona lebendiger machen. Zudem sollten die Ziele der Persona mit dem zu entwickelnden System genannt werden. Personas sind in der Regel auf das aktuelle Projekt zugeschnitten. Das Beispiel gibt eine Zusammenfassung von „Paul“:

Paul ist 22 Jahre alt und studiert im 6. Semester Diplom-Informatik. Neben dem Studium arbeitet er als wissenschaftliche Hilfskraft am Fraunhofer IGD und hat daher vielfältige Erfahrungen mit Computertechnik. In einem Smart Environment war er allerdings noch nicht. Paul hat bereits Plakate zur „Langen Nacht der Wissenschaften“ gesehen mit der Ankündigung einer Veranstaltung „Wenn der Beamer mit der Lampe spricht“, war aber nicht dort. Daher hat er nur eine vage Vorstellung, dass in solch einem Raum vermutlich eine Form von Intelligenz präsent ist, aber keine Idee, worin sich dies äußert. Paul hat sein Mobiltelefon und seinen Laptop in der Universität immer dabei und nutzt das WLAN

während der Lehrveranstaltungen, um Vortragsfolien abzurufen. Zur Zwischenpräsentation seiner Studienarbeit betritt er erstmals ein Smart Environment, in dem er sogleich seinen Vortrag halten soll.

An Hand solcher Personas können Designalternativen geprüft und Entscheidungen für das Design getroffen werden. Die Persona-Methode ist heute auch in der Wirtschaft weit verbreitet für die Benutzeranalyse [PA06] und wurde seid ihren Anfängen weiterentwickelt. So schlagen Cooper et al. [CRC07] die Unterscheidung von fünf verschiedenen Typen von Personas vor („Primnary“, „Secondary“, „Supplemental“, „Customer“, „Served“ und „Negative“), um differenzierte Perspektiven zu erlauben. Pruitt und Adlin [PA06] betrachten die Verwendung von Personas als Lebenszyklus vom Sammeln der Daten für Personas, über ihre Erstellung und Nutzung, bis hin zur Anpassung oder Ersetzen durch neue Personas. Mulder et al. [MY07] stellen fest, dass es sich bei Personas um eine rein qualitative Methode handelt und schlagen vor, diese auf die Basis quantitativer Daten zu stellen. Sie validieren Personas durch statistische Auswertungen von Interviews und Fragebögen.

Nachdem Personas erstellt sind, lassen sich Szenarien entwickeln, in denen die Personas mit dem zu entwerfenden System interagieren. Eine Geschichte beschreibt eine konkrete Situation und wie die einbezogenen Nutzer interagieren, um vorgegebene Aufgaben zu erfüllen. Ein Beispiel ist das Halten einer Präsentation, unterbrochen durch Zwischenfragen, deren Antworten an einer Tafel illustriert werden sollen. Die so entwickelten Personas und Szenarien sollen nun für das Validieren der Designmodelle verwendet werden.

4.2.3 Vorgehen bei der Evaluation mit dem Aufgabenmodellsimulator

Nachdem der Simulator als Hilfsmittel zur Evaluation der erstellten Aufgabenmodelle der Anforderungsanalyse vorgestellt wurde, soll nun das Vorgehen bei der Evaluation mit Nutzern und Experten erläutert werden.

Die Evaluation erfolgt als interaktives Durchlaufen der entwickelten Modelle. Dabei ist ein Durchlaufen (engl. „Walkthrough“) [Rub94] eine Methode, bei der ein frühes Konzept oder Prototyp schrittweise durchgegangen wird. Dies kann wahlweise durch Experten oder Nutzer erfolgen. Beide Varianten sollen vorgestellt werden.

4.2.3.1 Durchlaufen durch Experten

Die Evaluation kann durch Experten durchgeführt werden, die mit der Anwendungsdomäne vertraut sind. Wie bei einem „Cognitive Walkthrough“ [WRL+94] versetzen sich dabei Experten in die Rolle von Nutzern und vollziehen deren Verhalten schrittweise nach.

Die zuvor erstellten Personas und Szenarien dienen als Vorlage, an Hand der Fachexperten die vorliegenden Aufgabenmodelle mit dem vorgestellten Aufgabensimulator animieren. Dabei werden verschiedene Szenarien verschiedener Personas animiert. Gefundene Probleme werden notiert, um die Modelle später zu verbessern. Im Vordergrund steht dabei die Frage, ob die Interaktionen der Personas gut mit den modellierten Abläufen beschrieben werden.

Diese Methode erlaubt eine schnelle und kostengünstige Evaluation, da keine Nutzer eingeladen werden. Zu bedenken ist dabei, dass auch eine gute Expertenevaluation im Ergebnis nur eine Annäherung an die Einbeziehung realer Nutzer liefern kann. Allerdings ist die Methode sehr flexibel einsetzbar. Eine Anwendung ist der Einsatz als Vortest einer Nutzerstudie, da so grobe Fehler im Vorfeld beseitigt werden können.

4.2.3.2 Durchlaufen durch Nutzer

Alternativ zur Evaluation durch Experten, kann eine Evaluation durch Nutzer erfolgen. Dabei wird die Evaluation mehrfach durchgeführt, so dass jeder Nutzer verschiedene Szenarien durchläuft. Für jeden Test werden ein Nutzer und ein Entwickler, der bei der Anforderungsanalyse beteiligt war, eingeladen. Zudem wird eine Menge von Beispielszenarien festgelegt. Ein Szenario enthält eine kurze Beschreibung der zu erfüllenden Aufgabe und der vorhandenen Umgebung. So können Beschränkungen vorliegen in Bezug auf vorhandene Geräte oder die Ausstattung des Raumes. Das vorliegende Aufgabenmodell wird animiert. Der Nutzer versetzt sich in die konkrete Situation und schaltet im Simulator die Aufgabenausführung weiter. Um Modelle einzelner Rollen zu evaluieren, werden notwendige Zuarbeiten anderer Nutzer als gegeben vorausgesetzt. Durchläuft man bspw. das Aufgabenmodell eines Vortragenden, wird vorausgesetzt, dass der Vorsitzende bereits das Wort erteilt hat. Wenn während der Evaluation Fragen des Testnutzers auftreten, bspw. wenn eine Aufgabe unverständlich betitelt ist oder die Modellanimation einen unerwarteten Verlauf nimmt, steht ein Entwickler zur Seite und beantwortet die Fragen. Der Entwickler notiert solche Probleme. Ist eine Aufgabensequenz so nicht möglich wird diese notiert, um das Modell später entsprechend zu ändern. Dadurch, dass nur ein Nutzer zurzeit eingeladen wird, kann dieser in einem selbst gewählten Tempo den Test absolvieren, ohne dass andere schnellere Nutzer warten müssen und damit den roten Faden ihrer persönlichen Arbeitsweise verlieren. Szenarien zur Evaluation der Kooperation mehrerer Nutzer werden in Kapitel 5.2 erläutert.

Durch den Einbezug von Nutzern in dieser recht frühen Entwicklungsphase können Fehlentwicklungen frühzeitig aufgedeckt werden. Beteiligte Entwickler erhalten zudem Einblicke in die Herangehensweise der späteren Nutzer. Damit Nutzer mit dieser noch recht abstrakten Repräsentation zurecht kommen, ist eine Einführung in die Funktionsweise des Aufgabenmodellsimulators vor dem Test notwendig.

4.2.4 Wizard of Oz

Während der Anforderungsanalyse werden die zu unterstützenden Aufgaben und die Art und Weise der Assistenz festgelegt. Für die Analyse der Aufgaben und deren Struktur wurde im Abschnitt zuvor ein Aufgabensimulator vorgestellt, der eine interaktive Animation der Aufgabenmodelle im virtuellen Smart Environment erlaubt. Die Anforderungen an die Assistenz sollen nun durch „Wizard of Oz“-Experimente [DJA93, LHL07] in der realen Umgebung evaluiert werden.

Ausgangspunkt der Entwicklung ist häufig ein bestehender Raum des Kunden, der für bestimmte Zwecke (bspw. Besprechungen) genutzt wird und nun zu einem Smart Environment ausgebaut werden soll. Um Anforderungen daran zu ermitteln, können spätere Nutzer, der Auftraggeber oder andere Beteiligte befragt werden. Da die meisten zukünftigen Nutzer keine Erfahrungen mit Smart Environments haben, ist es jedoch schwer für sie Wünsche zu äußern. Dazu müssten sie abschätzen, was technisch möglich ist und sich vorstellen, wie diese Assistenz den Arbeitsalltag beeinflussen würde. Um die Möglichkeiten und Konsequenzen der Assistenz besser vorstellbar zu machen, kann man einerseits einen funktionalen Prototyp des späteren Raumes bauen, was allerdings sehr aufwendig ist. Andererseits kann man in einem realen Raum die spätere Funktionalität durch ein „Wizard of Oz"-Experiment simulieren.

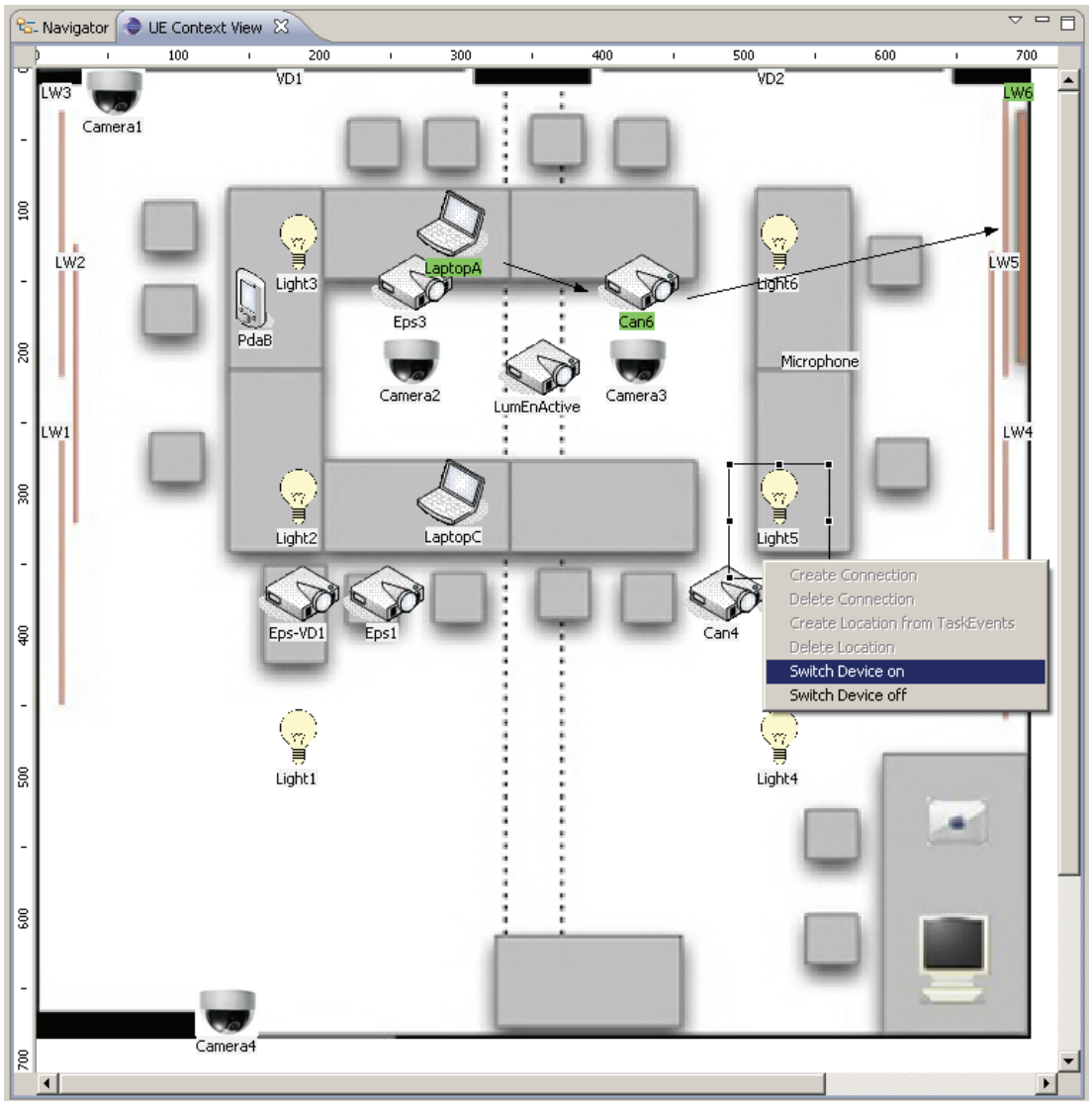

Abbildung 4.5: Steuerung von „Wizard of Oz"-Experimenten

Nutzer sollen dabei Gelegenheit haben, eine Umsetzung ihrer Anforderungen im realen Umfeld zu reflektieren und entsprechend zu ändern oder zu vervollständigen. Ein paar mögliche Fragestellungen dabei sind: Wie „fühlt“ sich die Assistenz in der konkreten Situation an? Ist sie angemessen für die aktuelle Situation? Werden Tätigkeiten automatisiert, die mehr Freude bereiten, wenn man sie weiterhin selbst ausführt? Entsteht ein Gefühl der Bevormundung? Sind Nutzer überrascht, wenn Geräte im Raum ohne explizite Steuerung Aktionen ausführen?

Zur Durchführung des Experimentes wird der zu instrumentierende Raum mit Kameras ausgerüstet, so dass ein Usability-Experte im Nachbarraum die Interaktionen der Nutzer mitverfolgen kann. Es sind keine Sensoren oder andere Geräte im Raum nötig. Die Nutzer können entweder eine reale Besprechung durchführen oder ein Testszenario abarbeiten, dass für möglichst viele der genannten Anforderungen die Auswirkungen praktisch zeigt. Der Usability-Experte sammelt die Anforderungen an die Assistenz und macht sich damit so vertraut, so dass er während der Durchführung die Geräte entsprechend steuern kann. Die Nutzer sollen ein funktionstüchtiges Smart Environment wahrnehmen, so wie es nach der Entwicklung auf Basis der genannten Anforderungen funktionieren würde. Eine Anforderung kann bspw. darin bestehen, dass bei Betreten einer definierten Präsentationszone automatisch die Leinwand 5 herunterfährt. Wenn während der Testdurchführung sich einer der Nutzer entsprechend bewegt, sorgt der Usability-Experte im Hintergrund für die Steuerung der Leinwand.

Zur Steuerung der Geräte wurde das in Abbildung 4.5 dargestellte virtuelle Smart Environment entwickelt. Die Oberfläche zeigt die Draufsicht auf den zu steuernden Raum mit den vorhandenen Geräten. Nach Auswahl eines Gerätes kann über ein Kontextmenü eine Aktion für das Gerät ausgeführt werden. So können bspw. Lampen ein- und ausgeschaltet oder Leinwände und Fensterverdunklungen hoch- und runtergefahren werden. Pfeile zeigen die Verknüpfung von Laptops, Projektoren und Leinwänden zur Übertragung und Darstellung eines Videobildes.

Parallel zu der gezeigten Steuerungssoftware wird in einem Videobetrachter das Bild der im Raum angebrachten Kamera angezeigt, auf dessen Basis der Usability-Experte die Steuerung der Geräte vornimmt und die korrekte Reaktion dieser Geräte überprüft.

4.3 Verbesserung der Usability

4.3.1 Nach der Evaluation durch interaktives Durchlaufen

Ziel des interaktiven Durchlaufens ist das Validieren der erstellten Aufgabenmodelle mit der Frage: Bestehen Abweichungen zwischen Modell und tatsächlichen Arbeitsabläufen von Nutzern? Typische Probleme sind:

- unberücksichtigte Beteiligte im Prozess (weitere Rolle, für die ein Aufgabenmodell benötigt wird)

- fehlende Aufgaben
- zu wenig verfeinerte Aufgaben (Aufgaben sind so zu verfeinern, dass Blätter im Aufgabenbaum direkt vom Smart Environment unterstützt werden)
- fehlende oder unzutreffende temporale Operatoren

Um diese Probleme zu beheben, sind bei Bedarf neue Aufgabenmodelle für weitere Rollen hinzuzufügen und bestehende Aufgabenmodelle zu verändern. Dabei können mit Hilfe eines Aufgabenmodelleditors (Abbildung 4.2) Aufgaben hinzugefügt, entfernt oder weiter verfeinert werden. Darüber hinaus können temporale Operatoren verändert und eingefügt werden. Eine erneute Animation der veränderten Modelle zeigt, ob die Szenarien nun ebenfalls korrekt abgedeckt werden.

4.3.2 Nach der Evaluation durch „Wizard of Oz“-Experiment

Ziel der „Wizard of Oz“-Evaluation ist es, den zukünftigen Nutzern in einer realen Umgebung einen Ausblick auf die gewünschte Assistenz zu geben, damit diese die Anforderungen überdenken können:

- Erweist sich die gewünschte Assistenz in einem realen Szenario als praktisch? (Soll eine Assistenz für die untersuchte Umgebung eingeführt werden?)
- Ist der Umfang der Assistenz angemessen? (Sollen zusätzliche Aktionen in die Assistenz einbezogen werden oder Funktionen herausgenommen werden? Welche Geräte sollen gesteuert werden, welche nicht?)
- Ist die Art der Interaktion passend? (Soll die Kommunikation mit dem System visuell, akustisch oder haptisch erfolgen?)
- Greift die gewünschte Assistenz in der richtigen Situation ein? (Sollen die Kontextbedingungen für die Assistenz verändert werden?)

Die zu unterstützenden Szenarien werden entsprechend der Rückmeldungen der Nutzer angepasst, und die Evaluation bei Bedarf wiederholt.

4.4 Zusammenfassung

Während der Anforderungsanalyse werden die Eigenschaften des zu entwickelnden Smart Environments festgelegt. Dabei ist es wichtig, ein Verständnis für die Nutzer und deren Arbeitsabläufe zu gewinnen. Zur Durchführung einer entsprechenden Aufgabenanalyse stehen zahlreiche Aufgabenmodellierungsnotationen bereit, von denen CTT für die weitere Arbeit ausgewählt wurde. Nutzer können im Smart Environment verschiedene Rollen einnehmen, wie bspw. Vortragender, Teilnehmer oder Vorsitzender. Für jede Rolle wird ein Aufgabenmodell spezifiziert.

Um die entstandenen Modelle evaluieren zu können, wurde ein bestehender Aufgabenmodellsimulator verwendet und um die Aufzeichnung von TaskTraces erweitert. Die Modelle werden im Rahmen eines interaktiven Durchlaufens animiert. Identifizierte Modellinkonsistenzen werden verbessert und Aufgaben iterativ verfeinert. Die aufgabenmodellbasierte Usability-Evaluation ist im Bereich von Arbeitsplatzapplikationen [PBF08], mobilen Geräten [PRS07] und Internetanwendungen [PP02] eine erprobte Methode und wird in der vorliegenden Arbeit auf Smart Environments ausgeweitet.

Da Nutzer häufig keine Erfahrung mit Smart Environments haben, fällt es ihnen schwer, Anforderungen an eine zukünftige Assistenz zu formulieren. Ein „Wizard of Oz"-Experiment kann in dieser Situation helfen, ein Gefühl für die Assistenz zu geben, so dass Nutzer ihre Anforderungen in einem praktischen Beispiel reflektieren können. Das entwickelte virtuelle Smart Environment dient hierbei als Werkzeugunterstützung.

Kapitel 5

Design

Nachdem die Anforderungen analysiert sind, ist auf dieser Basis ein Design zu spezifizieren. Dies umfasst das Interaktionsdesign und das Design von optional vorhandenen grafischen Benutzungsschnittstellen. Das Kapitel stellt zunächst einen modellbasierten Ansatz für die Entwicklung vor und erläutert anschließend eine Methode zur modellbasierten Usability-Evaluation. Dabei werden dieselben Modelle verwendet, was eine schnelle Iteration aus Erstellung von Modellen, deren Animation, Evaluierung, Verbesserung der Modelle und erneuter Evaluation erlaubt.

5.1 Entwicklungsmethode

Ausgehend von den in der Anforderungsanalyse erstellten Aufgabenmodellen, textuellen Szenarien und weiteren Dokumenten, sollen die Artefakte nun so verfeinert werden, dass eine Implementation möglich wird. Dazu werden die Aufgabenmodelle um kooperative Ausdrücke und Bezüge zum Ausführungskontext erweitert. Eine besondere Rolle spielt dabei der Ort, der bei mobilen Nutzern die Aufgabenausführung beeinflusst. In Besprechungsräumen ist es bspw. typisch, dass der Vortragende vor das Auditorium tritt und so von allen gleichermaßen gesehen werden kann.

Mehrere Autoren haben Vorschläge entwickelt, wie man Aufgabenmodelle erweitern kann, um sie in Smart Environments einsetzen zu können. Eine Reihe relevanter Ansätze werden nachfolgend vorgestellt und anschließend der gewählte im Detail vorgestellt.

5.1.1 Vergleich: Aufgabenmodellierung für Smart Environments

Bei der Aufgabenmodellierung für Smart Environments sind eine Reihe von Besonderheiten zu berücksichtigen. So werden Aufgaben nicht isoliert ausgeführt, sondern sind vom Fortschritt der Aufgabenausführung anderer Nutzer abhängig. Nutzer können verschiedene Rollen besitzen und diese wechseln. Zudem sind Kontextparameter, wie der Aufenthaltsort, vorhandene Geräte und Gegenstände, sowie Zustände der Geräte in Betracht zu ziehen.

Raumaufgabenmodelle

Trapp und Schmettow [TS06] unterscheiden Aufgaben in drei Kategorien: persönlich durchzuführende, an eine Rolle gebundene und vom Ort abhängige. Sie beschreiben die Fähigkeiten für jedes Gerät durch ein Aufgabenmodellfragment („device functionality model“, DFM). Immer, wenn ein Nutzer ein neues Gerät in den Raum bringt, wird das entsprechende Modellfragment zum Raumaufgabenmodell („room task model“, RTM) hinzugefügt. Das RTM ist eine Aggregation aller DFM's und beschreibt damit die durchführbaren Aufgaben an allen Geräten im Raum. Die Kombination mehrerer DFM's kann zur Synthese neuer Aufgaben führen. Bspw. bietet die Kombination von Scannen und Drucken zusammen das Kopieren an. Anschließend wird das RTM mit dem Domänenmodell („domain task model“, DTM) verglichen und alle im DTM nicht vorhandenen Teile werden aus dem RTM entfernt. Die übrig bleibenden Teile des Aufgabenmodells werden als Bedienoberfläche für den Nutzer bereitgestellt.

Interpretation zur Laufzeit

Feuerstack et al. [FBA07] [BLF+08] erweitern Aufgabenmodelle in CTT-Notation zu zustandsbehafteten und ausführbaren Modellen, die zur Laufzeit verwendet werden können. Ergänzend zum Aufgabenmodell werden dafür weitere Modelle verwendet. Das Domänenmodell beschreibt dazu die Struktur und die Laufzeitinformationen der Domänenobjekte. Ein Beispiel ist eine Email mit Betreff, Adressat, Inhalt und Anlage. Der so modellierte Objektfluss dient zur Synchronisation der Aufgabenmodelle der verschiedenen Nutzer. Domänenobjekte können auf zwei Arten verändert werden. Einerseits beschreibt das Servicemodell wie das System bei Applikationsaufgaben auf Objekte zugreift. Andererseits beschreibt das Interaktionsmodell wie Nutzer bei Interaktionsaufgaben auf Objekte zugreifen. Ein ergänzendes Kontextmodell beinhaltet die Sensordaten, die Sensoren in der Umgebung liefern. Alles in allem dient das Aufgabenmodell zum Verfolgen von Aufgabenausführungen in der Umgebung, die den Zustand mehrerer Modelle ändern und letztlich die Generierung einer Benutzungsoberfläche auslösen.

CCTT

Aufbauend auf CTT stellt Paterno CCTT (Cooperative ConcurTaskTrees) [MPS02] als eine Erweiterung zur Beschreibung kooperativer Arbeit vor. Die Aufgabentypen und temporalen Operatoren von CTT bleiben bestehen. Zur Definition von Kooperation lassen sich Aufgaben mit einem neuen Attribut als kooperativ kennzeichnen. Für jede Rolle wird ein separates Aufgabenmodell entwickelt. Ein zusätzliches Kooperationsaufgabenmodell beschreibt die Abhängigkeiten zwischen den Aufgaben der Rollen und sorgt damit für deren Synchronisation. Wie auch in CTT können für jede Aufgabe die Domänenobjekte angegeben werden, die während ihrer Ausführung manipuliert werden [Pat99].

CTML

Die „Collaborative Task Modelling Language“ (CTML) wurde von Wurdel [WPF08] entwickelt und setzt ebenfalls auf der CTT-Notation auf. Jeder Akteur nimmt eine Menge von Rollen ein, die jeweils durch ein Aufgabenmodell beschrieben werden. Die Synchronisa-

tion der Modelle wird nicht wie bei CCTT durch ein zusätzliches Aufgabenmodell erreicht, sondern durch die Annotation von Vorbedingungen und Effekten an Aufgaben. Eine Vorbedingung beschreibt Anforderungen an den Weltzustand, der neben den temporalen Operatoren als weitere Restriktion für das Ausführen einer Aufgabe gilt. Ein Effekt beschreibt eine Änderung des Weltzustandes durch das Ausführen einer Aufgabe, der dann die Vorbedingung für das Ausführen einer anderen Aufgabe schaffen kann. Diese Annotationen können sich auf den Fortschritt der Aufgabenausführung anderer Nutzer beziehen, ein Domänenmodell oder vorhandene Geräte.

MoDIE

Luyten et al. [LBV+06] stellen mit MoDIE („mobile distributed interface engineering“) eine Methode für den Entwurf von Benutzungsoberflächen für intelligente Umgebungen vor. Grundlage sind dabei Aufgabenmodelle in CTT-Notation, zu deren Interaktionsaufgaben abstrakte Beschreibungen von Bedienoberflächen und Interaktionsressourcen annotiert werden. Idee ist dabei, dass Aufgaben nur in einem bestimmten Bereich um eine Ressource ausführbar sind. Ein Editor erlaubt die Visualisierung des Raumes zusammen mit den Beziehungen zum Aufgabenmodell. Ein ergänzendes Kontextmodell wird durch eine Ontologie repräsentiert. Jedes neu hinzutretende Gerät bringt ein entsprechendes Ontologiefragment zur Beschreibung der eigenen Eigenschaften mit. Die Suche nach Interaktionsgeräten erfolgt über „Universal Plug and Play“ (UPnP).

Die vorgestellten Methoden erweitern auf verschiedene Art und Weise bestehende Aufgabenmodellierungsmethoden. Alle genannten Ansätze, außer der der Raumaufgabenmodelle, verwenden die verbreitete CTT-Notation und ein großer Teil verfolgt die Generierung von Bedienoberflächen als primäres Ziel. Im weiteren soll CTML verwendet werden, da diese Notation auf CTT aufbaut und durch ihre OCL-artigen Ausdrücke sehr mächtig ist. CCTT ist im direkten Vergleich besser visualisierbar, bietet aber nicht so umfangreiche Möglichkeiten zum Festlegen von Bedingungen und unterstützt weder das Ausführen einer Rolle durch mehrere Nutzer, noch das Innehaben mehrerer Rollen durch einen Nutzer. Interessant ist bei MoDIE die Modellierung des Kontextes als Ontologie. Im Rahmen dieser Arbeit wurde ebenfalls die Anbindung einer Ontologie an CTML geschaffen.

5.1.2 Ausgewählte Methode

Da im weiteren Verlauf der Arbeit CTML zur Spezifikation kooperativen Arbeitens in Smart Environments verwendet wird, soll die Methode kurz erläutert werden.

Ein kooperatives Aufgabenmodell [WPF08] [Wur09] definiert eine Menge von Akteuren, Geräten, Orten und Domänenobjekten, sowie kooperativen Aufgabenausdrücken. Jeder Akteur kann eine Menge von Rollen besitzen und der Handlungsvorrat einer Rolle wird jeweils durch ein Aufgabenmodell beschrieben. Das Aufgabenmodell liegt wie in Kapitel 4.1.2 beschrieben in CTT-ähnlicher Notation vor, wobei an jeder Aufgabe kooperative Ausdrücke in OCL-ähnlicher Notation annotiert werden können. Bei den Ausdrücken handelt es sich um Vorbedingungen und Effekte. Eine Vorbedingung kann die Vergleichs-

operatoren (=, <>, <, >), die logischen Operatoren (und, nicht, oder) und die Quantoren (All- / Existenzquantor) enthalten. Eine Aufgabe ist genau dann ausführbar, wenn die Kontextbedingung und der temporale Operator erfüllt sind. Eine Vorbedingung kann sich beziehen auf:

- den Ausführungsfortschritt einer Aufgabeninstanz eines bestimmten Akteurs im Raum (bspw. „der Architekt ist mit der Aufgabe ‚take_a_seat' fertig", Bedingung: „Architect.take_a_seat.completed") oder eines Akteurs einer bestimmten Rolle (bspw. „ein Zuhörer führt ‚ask_question' durch", Bedingung: „Participant.oneInstance.ask_question.completed"),
- den Zustand eines Gerätes (bspw. „Projektor 1 ist eingeschaltet", Bedingung: „Projector1.active"),
- den Zustand eines Domänenobjektes (bspw. „Adresse der Email ist gesetzt", Bedingung: „mail.adress != ,'") oder
- den Aufenthaltsort eines Akteurs (bspw. „Architekt befindet sich in der Präsentationszone", Bedingung: „Architect.isLocatedAt.PresentationZone").

Nachdem eine Aufgabe erfolgreich beendet ist, werden annotierte Effekte ausgelöst. Die Effekte können dabei die gleichen Entitäten umfassen wie Vorbedingungen. Im Ergebnis ändert sich der Kontext der Akteure. Bspw. kann bei einem Vortragenden mit Abschluss der Aufgabe „move_to_front" der Projektor, der auf die Hauptleinwand zeigt, eingeschaltet werden. Das dabei verwendete Domänenmodell ist als UML-Klassendiagramm modelliert. Ortsmodelle werden derzeit manuell erstellt. Um diesen Schritt zu vereinfachen, wäre es auch möglich auf bestehende Verfahren der automatischen Extraktion von Modellen aus annotierten Fotos des Raumes zurückzugreifen (bspw.. [RKV+09]).

Der Modellierungsprozess verläuft iterativ durch schrittweises Verfeinern der Aufgabenmodelle.

5.1.3 Metamodell der Aufgabenausführung in Smart Environments

Basierend auf dem Aufgabenmetamodell (vorgestellt in Kapitel 4.1) und den kooperativen Aufgabenausdrücken der CTML (vorgestellt in Kapitel 5.1) zeigt Abbildung 5.1 ein Metamodell, dass die Spezifikation der Aufgabenausführung in Smart Environments erlaubt.

Geräte dienen dabei als Arbeitswerkzeug, mit dem Arbeitsgegenstände („artifacts") bearbeitet werden können. Die modellierten Arbeitsgegenstände können während der Arbeit ihren Zustand ändern, bspw. eine neue Position im Raum einnehmen.

Das „LocationModel" beschreibt den Grundriss des Raumes, also seine Ausdehnung und Form. Zudem kann es den gegebenen Raum in verschiedene Zonen einteilen, die für die Aufgabenausführung von Bedeutung sind. Im Fall von Besprechungen gibt es je nach Aufstellung von Stühlen, Tischen und Leinwänden bspw. Zonen mit Sitzplätzen für das Publikum und meist eine Zone für Vortragende vorne vor dem Publikum.

Das „CollaborationModel" enthält die kooperativen Ausdrücke (Vorbedingungen und Effekte). Die Ausdrücke werden somit separiert vom Aufgabenmodell und können bei Bedarf als Annotation zu Aufgaben geladen werden.

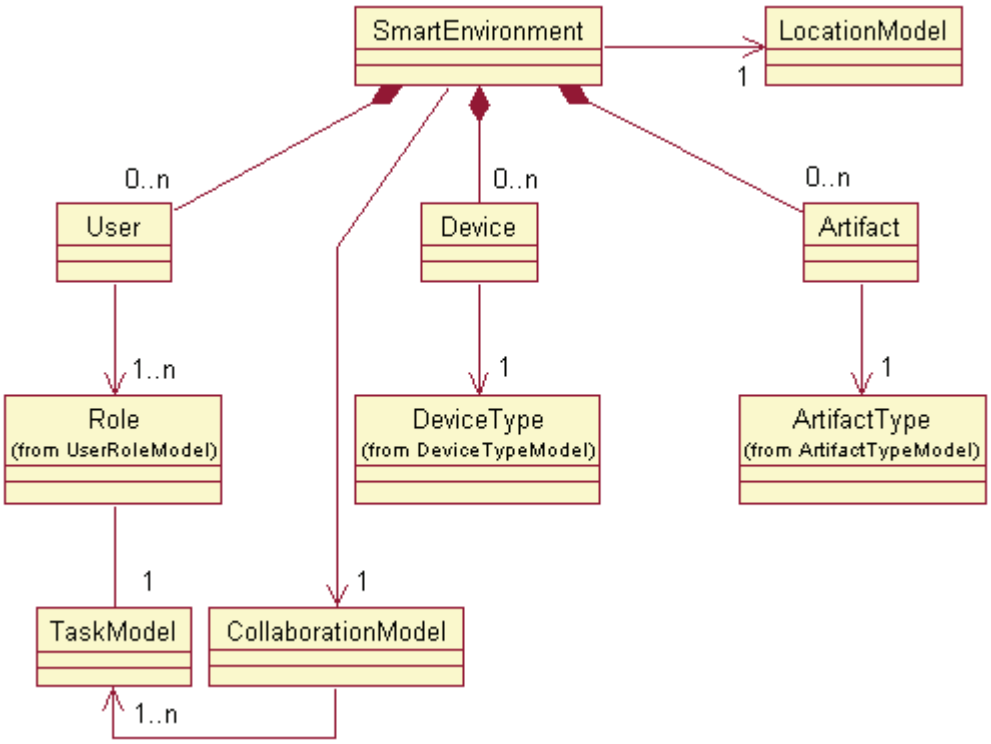

Abbildung 5.1: Metamodell für Smart Environments

5.1.4 Grafische Benutzungsschnittstellen

In einem Smart Environment sind vielfältige Geräte verfügbar, wie Projektoren, Mobiltelefone und PDAs, die teilweise durch die Assistenz des Raumes automatisch gesteuert werden. Darüber hinaus bleiben Funktionen übrig, wie bspw. das Beantworten von Mails, die vom Nutzer persönlich bearbeitet werden. Auch wenn die Assistenz einmal nicht angemessen sein sollte, kann das manuelle Nachjustieren von Geräten sinnvoll sein. Solche Aufgaben werden in CTT-Modellen als Interaktionsaufgaben spezifiziert und können als Ausgangspunkt für die teilautomatisierte Generierung grafischer Benutzungsschnittstellen dienen.

Das weit verbreitete Cameleon-Referenz-Rahmenwerk [CCT+03] verwendet für die Generierung vier Schritte, die folgende Artefakte erzeugen: (1) Aufgaben und Konzepte, (2) abstrakte Benutzungsschnittstelle (AUI), (3) konkrete Benutzungsschnittstelle (CUI), (4) finale Benutzungsschnittstelle (FUI). Konkrete Ansätze für die Entwicklung von grafischen Benutzungsschnittstellen sind bspw. Adept [JJW95], Dynnamo-Aid [CLC04], Teresa [MPS03], Dygimes [CLV+03] und die Verwendung von Dialoggraphen [RFD04, WFR05].

Nachfolgend werden Dialoggraphen verwendet, um auf der Basis der spezifizierten Aufgabenmodelle Dialoge zur Bedienung von Geräten zu beschreiben. Als Beispiel dient ein Projektor aus dem Smart Environment, der normalerweise vom Assistenzsystem automatisch angesteuert wird und für Ausnahmefälle auch eine grafische Steuerung erhalten soll, bspw. um Helligkeit und Kontrast manuell justieren zu können. Dafür sind zunächst die am Projektor durchführbaren Aufgaben in einem Aufgabenmodell zu erfassen. Ein Dialoggraph

gruppiert diese Aufgaben in Dialogsichten. Eine Sicht enthält alle gleichzeitig darzustellenden Aufgaben. Gerichtete Kanten von einer Dialogsicht zur anderen zeigen mögliche Übergänge zu anderen Dialogsichten. Je nachdem welche Aufgabe ausgeführt wurde, kann eine Transition zu einer anderen Sicht oder dem Dialogende stattfinden. Ob eine Aufgabe gerade ausführbar ist oder nicht, richtet sich nach ihrem Zustand im dazugehörigen Aufgabenmodell. Abbildung 5.2 zeigt einen entsprechenden Dialoggraphen im von Reichart entwickelten Dialoggrapheditor [RDF04]. Der Graph zeigt eine vereinfachte Bedienfolge von links nach rechts mit den Dialogsichten „start“, „projection“ und „finish“.

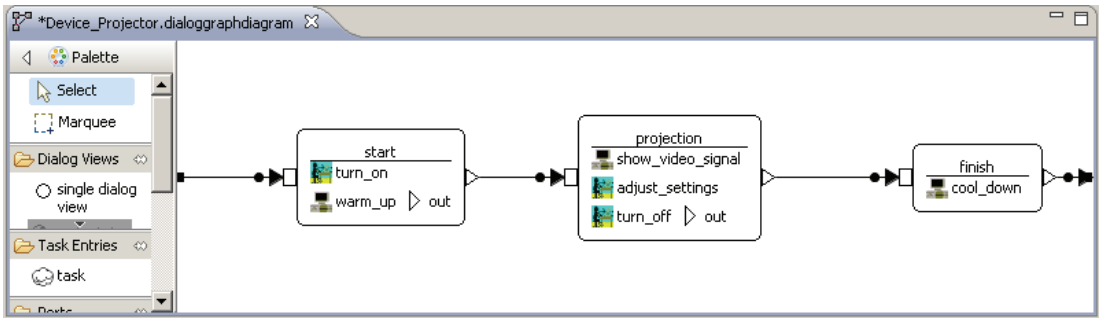

Abbildung 5.2: Dialoggraph für einen Projektor

5.2 Usability-Methode

5.2.1 Kooperativer Aufgabenmodellsimulator

Der bestehende Aufgabenmodellsimulator [RFD04], wie auch in Kapitel 4.2.1 verwendet, animiert die Aufgabenausführung in einem einzelnen Aufgabenmodell und Dialoggraphen. Darauf aufbauend wurde im Rahmen der vorliegenden Arbeit ein kooperativer Simulator entwickelt, der bei der Aufgabenausführung (1) Vorbedingungen und Effekte im Sinne der CTML [WPF08] unterstützt, (2) grafische Benutzungsoberflächen für Geräte animiert [RFD04] und (3) Informationen über die Positionierung von Nutzern und Geräten im Raum visualisiert. Diese drei Aspekte werden in den nachfolgenden Unterkapiteln erläutert.

Der Simulator ist in Abbildung 5.3 dargestellt. Die grafische Benutzungsoberfläche ist aus mehreren Sichten aufgebaut, die beliebig angeordnet werden können. Die abgebildete Konfiguration zeigt in der linken Bildschirmhälfte eine Draufsicht des virtuellen Smart Environments. Die rechte Bildschirmhälfte zeigt im oberen Teil die einzelnen Nutzer mit ihrem Fortschritt bei der Aufgabenausführung, sowie für Geräte, für die ein Dialoggraph spezifiziert wurde, eine Animation der entsprechenden Dialoge. Unten werden die Eigenschaften Detailinformationen zu den Aufgabenmodellinstanzen angezeigt, wie bspw. Vorbedingungen und Effekte der Aufgabeninstanz. Zu sehen ist hier eine Vorbedingung, die für die Aufgabe „Presentation A“ spezifiziert, dass der Nutzer sich in seiner Präsentationszone aufhalten muss, um beginnen zu können.

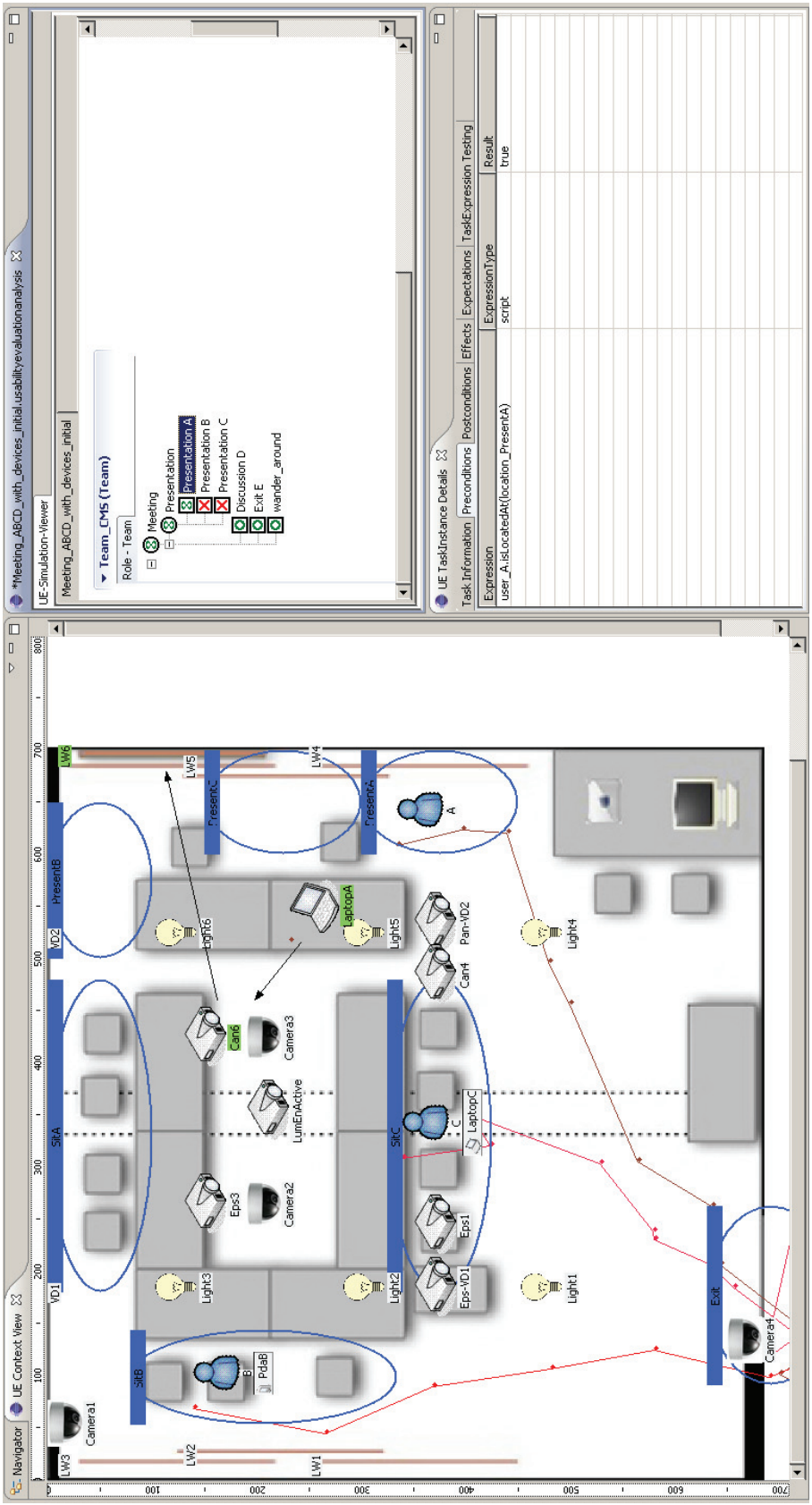

Abbildung 5.3: kooperativer Aufgabenmodellsimulator

5.2.1.1 Aufgabenausführung

Für jeden der anwesenden Nutzer wird für jede Rolle das entsprechende Aufgabenmodell animiert. Ein Beispiel ist in Abbildung 5.4 dargestellt.

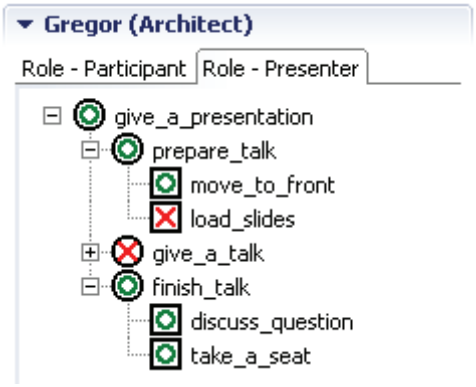

Abbildung 5.4: Beispielinstanz eines einzelnen Nutzers

Dem Metamodell aus Abbildung 5.1 folgend ist die Hierarchie „User“, „Role“ und „TaskModel“ zu erkennen. Die modellierte Entität „Architect“ ist hier mit dem konkreten Nutzer „Gregor“ instantiiert. Dadurch kann in späteren Nutzertests die Teilnahme der verschiedenen Probanden für jeden Test protokolliert werden. Für die beiden Rollen „Participant“ und „Presenter“ sind jeweils als Reiter dargestellt, mit denen man zwischen den Rollen wechseln kann. Innerhalb eines Reiters ist das Aufgabenmodell der Rolle animiert. Gegenüber dem einfachen Aufgabenmodellsimulator aus Kapitel 4.2 hat die Ausführung einer Aufgabe nun nicht nur Auswirkungen lokal in der Modellinstanz, sondern auch auf andere Modellinstanzen.

5.2.1.2 Animation grafischer Benutzungsschnittstellen

Wenn für ein Gerät ein Dialoggraph spezifiziert wurde, wird dieser entsprechend als einfache Benutzungsschnittstelle angezeigt. Dazu wurde der Dialoggraphsimulator von Reichart et al. [RDF04] eingebunden, der Dialoggraphen in einfache Benutzungsschnittstellen transformiert. Dialogsichten werden dabei als Fenster dargestellt und Aufgaben als Schaltflächen. Ein Mausklick auf eine Schaltfläche löst die Ausführung der entsprechenden Aufgabe aus. Erfolgt daraufhin eine Transition im Dialoggraphen öffnet sich ein neues Fenster mit der nächsten Dialogsicht. Um die Oberfläche realistischer erscheinen zu lassen, bzw. den Schritt hin zu einer funktionalen Oberfläche zu erlauben, können Schaltflächen durch komplexere interaktive Elemente ersetzt werden. Wie der Dialoggraph des Projektors aus Abbildung 5.2 animiert werden kann, ist in Abbildung 5.5 dargestellt.

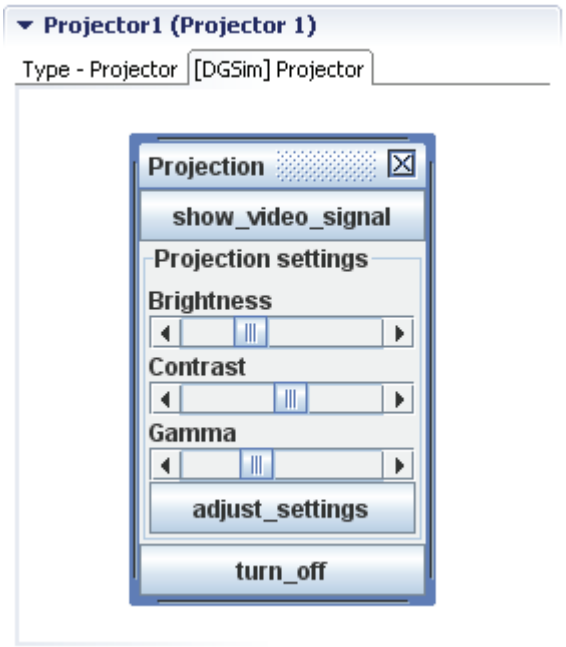

Abbildung 5.5: Animation des Dialoggraphen des Projektors

Die Dialogsicht „Projection“ (Abb.5.2) besitzt im Dialoggraphen die drei Aufgaben „show_video_signal“, „adjust_settings“ und „turn_off“, die jeweils als Schaltfläche dargestellt sind. Um das Einfügen weiterer Steuerelemente zu illustrieren, wurden Schieberegler zur Einstellung von Helligkeit, Kontrast und Gammawert eingefügt.

Bei der Interaktion mit der grafischen Benutzungsschnittstelle wird das entsprechende Aufgabenmodell aktualisiert.

5.2.1.3 Visualisierung ortsbezogener Daten

Der in Abbildung 5.3 dargestellte kooperative Aufgabenmodellsimulator zeigt in der linken Hälfte eine Draufsicht auf das virtuelle Smart Environment. Visualisiert werden Nutzer, Geräte (bspw. Laptops, Projektoren, Lampen) und weitere Objekte (bspw. Tische, Stühle). Diese Entitäten sind zur Laufzeit gekennzeichnet über ihre Position, Breite, Länge und Typ. Sie können grundsätzlich bewegt und von Nutzern getragen werden, was durch entsprechende Änderung der Eigenschaften eingeschränkt werden kann. So können bspw. an der Decke fest montierte Projektoren als nicht beweglich gekennzeichnet werden.

Orte im Raum, die eine besondere Bedeutung für die Ausführung der definierten Aufgaben haben, sind als blau umrandete Zonen hervorgehoben. Diese können innerhalb von kooperativen Ausdrücken verwendet werden. Bspw. kann als Vorbedingung für den Start der Aufgabe „Präsentation halten“ („give_a_talk“) als Aufenthaltsort eine „Präsentationszone“ vorne vor dem Publikum festgelegt werden.

Durch Interaktion mit der Maus können Nutzer, Geräte und Gegenstände frei bewegt werden, solange keine definierte Eigenschaft dies einschränkt. Zieht man ein Gerät oder Gegenstand auf einen Nutzer, wird dies als Aufnehmen der Entität in die Hand aufgefasst.

Die Entität wird beim Nutzer als Liste der getragenen Objekte dargestellt und bei jeder Bewegung des Nutzers entsprechend mitbewegt.

Zu allen dargestellten Entitäten lassen sich die aktuellen Eigenschaften anzeigen und bearbeiten. Es lassen sich beliebig viele weitere Eigenschaften in das Datenmodell aufnehmen und visualisieren. Da das Labor der Universität Rostock umfassend mit Projektoren und Leinwänden ausgestattet ist, wurde zusätzlich die Visualisierung von Videoverbindungen zwischen Geräten in das virtuelle Smart Environment einbezogen. So kann bspw. der Laptop des Vortragenden, der die Präsentationsfolien enthält, mit über den Videoausgang mit einem Projektor verbunden werden und drehbare Projektoren können in Richtung einer bestimmten Leinwand gedreht werden, um schließlich die Präsentation anzuzeigen.

Tabelle 5.1 gibt einen Überblick, wie die grafischen Elemente zu interpretieren sind.

Grafisches Element	Bedeutung
Position der Symbole	Position der Entitäten
Art der Symbole	Typ der Entitäten (Nutzer, Geräte, Objekte)
Eigenschaftsanzeige	Attribute der Entitäten (bspw. Alter, Beruf, körperliche Einschränkungen)
Gerät / Objekt einem Nutzer angefügt	Tragen von Geräten / Objekten
Zone	Lage eines Ortes
Gerätename farbig hinterlegt	Aktivierungszustand (an / aus)
Pfeilverbindungen zwischen Geräten	Videoverbindung zwischen Geräten aufgebaut, bzw. Projektor in Richtung einer Leinwand gedreht

Tabelle 5.1: Bedeutung der grafischen Elemente im virtuellen Smart Environment

5.2.2 Vorgehen bei der Evaluation

In der aktuellen Phase der Entwicklung ist das Smart Environment modelliert als eine Menge von Nutzern, Geräten und Gegenständen, die räumlich angeordnet und in die Aufgabenausführung involviert sind. Eine Implementation als physische Umgebung liegt noch nicht vor. Daher können noch keine Nutzerstudien mit einem realen System durchgeführt werden. Stattdessen stehen im Rahmen der Aufgabenanalyse Methoden des Durchsprechens („talkthrough“) und Durchlaufens („walkthrough“) zur Verfügung [KA92]. Betrachtet nach dem Grad der Detailliertheit der Repräsentation von Aufgaben und Nutzerschnittstellen spannt sich ein ganzes Spektrum von Methoden auf. Am unteren Ende der Detailliertheit befindet sich das Durchsprechen einer ersten Beschreibung der spezifizierten Aufgaben und Benutzungsschnittstelle. Am oberen Ende befindet sich das Durchlaufen einer realen Umgebung in Echtzeit. Dabei lösen Nutzer Aufgaben in realer Umgebung, indem sie auf die jeweils verwendeten Geräte zeigen und erläutern, was sie in der konkreten Situation tun würden. Das Kontinuum zwischen Durchsprechen und Durchlaufen besitzt Zwischenstufen, bspw. bei der Benutzung von Oberflächenprototypen, Simulationen oder realen Umgebungen.

Der hier verwendete Ansatz ist eine solche Zwischenstufe. Der beschriebene kooperative Aufgabensimulator animiert das modellierte Smart Environment und unterstützt damit das Durchlaufen des virtuellen Smart Environments. Für die Durchführung der konkreten Evaluation gibt es mehrere Möglichkeiten. Im Rahmen einer Expertenevaluation können wahlweise in der Anforderungsanalyse festgelegte Szenarien durchlaufen werden oder aber rein explorativ ausgewählte Entitäten innerhalb der Modelle evaluiert werden. Zudem kann mit Nutzern ein Durchlaufen der Teamsituation im virtuellen Smart Environment erfolgen. Die Möglichkeiten werden nachfolgend vorgestellt.

5.2.2.1 Expertenevaluation nach vorgegebenem Szenario

Bei der Durchführung einer Expertenevaluation versetzen sich Fachexperten in die Lage von Nutzern und vollziehen Nutzerverhalten im virtuellen Smart Environment nach. Dabei werden die erstellten Modelle gegen die in der Anforderungsanalyse ermittelten Funktionalitäten validiert.

Je nach Umfang der Evaluation nehmen ein oder mehrere Experten teil. Von den in der Anforderungsanalyse erstellten Szenarien werden einige ausgewählt. Der Experte macht sich zunächst mit der vorgegebenen Situation vertraut: den beteiligten Personas, Kontextbedingungen und dem geplanten Ablauf der Interaktion. Anschließend wird das virtuelle Smart Environment in die Ausgangssituation versetzt. Alle Nutzer, Geräte und Gegenstände werden entsprechend platziert. Im Beispiel einer Besprechung werden alle Nutzer außerhalb des Raumes positioniert und mitgebrachte persönliche Geräte an die entsprechenden Nutzer geheftet. Im Besprechungsraum werden alle Möbel und weiteren Gegenstände verteilt, so dass die gewünschte Ausgangskonfiguration entsteht. Der Experte bewegt nun alle Nutzer in der virtuellen Umgebung so, wie sie sich auch in der realen Umgebung verhalten würden. Wesentlicher Unterschied ist die Abstraktion der realen Eigenschaften der Entitäten und deren Verhalten, wie bspw. Muskelanspannung beim Aufnehmen von Geräten oder 3-dimensionale räumliche Bewegungen, auf eine vereinfachte Darstellung in der virtuellen Umgebung, wie bspw. Aufnehmen eines Gerätes per „Drag & Drop“ in Simulationsumgebung oder 2-dimensionale Bewegungen. Für die Validität dieser Vereinfachung spricht zum einen, dass manches irrelevant ist für die Evaluation. So soll im Beispiel nicht die Bewegung von Muskeln im Detail untersucht werden, sondern nur die Interaktion mit Gegenständen im Sinne von Aufnehmen, Funktionalität auswählen und Ablegen. Zum anderen führt die Vereinfachung der virtuellen Umgebung dazu, dass die Bedienung einfach bleibt und nicht vom Wesentlichen ablenkt.

Während der Evaluation achtet der Experte darauf, ob bei der simulierten Interaktion von Nutzern die entwickelten Modelle den Anforderungen gemäß reagieren.

Eine Nutzerstudie in Kapitel 7.1 berichtet über Erfahrungen mit der Methode, um zu zeigen, wie gut Usability-Probleme in der virtuellen Umgebung gefunden werden können.

5.2.2.2 Explorative Expertenevaluation

Alternativ zur Abarbeitung eines konkreten Szenarios kann die Evaluation explorativ durchgeführt werden. Ein eingeladener Fachexperte beginnt mit einer konkreten Situation, so wie bei der Evaluation anhand eines Szenarios, animiert das Verhalten der anwesenden Nutzer im Gegensatz dazu aber völlig frei. Getrieben durch die eigene Intuition vollzieht der Fachexperte das aus seiner Sicht als typisch empfundene Verhalten nach. Der Schwerpunkt liegt dabei auf dem Test der Vorbedingungen von Aufgaben. Dazu werden Situationen im virtuellen Smart Environment nachgestellt, bei dem sich bspw. Nutzer in bestimmten Zonen im Raum aufhalten, Geräte bei sich haben und diese Geräte sich in einem gewissen Zustand befinden. Der Experte vergleicht, welche Aufgaben entsprechend der definierten Anforderungen aktivierbar sein müssten und welche tatsächlich aktivierbar sind. Außerdem ist zu kontrollieren, ob die erwarteten Effekte von Aufgaben korrekt ausgelöst werden.

5.2.2.3 Nutzerevaluation

Eine weitere Methode ist der Einbezug von Nutzern im Rahmen eines „team walkthrough“ [KA92 (S.163)]. Zunächst wird ein durchzuspielendes Szenario festgelegt. Alle involvierten Akteure des virtuellen Smart Environments werden durch jeweils einen der anwesenden Nutzer gespielt. Jeder Nutzer bekommt eine Beschreibung seines Akteurs und einen Überblick über das gesamte Szenario. Die Nutzer setzen sich so, dass sie das virtuelle Smart Environment am Bildschirm sehen können und achten dabei besonders auf die Darstellung des eigenen Akteurs. Wenn alle Fragen der Nutzer geklärt sind, übernimmt ein Experte die Aufgabe, die Animation zu steuern. Jeder Nutzer sagt, was sein virtueller Akteur tun soll, bspw.: „Ich gehe nun nach vorne, um die Präsentation zu beginnen“. Nachdem der Akteur diesen Ort erreicht hat, steuern weitere Anweisungen der Testnutzer die Interaktionen. Lassen sich bestimmte Aufgaben nicht ausführen, wird dies vom Experten annotiert. In Fällen, bei denen der Experte die Ursache des Problems erkennt, erläutert er kurz, woran es liegt und übergeht das Problem manuell, um die Suche nach weiteren Problemen zu ermöglichen. Jedes Szenario sollte mehrfach mit anderen Nutzergruppen durchgespielt werden.

Analog zur Expertenevaluation kann auch die Nutzerevaluation explorativ ohne vorgegebenes Szenario erfolgen.

5.3 Verbesserung der Usability

Das Ziel der Evaluation in der Designphase ist die Identifikation von Abweichungen zwischen den definierten Anforderungen und den daraus abgeleiteten Designmodellen. Bei den kooperativen Aufgabenmodellen liegt der Schwerpunkt auf der Evaluation der Kooperation zwischen den einzelnen Nutzern und der Abhängigkeit der Aufgabenausführung vom Kontext. Daher werden Vorbedingungen und Effekte von Aufgaben untersucht. Dabei werden insbesondere die festgelegten Zonen im Ortsmodell betrachtet.

Außerdem zeigen sich bei der Animation der Modelle für die einzelnen Nutzer das Pensum an durchzuführenden Aufgaben und die Anzahl an Rollenwechseln als Indikatoren für eine hohe Beanspruchung. Weiterhin deuten umfangreiche Vorarbeiten anderer Nutzer für eine bestimmte Tätigkeit auf Wartezeiten in der Aufgabenbearbeitung hin.

Wenn Unstimmigkeiten zwischen definierten Anforderungen und Designmodellen aufgedeckt werden, stehen Editoren bereit, um die Modelle der Realität anzupassen.

Der Arbeitsablauf sieht dabei typischerweise so aus, dass zunächst die zu evaluierenden Modelle im kooperativen Aufgabensimulator animiert werden. Dabei wird das Verhalten der Modellinstanzen beobachtet, ob Vorbedingungen und Effekte in erwarteter Weise reagieren. So kann bspw. eine Aufgabe, die man gerade ausführen möchte, nicht ausführbar sein, weil eine Vorbedingung falsch spezifiziert wurde. Daraufhin kann man die letzten Interaktionsschritte per „undo“ rückgängig machen bis zum letzten konsistenten Zustand. Anschließend korrigiert man die entsprechenden Vorbedingungen und Effekte und wiederholt die gleiche Sequenz von Interaktionschritten, um das Verhalten der verbesserten Modelle zu überprüfen. Die im Ortsmodell definierten Zonen können ebenfalls auf diese Weise angepasst werden.

5.4 Zusammenfassung

Basierend auf den Dokumenten der Anforderungsanalyse wird das Design durchgeführt. Verschiedene Modellierungsmethoden zur Beschreibung kooperativer Arbeit wurden diskutiert und CTML für die weitere Arbeit ausgewählt.

Das entwickelte virtuelle Smart Environment erlaubt die Animation von kooperativen Aufgabenmodellen, um diese durch interaktives Durchlaufen („Walkthrough“) zu evaluieren. Eine grafische Ansicht des Raumes dient dabei zur Visualisierung der Nutzer, Geräte und Gegenstände im Raum.

Die spezifizierten Aufgabenmodelle können zur modellbasierten Entwicklung von grafischen Benutzungsschnittstellen für Geräte, wie PDAs oder Mobiltelefone, verwendet werden. Prototypen dieser Schnittstellen werden vom virtuellen Smart Environment dargestellt und lassen sich so in die Evaluation einbeziehen.

Kapitel 6

Implementation

Der bisher dargestellte modellbasierte Entwicklungsprozess hat Aufgabenmodelle zur Beschreibung von Nutzerverhalten verwendet und wird diese nun in der Implementationsphase in statistische Modelle überführen, um so die Intentionserkennung in Smart Environments zu unterstützen.

Die Evaluation läuft dabei in zwei Phasen ab: (1) Solange der Raum noch nicht fertig gestellt ist werden mit Hilfe des virtuellen Smart Environments einzelne bereits implementierte Komponenten evaluiert. Die notwendigen Kontextinformationen werden währenddessen als künstliche Sensordaten generiert. (2) Nachdem der Raum fertig gestellt ist, können Nutzerstudien in der realen Umgebung durchgeführt werden. Das virtuelle Smart Environment sorgt hierbei für die Visualisierung der Nutzerinteraktionen.

6.1 Entwicklungsmethode

Zunächst wird das Prinzip der zielbasierten Interaktion vorgestellt, um die Funktionsweise des Assistenzsystems zu erläutern. Anschließend wird die Transformation von Aufgabenmodellen zu statistischen Modellen der Intentionserkennung beschrieben

6.1.1 Zielbasierte Interaktion

Bevor die Entwicklungsmethode vorgestellt wird, soll ein Ausblick auf die geplante Funktionsweise des Smart Environments gegeben werden, auf dem die Entwicklung und spätere Evaluation aufbauen.

Die Steuerung von Geräten erfolgt im Alltag derzeit funktionsbasiert über deren direkte Bedienung. Jedes Gerät bietet dafür eine Menge von Funktionen. So sind bspw. an einem Projektor typischerweise „an“, „aus“ und diverse Einstellungen für die Bilddarstellung verfügbar. Im konkreten Beispiel einer Besprechung hat ein Vortragender vor Beginn des Vortrages mehrere Vorbereitungen zu treffen. Zunächst nimmt er seinen Laptop mit seiner Präsentation und geht nach vorne. Dort ist der eigene Laptop mit der Rauminfrastruktur zu

verbinden, ein Projektor und eine Leinwand sind auszuwählen und in den betriebsbereiten Zustand zu bringen.

Eine Alternative ist die zielbasierte Interaktion. Hierbei muss der Nutzer die einzelnen Geräte und deren Funktionen im Raum nicht detailliert kennen. Es genügt, ein Ziel auszudrücken. Dem Beispiel folgend lautet dies: „Ich habe Präsentationsfolien auf meinem persönlichen Gerät und möchte diese für alle Anwesenden sichtbar dargestellt haben.“ Das im Raum installierte Assistenzsystem ermittelt darauf hin, welche Funktionalität dafür von welchen Geräten benötigt wird und steuert diese Geräte mit der entsprechenden Parametrisierung an. Das Prinzip zielbasierter Interaktion wird in Abbildung 6.1 gezeigt.

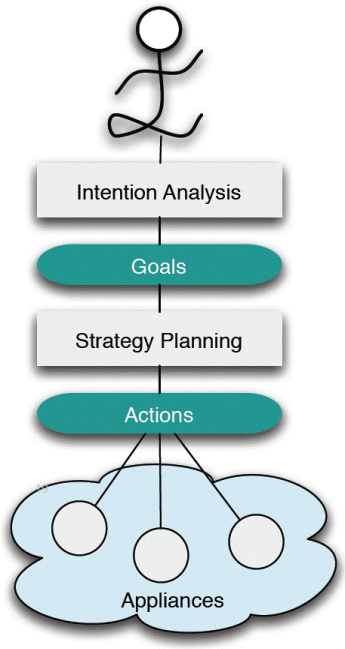

Abbildung 6.1: Prinzip der zielbasierten Interaktion [AE06 (S.331)]

Nutzerinteraktionen und deren Ausführungskontext werden durch Sensoren im Raum erfasst. Sensorwerte können dabei bspw. Temperatur, Positionen von Nutzern und Geräten, berührte Gegenstände und Bewegungen von menschlichen Gelenken umfassen. Diese Daten dienen als Eingabe für die Intentionsanalyse, die darauf basierend versucht, die Ziele der Nutzer vorherzusagen. Die nachfolgende Strategieplanung ermittelt die zur Erreichung dieser Ziele notwendigen Aktionssequenzen. Dazu sind die verfügbaren Geräte und deren Funktionalitäten zu bestimmen, die relevanten auszuwählen und eine zielführende Sequenz von Aktionen zu synthetisieren. Die entsprechenden Geräte im Raum werden angesteuert.

Abbildung 6.1 stellt diesen Zusammenhang als linearen Informationsfluss von oben nach unten dar. Die Abbildung lässt sich durch eine Verbindung von „Appliances“ zurück zum Nutzer zu einem Kreislauf ergänzen. Denn resultierend aus der Durchführung der Strategie können sich zum einen direkt Kontextbedingungen ändern, was die eingehenden Sensorwerte der Intentionserkennung ändert. Zum anderen kann durch die Aktionen der Geräte das Verhalten der Nutzer beeinflusst werden. Ist es bspw. im Raum kalt, kann das Einsetzen der

Heizung dazu führen, dass der Vortragende seine Strickjacke auszieht und zum Kleiderständer bringt, wodurch er seine Position verändert.

6.1.2 Transformation von Aufgabenmodellen zu statistischen Modellen

Aus den spezifizierten Aufgabenmodellen wird die Software zur Intentionserkennung generiert. Abbildung 6.2 zeigt in Anlehnung an [BPK+09] den Prozess.

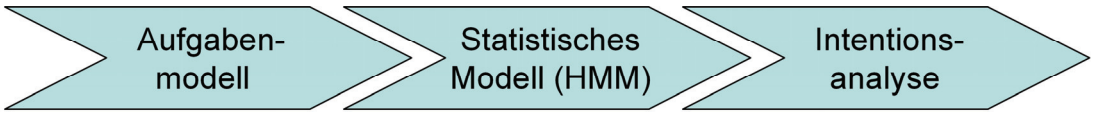

Abbildung 6.2: Transformation vom Aufgabenmodell zur Intentionsanalyse

Für die Transformation eines Aufgabenmodells zu einem statistischen Modell sind zunächst an die Aufgaben Prioritäten zu annotieren. Eine Priorität macht das Zusatzwissen eines Experten explizit, wie stark ein Nutzer eine Aufgabe gegenüber anderen präferiert, wenn es die entsprechende Alternative zur Wahl gibt. Das Beispiel in Abbildung 6.3 zeigt das Prinzip. Drei Vortragende A, B und C präsentieren in beliebiger Reihenfolge und abschließend findet eine Diskussion statt. Die Prioritäten sind relativ zueinander und ergänzen sich hier der Übersichtlichkeit halber zu 100, könnten aber auch beliebige positive Zahlen enthalten. Die Besprechung sollte im konkreten Fall laut Agenda mit einer Wahrscheinlichkeit von 90% mit A beginnen, zu 9% mit B und zu 1% mit C.

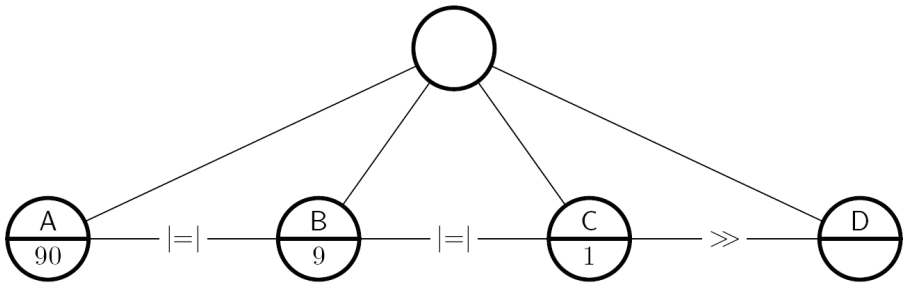

Abbildung 6.3: Aufgabenmodell mit Prioritäten annotiert [GFF+07]

Ein HMM („Hidden Markov Model“) [Rab89] ist ein Quintupel $\lambda = (S, V, A, B, \pi)$ mit:

- $S = \{s_1,...,s_N\}$ als Menge aller (verdeckten) Zustände
- $V = \{v_1,...,v_M\}$ als Menge aller für diese Zustände beobachtbaren Merkmale
- $A = \{a_{ij}\}$ als Zustandsübergangsmatrix, wobei a_{ij} die Wahrscheinlichkeit angibt, dass von Zustand s_i in Zustand s_j gewechselt wird
- $B = \{b_j(k)\}$ als Wahrscheinlichkeit der Beobachtung eines Merkmals in Zustand j
- $\pi = \{\pi_i\}$ als Wahrscheinlichkeitsverteilung des Initialzustandes

Der in Giersich, Forbrig et al. [GFF+07] vorgestellte Algorithmus erlaubt das Generieren von S, A und π für das HMM, während die übrigen Parameter manuell ergänzt werden. An einer Implementation eines Generators für diese Parameter wird derzeit an der Universität Rostock gearbeitet.

Der Algorithmus aus [GFF+07] ermittelt alle möglichen Durchläufe durch das Aufgabenmodell und deren Wahrscheinlichkeiten auf Basis der annotierten Prioritäten. Das Ergebnis ist beispielhaft in Abbildung 6.4 dargestellt. Der gerichtete Graph beschreibt eine Menge von Pfaden vom Startzustand (initialer Zustand der Aufgabenausführung) zum Zielzustand (Aufgabenmodell erfolgreich durchlaufen). Jeder Knoten enthält die Menge der in diesem Zustand bereits ausgeführten Aufgaben. Jede Kante stellt die Ausführung einer bestimmten Aufgabe als Übergang von einem Zustand zum nächsten dar. Die Beschriftung an den Kanten enthält die Wahrscheinlichkeit für diesen Übergang. Die Summe aller Ausgangswahrscheinlichkeiten beträgt 1,0 (also 100%). Weitere Entitäten des HMMs, wie bspw. die beobachteten Merkmale V oder die Beobachtungswahrscheinlichkeiten aus B sind zur besseren Übersicht nicht dargestellt.

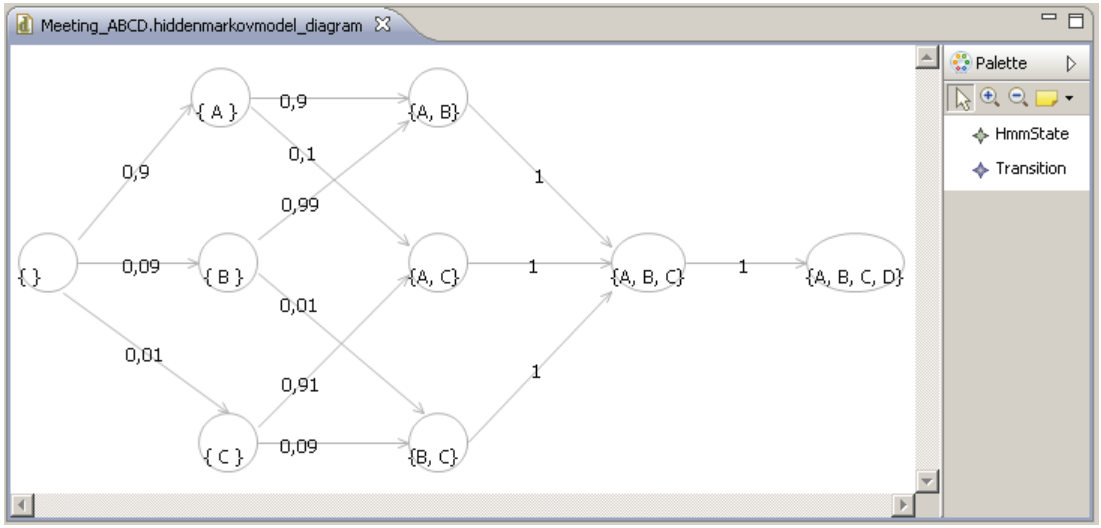

Abbildung 6.4: Transitionsmatrix eines generierten HMMs

Das automatisch generierte HMM kann manuell nachbearbeitet werden, um eine Feinjustierung der Wahrscheinlichkeiten vorzunehmen. Zudem können an die Zustände Beobachtungen annotiert werden, wie bspw. Orte, verwendete Geräte und andere Kontextinformationen (als beobachtete Merkmale V). Anschließend erfolgt die Transformation des HMM in den Quelltext einer konkreten Ausführungsumgebung für HMMs, die das HMM zur Laufzeit im Smart Environment für die Intentionserkennung betreibt.

6.2 Usability-Methode

Für die Usability-Evaluation während der Implementationsphase werden nachfolgend zwei Methoden vorgestellt: (1) Die erste ist eine Expertenevaluation einzelner fertiger Softwarekomponenten zu einem frühen Zeitpunkt. (2) Die zweite unterstützt den Nutzertest nachdem das physische Smart Environment komplett eingerichtet ist.

6.2.1 Expertenevaluation von Teilkomponenten

Einzelne implementierte Softwarekomponenten, wie bspw. die Intentionserkennung oder die Strategieplanung, sollen möglichst frühzeitig evaluiert werden, um die iterative Verbesserung zu erlauben. Da diese Komponenten vom Kontext abhängig sind, werden Kontextinformationen für die Evaluation benötigt. Später im alltäglichen Betrieb des Smart Environments werden diese Daten von Sensoren geliefert. Wenn während der Entwicklung noch keine physische Umgebung vorhanden ist, scheidet dieser Weg jedoch aus. Daher wurde die bereits vorgestellte virtuelle Umgebung so erweitert, dass künstliche Sensordaten generiert werden können.

Angestrebt ist dabei beides: (a) einzelne bereits implementierte Komponenten frühzeitig zu testen und (b) später, wenn ein physisches Smart Environment aufgebaut ist, Fehlerursachen einzugrenzen. Dazu können einzelne Komponenten isoliert mit frei wählbaren Sensordaten als Eingabe versorgt werden, um die Reaktion darauf zu testen.

Im Rahmen einer Expertenevaluation werden verschiedene Szenarien durchlaufen. Analog zur Evaluation während der Designphase wird ein virtuelles Smart Environment genutzt, in dem Interaktionen der physischen Umgebung animiert werden.

Der Ablauf der Evaluation besteht aus drei Schritten: (1) das Design von Szenarien innerhalb des virtuellen Smart Environments als Sequenzen von manuell erstellten Nutzerinteraktionen, (2) das automatische Generieren von künstlichen Sensordaten auf Basis der Szenarien, wobei Fehler eingeführt werden und (3) das Einspielen der Daten in die zu testenden Komponenten mit Analyse der Ausgaben.

6.2.1.1 Design von Szenarien im virtuellen Smart Environment

Für das Design von Szenarien bestehen zwei Möglichkeiten: das Verhalten aller Nutzer kann parallel animiert werden oder der Experte animiert alle Nutzer nacheinander, um sie danach zu einem Datensatz zusammenzusetzen.

Beim parallelen Animieren aller Nutzer werden Interaktionen der einzelnen Nutzer abwechselnd und ineinander verschränkt animiert. Dies kann in verlangsamter Geschwindigkeit geschehen, um dem Experten mehr Zeit zur Erstellung einzelner Details zu geben. Das Abspielen der Daten erfolgt schneller, also in normaler Geschwindigkeit. Das Erstellen der Daten kann relativ schnell gehen, ist aber beschränkt im Bezug auf die parallele Ausführung von Interaktionen verschiedener Nutzer, da durch einen einzelnen Experten jeweils nur ein Interaktionsschritt zur Zeit animiert werden kann.

Die andere Möglichkeit besteht darin, jeweils einen Nutzer zurzeit zu animieren. Entsprechend des geplanten Szenarios wird zunächst nur das Verhalten eines einzelnen Nutzers für das komplette Szenario animiert. Im zweiten Durchlauf wird das Verhalten dieses Nutzers zur Orientierung abgespielt und parallel dazu ein weiterer Nutzer animiert. Iterativ werden alle Nutzer einbezogen. Auf diese Weise können Interaktionen von Nutzern auch durch nur einen Experten vollständig parallel animiert werden.

Als Quelle für die durchwanderten Szenarien gibt es zwei Möglichkeiten. Zum einen können die in der Anforderungsanalyse ermittelten Szenarien verwendet werden. Zum anderen können aufgezeichnete Interaktionen aus einem bestehenden Raum genutzt werden. Dies ist bspw. dann möglich, wenn ein bereits bestehender nicht instrumentierter Raum zum Smart Environment ausgebaut werden soll. Wenn das Einverständnis der Beteiligten vorliegt, können laufende Besprechungen als Video aufgezeichnet werden. Diese werden verlangsamt abgespielt, während wie oben beschrieben das Verhalten abkodiert wird. Diese realen Verhaltensdaten können in die Intentionserkennung eingespielt werden, um die automatisch erkannten Intentionen mit den tatsächlich beobachteten zu vergleichen.

6.2.1.2 Generieren von Sensordaten

Nachdem Nutzerinteraktionen animiert wurden, sollen daraus Sensordaten generiert werden. Tabelle 6.1 gibt einen Überblick über die animierten Interaktionen im virtuellen Smart Environment und die daraus generierten Sensordaten.

Interaktion im virtuellen Smart Environment	Generiertes Sensordatum für ein reales Smart Environment
Bewegen einer Entität	Ortsinformation für Nutzer, Gerät oder Gegenstand (UbiSense)
Gerät oder Objekt einem Nutzer anfügen	Erkennung eines RFID-Transponders durch ein Lesegerät (bspw. Schlaufe am Arm eines Nutzers)
Zustand eines Gerätes ändern (bspw. persönliches Gerät ein- oder ausschalten)	Gerätestatusmeldung
Zustand eines Gegenstandes ändern (bspw. ein Glas füllen, Kappe eines Stiftes abnehmen)	Attribute der Entitäten (Domänenmodell)
Beginn / Beendigung einer Aufgabe der aktuellen Rolle	Ausgeführte Aufgabe (als Ausgabe der Intentionserkennung)
Beginn / Beendigung einer Aufgabe in einer neuen Rolle (Rollenwechsel)	Wechsel der Rolle und Ausführung der Aufgabe (als Ausgabe der Intentionserkennung)

Tabelle 6.1: Generierung künstlicher Sensordaten aus Interaktionen im virtuellen Smart Environment

Die oberen vier Zeilen der Tabelle 6.1 enthalten Daten, die direkt physisch über Sensoren ermittelt werden können, wie bspw. die Positionsänderung von Nutzern über Positionsbestimmungssysteme oder das Aktivieren von persönlichen Geräten über ein Überwachen der Kommunikation der Rauminfrastruktur. Je nach Sensor können dabei verschiedenartige Fehler bei der Erkennung auftreten. Daher muss ein Experte für jeden Sensor separat die Fehlerverteilung und die Abtastrate festlegen.

Bei den durchgeführten Experimenten mit dem virtuellen Smart Environment wurden analog zum „Smart Environment“-Labor der Universität Rostock das Lokalisierungssystem UbiSense [Ubi09] und RFID-Lesegeräte berücksichtigt. Die Fehlercharakteristik des UbiSense-Systems ist von vielen Faktoren abhängig, bspw. der exakten Kalibrierung des Gesamtsystems, der Anzahl der vorhandenen UbiSense-Transponder, dem Wetter und den Materialien vorhandener Gegenstände. Da sich diese Faktoren teilweise über die Zeit hinweg ändern, wurden die Fehler auf Basis von Erfahrungen geschätzt. Als Näherung wurde eine Normalverteilung mit einer Standardabweichung von 1 m gewählt und eine Abtastrate von 3 Hz. Diese Werte lassen sich frei konfigurieren. Bei der Erkennung von RFID können Wahrscheinlichkeiten für die Fehler erster und zweiter Art angegeben werden. Fehler erster Art („falsch positiv“) liegen bei RFID vor, wenn ein nicht verfügbarer RFID-Transponder trotzdem erkannt wird. Fehler zweiter Art („falsch negativ“) liegen vor, wenn sich ein RFID-Transponder in Reichweite des Lesegerätes befindet, aber dennoch nicht erkannt wird. Bei bisherigen Experimenten traten Fehler erster Art nicht auf.

Reale Sensordaten mit hohen Abtastraten enthalten viele Tausend oder Hunderttausend Datensätze. Diese mit genauso vielen Mausklicks in einer virtuellen Umgebung nachzuempfinden wäre zeitaufwendig. Daher sind die vom Experten durchlaufenen Interaktionen als Eckpunkte der Interaktion zu verstehen, die die charakteristischen Teile der Handlung beinhaltet. Für RFID bspw. genügt die Kennzeichnung von Aufnehmen und Ablegen eines Gegenstandes. Die dazwischen liegenden Sensordaten werden an Hand der Abtastrate und Fehlercharakteristik generiert. Bei Lokalisationsdaten werden in der virtuellen Umgebung Wegpunkte markiert, zwischen denen eine einigermaßen gleichmäßige Geschwindigkeit und Bewegungsrichtung herrscht. Die Lokalisationsdaten dazwischen werden ebenfalls an Hand der Abtastrate und Fehlercharakteristik als gleichmäßige Bewegung generiert.

Die unteren beiden Zeilen der Tabelle 6.1 enthalten Daten, die nicht direkt über Sensoren ermittelt werden können, sondern durch eine Intentionserkennung aus Sensordaten abgeleitet werden. Wenn eine Intentionserkennung noch nicht implementiert ist, können diese Daten alternativ manuell als Eingabe für die Strategieplanung erzeugt werden, um deren Funktionalität zu evaluieren.

6.2.1.3 Evaluation mit generierten Sensordaten

Mit Hilfe der generierten Sensordaten können verschiedene Softwarekomponenten des Smart Environments getestet werden. Dies erfolgt entlang der Datenverarbeitung von Sensordaten über Intentionserkennung, Strategieplanung bis hin zur Ansteuerung von Geräten (Abbildung 6.1).

Die Intentionserkennung, wie sie in [BPK+09] beschrieben wird, erlaubt dem Designer umfangreiches Vorwissen über charakteristische Kontextbedingungen der Aufgabenausführung einzubringen. Bspw. „Ein Nutzer, der die Aufgabe des Präsentierens ausführt, kann entweder in der Zone vor der Tafel stehen mit einem Stift oder an der Leinwand mit einer Präsentationsfernbedienung“. Bei dem Einsatz vieler verschiedener Sensoren in Kombination und einem großen Zustandsraum kann die Modellierung unübersichtlich werden. Zur Evaluation werden generierte Sensordaten entsprechend der zu testenden Szenarien an die Intentionserkennung übermittelt. Die Ausgaben werden im animierten Aufgabenmodell dargestellt. Ein Vergleich zwischen der nach Testszenario auszuführenden und der erkannten Aufgabe zeigt, wo die Erkennung zu verbessern ist. Das HMM und die annotierten Beobachtungen sind zu überprüfen.

Die Strategieplanung verwendet die Ergebnisse der Intentionserkennung weiter und leitet daraus eine Konfiguration der Geräte ab, die die Nutzerziele möglichst gut unterstützen soll. Manche Ziele können auch durch verschiedene Aktionen erreicht werden. Sollen bspw. Folien für eine Präsentation auf eine bestimmte Leinwand projiziert werden, können mehrere drehbare Projektoren in Frage kommen, die dieses Ziel erreichen. Auch hier sollen Testszenarien zur Evaluation verwendet werden. Dazu werden im Aufgabenmodellsimulator Sequenzen von Interaktionen durchlaufen und entsprechend als Eingabe an die Strategieplanung übergeben. Diese ermittelt eine geeignete Konfiguration, die im virtuellen Smart Environment dargestellt wird. Nun kann evaluiert werden, ob bspw. das mobile Gerät des Vortragenden mit dem entsprechenden Projektor verbunden ist und dieser auf die richtige Leinwand zeigt.

6.2.2 Nutzertest zur Evaluation des kompletten Systems

Nachdem das komplette Smart Environment als physische Umgebung eingerichtet wurde, können Nutzertests zur Evaluation aller Aspekte durchgeführt werden. Die vorausgehenden Evaluationen in früheren Entwicklungsphasen konnten jeweils nur die vorliegenden Artefakte evaluieren. Nun können insbesondere auch physische Eigenschaften wie Farben und Haptik einbezogen werden.

6.2.2.1 Spezifikation von Erwartungen

6.2.2.1.1 Methode: Modellierung von Erwartungen

Während der Durchführung eines Nutzertests werden vielfältige Interaktionsdaten aufgezeichnet (bspw. Positionsdaten, Informationen über verwendete Gegenstände oder Zustände von mobilen Geräten und Leinwänden). Die Fülle der Daten ist meist für die manuelle Auswertung zu groß. Zudem ist die Interpretation von Interaktionen auf dem Niveau physischer Ereignisse, wie Positionsdaten oder Tastaturanschlägen, schwierig. Ein Ausweg wurde bereits für die Evaluation klassischer Applikationen aufgezeigt. Hilbert et al. [HR00] abstrahieren physische Ereignisse über mehrere Abstraktionsstufen auf das Niveau von Aufgaben bzw. das von Nutzerzielen. Eine Sequenz von physischen Interaktionsereignissen

lässt sich so als wesentlich kürzere Sequenz von durchgeführten Aufgaben ausdrücken, was die direkte Interpretierbarkeit erlaubt. Bei den hier betrachteten Smart Environments sorgt eine Intentionserkennung durch statistische Modelle für diese Abstraktion. Nachdem eine Sequenz bearbeiteter Aufgaben automatisch ermittelt wurde, vergleicht ein Usability-Experte diese mit den für eine effiziente Lösung des Testfalles notwendigen. Bei Abweichungen ist zu untersuchen, ob ein Problem bei der Bearbeitung der nächsten erwarteten Aufgabe bestand oder ob ein alternativer effizienter Lösungsweg gefunden wurde. Dieser Vergleich zwischen beobachteter und erwarteter Aufgabensequenz ist ein manueller, zeitaufwendiger Prozess.

Um diesen Prozess zu automatisieren wird nachfolgend eine Methode vorgestellt, um das implizit beim Usability-Experten vorhandene Wissen explizit zu machen. Bevor ein Usability-Test durchgeführt wird, ist ein Testplan [Rub94] aufzustellen. Dieser enthält bspw. den Zweck des Tests, gewünschte Eigenschaften von Probanden, zu erfüllende Aufgaben, eine Beschreibung der Testumgebung, Hinweise zur Rolle des Moderators und eine Liste der aufzuzeichnenden Daten. Während des Entwurfes der Testaufgabenstellung und der zu schaffenden Rahmenbedingungen erhält der Usability-Experte und andere beteiligte Mitarbeiter von Fachabteilungen bereits einen guten Eindruck, wie die Nutzer während des Tests die Aufgabe effizient lösen können. Diese Erwartung soll nun maschinenlesbar spezifiziert werden, um einen automatischen Vergleich mit dem beobachteten Verhalten während des Testablaufs zu ermöglichen.

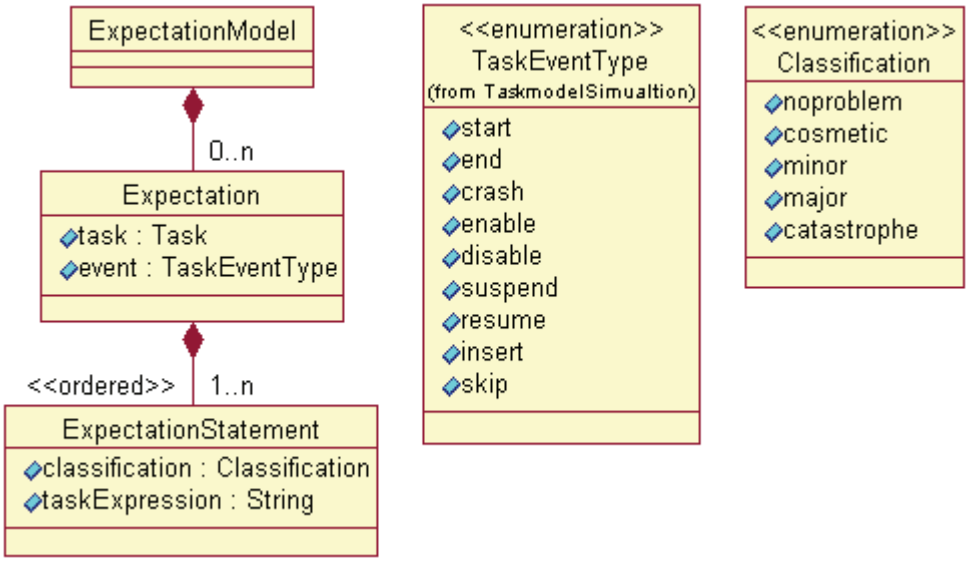

Abbildung 6.5: Aufbau eines ExpectationModel

Grundlage für die Spezifikation von Erwartungen ist ein Aufgabenmodell in der in Kapitel 4 beschriebenen Form. An jede Aufgabe lassen sich Erwartungen annotieren. Abbildung 6.5 zeigt den Aufbau eines ExpectationModel. Ein ExpectationModel beinhaltet eine Menge von Expectations. Jede dieser Expectations wird genau dann geprüft, wenn im Aufgabensimulator an eine Instanz einer bestimmten Aufgabe ein bestimmtes Ereignis geschickt wird.

Jede der Expectations beinhaltet eine Liste von ExpectationStatements, die der Reihe nach ausgewertet werden. Jedes ExpectationStatement enthält eine TaskExpression, die einen wahrheitswertigen Ausdruck auf dem aktuellen Weltzustand definiert, wie bspw. „user_a.isLocatedAt(location_PresentA)“ (Hält sich Teilnehmer A gerade in seiner Präsentationszone auf?). Ist dieser Ausdruck wahr, wird die zugeordnete Classification als Bewertung der laufenden Interaktion zurückgeliefert und im TaskEventTrace für das aktuelle Ereignis protokolliert. Ist dieser Ausdruck falsch, wird das nächste ExpectationStatement überprüft. Die Liste wird sequentiell abgearbeitet.

6.2.2.1.2 Klassifikation des Schweregrades eines Usability-Problems

Für die Klassifikation des Schweregrades eines Usability-Problems („severity rating“) gibt es verschiedene Methoden. Ziel einer Klassifikation der Probleme nach Schweregrad ist es, gezielt die schwerwiegenden Probleme zu identifizieren und zuerst zu lösen [DR99]. Eine Diskussionsrunde auf der UPA 2003 [Wil03] zeigte, dass in der Praxis viele pragmatische, selbstentwickelte Einteilungen genutzt werden und der Blickwinkel der Betrachtung variiert. Eine gebräuchliche Einteilung ist die Skala von Nielsen [Nie94], die eine Einteilung in vier Schweregrade vornimmt: (0 – kein Problem), 1 – kosmetisches, 2 – geringes, 3 – bedeutendes, 4 – katastrophales Problem. Um die Einteilung vornehmen zu können, unterscheidet Nielsen drei Dimensionen: Häufigkeit (Welcher Anteil der Nutzer bekommt das Problem?), Einfluss (Wie stark wird die Aufgabenausführung beeinträchtigt?) und Persistenz (Wie gut kann man das Problem bewältigen, nachdem man es das erste mal hatte?). Lindgaard [Lin94] nennt als Dimensionen ebenfalls Häufigkeit und Einfluss und fügt Frequenz dazu (Wie häufig während der Aufgabenausführung hat ein Nutzer das Problem durchschnittlich?). Für die Schweregrade nimmt sie eine 4-Teilung vor in geringe, mittlere, schwere und kritische Probleme.

Schweregrad	Beschreibung
0 – no problem	es besteht kein Problem
1 – cosmetic problem	braucht nicht behoben zu werden, wenn keine zusätzliche Zeit verfügbar ist
2 – minor problem	beheben mit kleiner Priorität
3 – major problem	beheben mit hoher Priorität
4 – usability catastrophe	Behebung unabdingbar, bevor das Produkt ausgeliefert werden kann

Tabelle 6.2: Klassifikation: Schweregrade von Usability-Problemen (adaptiert von: [Nie94])

Hier soll die Klassifikation von Nielsen in vier Kategorien genutzt werden. Wenn im konkreten Anwendungsfall eine andere Einteilung sinnvoll erscheint, kann diese ebenfalls genutzt werden. Dazu kann die Werkzeugunterstützung auf eigene Kategorien von Schweregraden umgestellt werden. Tabelle 6.2 gibt einen Überblick über die von Nielsen [Nie94] vorgeschlagenen Schweregrade. Zur Identifikation und Klassifikation der Probleme sollten

von mehreren Personen unabhängige Urteile eingeholt und anschließend zusammengeführt werden. Nielsen empfiehlt drei Bewertende [Nie94], um eine aussagekräftige Klassifikation zu erhalten. Diese sollten mit dem zu evaluierenden System vertraut sein und nach Möglichkeit auch zur Bewertung Zugang zum System haben.

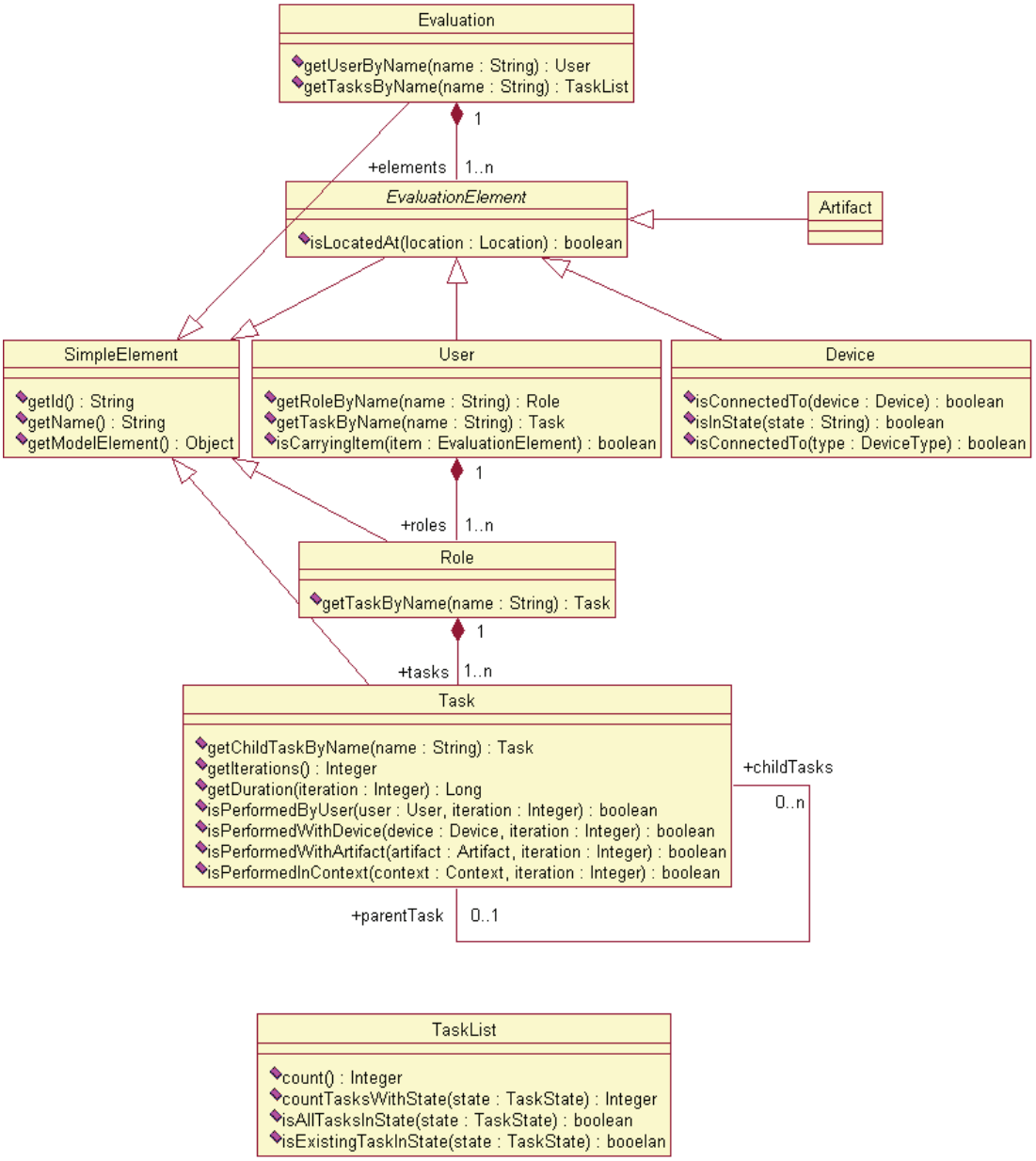

Abbildung 6.6: Modell für TaskExpressions zur Laufzeit

6.2.2.1.3 TaskExpression zur Beschreibung von Indikatoren von Problemen

Eine TaskExpression ist an eine Aufgabe im Aufgabenmodell gebunden und wird zur Laufzeit auf den entsprechenden Aufgabeninstanzen ausgewertet. So können Eigenschaften von Nutzern, ihrem Aufgabenfortschritt, dem Aufenthaltsort oder den mitgeführten Geräten abgefragt werden. Da die intern verwendeten Datenstrukturen viele Verwaltungsinformationen enthalten, wurde ein vereinfachtes Klassenmodell entworfen, das in Abbildung 6.6 dargestellt ist. Es enthält nur die für die Spezifikation von Erwartungen wichtigsten Entitäten. Wenn weitere Entitäten benötigt werden, kann von jedem Modellelement aus per „getModelElement()“ die dahinterliegende interne Entität abgefragt werden. Eine Methodenreferenz findet sich in Anhang A1.

Da der Ausdruck als JavaScript interpretiert wird, stehen die entsprechenden logischen und arithmetischen Operatoren bereit, sowie alle Attribute und Methoden der importierten Javaklassen. Möchte man die Sprache zur Beschreibung von TaskExpressions erweitern, genügt es, die in Abbildung 6.6 gezeigten Klassen zu verändern, die Anpassung eines Parsers ist nicht notwendig. Zur Auswertung einer TaskExpression werden alle aktuellen Objekte für Nutzer, Rollen, Aufgaben, Geräte und definierte Orte geladen und für die zu evaluierende TaskExpression zugänglich gemacht. Die Werkzeugunterstützung bietet bei der Eingabe von Ausdrücken automatisch passende Vorschläge als Textvervollständigung an.

In Tabelle 6.3 sind drei Beispiele dargestellt. Das erste Beispiel beschreibt eine Erwartung für den Fall, dass im Aufgabenmodell des gesamten Teams die Präsentation von Teilnehmer A beginnt. Wenn sich Nutzer A dabei nicht in der im Testplan festgelegten Präsentationszone befindet, wird der Problemkandidat als „major“ beurteilt, befinden sich Nutzer B oder C nicht an den vorab festgelegten Plätzen, als „minor“.

Nr.	Task	Event	Classification	TaskExpression
1	team.presentationA	start	major	!user_a.isLocatedAt(location_PresentA)
			minor	!user_b.isLocatedAt(location_SitB) || !user_c.isLocatedAt(location_SitC)
2	presenter.load_slides	end	minor	selfElement.getName().equals("A") && !device_LaptopA.isConnectedTo(device_LW5)
			catastrophic	selfElement.getName().equals("A") && !device_LaptopA.isConnectedTo(deviceType_ProjectionScreen)
3	participant.give_a_ presentation	end	minor	self_task.getDuration() < 30
			noProblem	self_task.getDuration() < 360
			major	self_task.getDuration() < 600
			catastrophic	true

Tabelle 6.3: Beispiele für spezifizierte Erwartungen

Das zweite Beispiel beschreibt den Fall, dass im Aufgabenmodell eines Vortragenden die Systemaufgabe „load_slides“ abgeschlossen wurde. Wenn der Videoausgang des Laptops von Nutzer A nicht auf Leinwand 5 ausgegeben wird, ist die Beurteilung „minor“. Wird der Videoausgang auf keinerlei Projektionsfläche ausgegeben, ist die Beurteilung „catastrophic“. Videoverbindungen zwischen Geräten (bspw. zwischen Laptop und Projektor) werden über die Rauminfrastruktur vorgenommen. Projektoren zeigen entweder statisch auf eine Leinwand oder lassen sich dynamisch auf eine Leinwand drehen. Die Methode „isConnectedTo(Device targetDevice)“ prüft, ob über direkte oder indirekte Weise eine Weiterleitung und/oder Projektion auf das angegebene Gerät/Leinwand erreicht wird. Es kann alternativ die Weitergabe auf einen Gerätetyp geprüft werden.

Das dritte Beispiel beschreibt den Fall, dass ein beliebiger Teilnehmer seinen Vortrag beendet hat. Wenn der Vortrag weniger als 30 Sekunden dauert, ist die Beurteilung „minor“, weil vermutlich eine Fehlerkennung der Intentionserkennung vorliegt, die durch die Erkennung der richtigen Aufgabe nach kurzer Zeit korrigiert wurde. Die Ausdrücke werden in der Reihenfolge ausgewertet, wie sie in der Tabelle 6.3 spezifiziert sind. Wenn der erste Ausdruck nicht zutrifft, liefert der zweite „noproblem“, der dritte „major“ oder der vierte dispositiv „catastrophic“.

Während der Durchführung des Testfalls liefert die automatische Auswertung der Erwartungen Kandidaten für Usability-Probleme. Eine manuelle Überprüfung wird damit nicht ersetzt, aber vereinfacht, und besteht in der Überprüfung dieser Klassifikation.

6.2.2.1.4 Evaluation

Das Ziel ist die automatische Erkennung von Kandidaten für Usability-Probleme, um aus der großen Masse aufgezeichneter Daten eine Vorauswahl zu treffen. Dazu werden Sensordaten und erkannte Aufgaben in chronologischer Reihenfolge in den kooperativen Aufgabensimulator geladen, der während der Aufgabenausführung die annotierten Erwartungen auswertet und protokolliert.

Unterschiede zwischen tatsächlicher und erwarteter Ausführung sind Kandidaten für Usability-Probleme. Ob tatsächlich ein Problem vorliegt, muss eine Untersuchung zusätzlicher Informationen ergeben. Dazu analysiert ein Usability-Experte für die betreffenden Interaktionen die Video- und Sensordatenaufzeichnungen. Mit einbezogen werden sollten auch weitere Verfahren, wie Fragebögen oder Interviews, um subjektive Empfindungen von Nutzern zu berücksichtigen.

6.2.2.1.5 Werkzeugunterstützung

Um die Spezifikation der Erwartungen zu vereinfachen, wurde ein grafischer Editor entwickelt. Dieser zeigt im linken Bildschirmbereich für jeden Nutzer des Testfalls die ausführbaren Aufgabenmodelle in einer Baumansicht an. Im rechten Bildschirmbereich stellt ein Gantt-Diagramm Restriktionen für die Reihenfolge und Dauer der Aufgabenausführung dar. Jeder Balken kann in mehrere Abschnitte geteilt werden. Entsprechend einer Farbkodierung sind Abschnitte grün („noProblem“), grüngelb („cosmetic“), gelb („minor“), orange

(„major“) oder rot („catastrophic“) gefärbt. Im unteren Bildschirmbereich kann für jede Aufgabe festgelegt werden, welche Nutzer, Geräte und Gegenstände bei der Aufgabenausführung beteiligt sein müssen.

Wenn für eine Aufgabe keinerlei Erwartungen festgelegt werden, kann diese in beliebiger Weise entsprechend des Aufgabenmodells ausgeführt werden und es findet keine Untersuchung der Kontextbedingungen statt. Nachdem die Erwartungen festgelegt sind, werden daraus automatisch die ExpectationStatements generiert. Für das Festlegen spezieller Ausdrücke können die generierten nachbearbeitet werden.

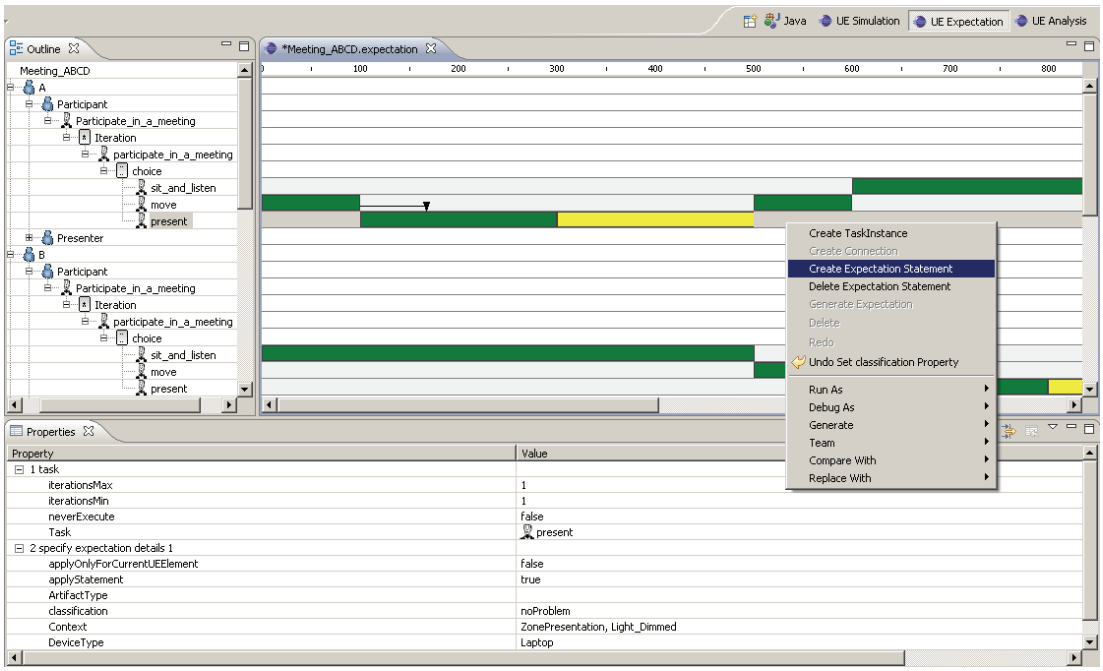

Abbildung 6.7: grafischer Editor zum Festlegen von Erwartungen

6.2.2.2 Datenanalyse

Während eines Nutzertests werden vielfältige Daten aufgezeichnet, sowohl native Sensordaten (bspw. Ortsinformationen der Nutzer, RFID-Informationen), als auch daraus abgeleitete Daten (bspw. der Intentionserkennung). Die Auswertung dieser großen Datenmengen ist eine Herausforderung. Nachfolgend wird ein Ansatz gezeigt, der semantisches Zusatzwissen aus den entwickelten Aufgabenmodellen nutzt, um eine Analyse aus der Perspektive der Aufgabenausführung zu ermöglichen.

Das Vorgehen gliedert sich in vier Schritte: (1) Datenvorbereitung, (2) Filterung, (3) Aggregation und (4) Normalisierung.

6.2.2.2.1 Datenvorbereitung

Die im physischen Smart Environment aufgezeichneten Daten liegen in verschiedenen, teils proprietären Formaten vor. Für das „Smart Environment“-Labor an der Universität Rostock wurde beispielhaft eine Anbindung implementiert, die die Daten in das virtuelle Smart Environment importiert. Der von der Intentionserkennung erkannte Fortschritt in der Aufgabenausführung wird dabei mit Hilfe des kooperativen Aufgabenmodellsimulators ausgewertet. Abweichungen zwischen erkannter und laut Modell erlaubter Aufgabenausführung werden automatisch protokolliert. Wie in Kapitel 4.2 beschrieben, werden in der Simulationsumgebung Ereignisse vom Start und Ende von Aufgaben aufgezeichnet. Nachfolgend wird diese Sequenz von Ereignissen der Aufgabenausführung als „TaskEventTrace“ bezeichnet. Ein TaskEvent besteht aus:

- einem Zeitstempel (wann das Ereignis ausgelöst wurde),
- der Aufgabeninstanz (mit Referenz auf die betroffene Aufgabe),
- dem Ereignis des Zustandsautomaten der Aufgabeninstanz (bspw. „start“, „end“, „skip“),
- dem Erfolg der Ausführung (Erlaubt das Aufgabenmodell dieses Ereignis im aktuellen Zustand? Einschränkungen ergeben sich durch temporale Abhängigkeiten und kontextuellen Vorbedingungen.),
- dem ausführenden Nutzer (bzw. dem ausführenden Team),
- dem Ort des Nutzers zu diesem Zeitpunkt und
- beteiligten Geräten oder anderen Gegenständen.

Besondere Bedeutung haben die Ereignisse „start“, „end“ und „crash“, da sie das Zeitintervall der Ausführung einer Aufgabeninstanz begrenzen. Entsprechend wurde ein Algorithmus implementiert, der aus den Ereignissen die ausgeführten Aufgaben und ihre Ausführungsdauer ableitet. Der so entstandene „TaskTrace“ umfasst die Sequenz aller ausgeführten Aufgaben für ein bestimmtes Aufgabenmodell.

6.2.2.2.2 Filterung

Das Protokoll der ausgeführten Aufgaben bezieht sich auf alle Nutzer im Raum. Je nach Ziel der Analyse sind allerdings nur bestimmte Daten von Interesse. Wenn bspw. ein neues Gerät in den Raum eingebaut wurde, kann man gezielt die Interaktionen filtern, die mit diesem Gerät in Zusammenhang stehen.

Abbildung 6.8 zeigt die Werkzeugunterstützung für das Filtern des aufgezeichneten TaskEventTrace. Folgende Filterkriterien werden in dieser Reihenfolge angezeigt:

- Evaluation: Jedes vorgegebene Testszenario wird mehrfach mit verschiedenen konkreten Nutzern durchgespielt. Die Daten jeder dieser Evaluationsläufe können separat oder vergleichend analysiert werden.

- Akteur: In jedem Testszenario werden mehrere handelnde Akteure festgelegt, die jeweils bestimmte Aufgaben bekommen und zum Gesamtziel beitragen.
- Aufgabe: von der Intentionserkennung ermittelte Aufgabe
- Ereignistyp: Typ des gefeuerten Ereignisses
- Aufgabenzustand: Zustand der Aufgabeninstanz nach Auswertung des Ereignisses
- Erfolg: Ist das eingehende Ereignis im aktuellen Zustand der Aufgabeninstanz erlaubt?
- Nutzer: Welche konkrete Testperson ist an der Ausführung beteiligt?
- Rolle: Welche Rolle hat der Nutzer?
- Gerät: Welches konkrete Gerät wird verwendet?
- Gerätetyp: Von welchem Typ ist dieses Gerät?
- Kontext: In welchem Kontext wurde die Aufgabe ausgeführt?
- Kontexttyp: Welche Kontextarten sind von Interesse?

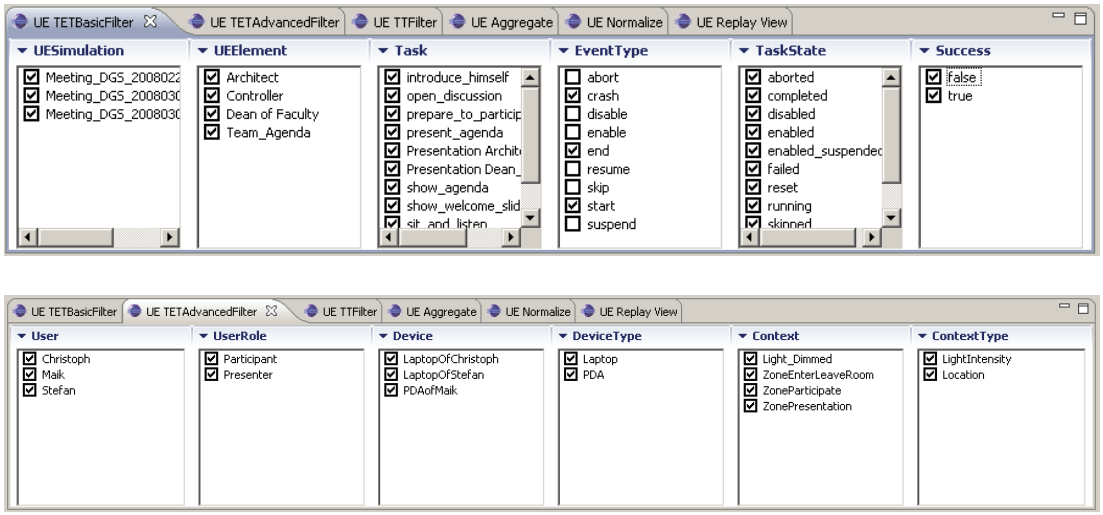

Abbildung 6.8: Filteroptionen für einen TaskEventTrace

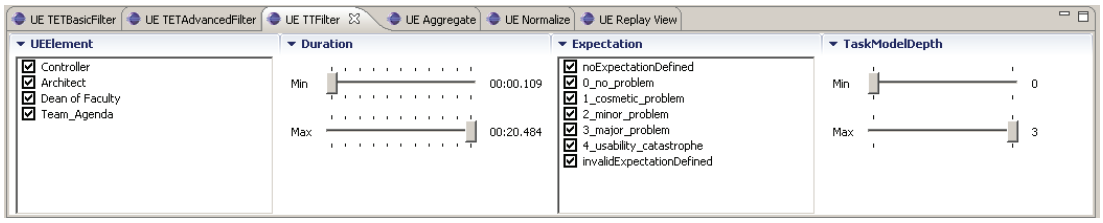

Abbildung 6.9: Filteroptionen für einen TaskTrace

Abbildung 6.9 zeigt die Werkzeugunterstützung für das Filtern des aufgezeichneten TaskTrace. Folgende Filterkriterien werden in dieser Reihenfolge angezeigt:

- Akteur: In jedem Testszenario werden mehrere handelnde Akteure festgelegt, die jeweils bestimmte Aufgaben bekommen und zum Gesamtziel beitragen.
- Dauer: Ausführungsdauer einer Aufgabe (Eine Möglichkeit ist das Filtern nach außergewöhnlich kurzen Aufgaben, um zu überprüfen, ob eine Aufgabe nach dem Beginn abgebrochen wurde.)
- Erwartung: Vergleich der tatsächlichen Aufgabenausführung mit der spezifizierten Erwartung für das Testszenario
- Aufgabenmodelltiefe: Bis zu welcher Detailtiefe sollen Aufgaben angezeigt werden? (Es empfiehlt sich zunächst mit einem geringeren Detailgrad zu beginnen, um Probleme grob einzugrenzen und nur die Ausführung dieser Aufgaben detaillierter zu untersuchen.)

6.2.2.2.3 Semantische Aggregation

Ein aufgezeichneter TaskTrace kann je nach Dauer und Detailgrad der Aufzeichnung einer Aufgabenausführung sehr lang und damit unübersichtlich werden. Je nach Ziel der Analyse sind nur bestimmte Teile des TaskTrace relevant. Während das Filtern irrelevante Daten komplett entfernt, wird nun eine Fokus- und Kontexttechnik vorgestellt, die alle Daten beibehält aber nur im Fokusbereich einen großen Detailgrad darstellt, während dieser für Daten außerhalb des Fokusbereiches verkleinert wird.

Fokus- und Kontexttechniken stammen aus dem Bereich der Datenvisualisierung [Car07 (S.536ff)]. Dahinter steckt die Idee, dass Nutzer beides brauchen: einen breiten Überblick über das Datenmaterial (Kontext) und Detailinformationen im Bereich des Interesses (Fokus). Der Kontextbereich soll wenig detailliert sein, um nicht vom Wesentlichen abzulenken, aber dennoch detailliert genug, um abschätzen zu können, wo sich eine weitere, genauere Analyse lohnt.

Im Rahmen dieser Arbeit wurde ein Algorithmus für die semantische Aggregation von TaskTraces entwickelt. Ausgangspunkt ist eine Sequenz ausgeführter Aufgaben und ein Aufgabenmodell, das die hierarchischen Beziehungen zwischen den Aufgaben festlegt. Der TaskTrace darf auch Aufgaben enthalten, die nicht im Aufgabenmodell enthalten sind. Diese werden zwar nicht aggregiert, aber im resultierenden TaskTrace an der richtigen Position in der zeitlichen Folge wieder angezeigt.

Abbildung 6.10 zeigt den Basisalgorithmus für die Aggregation in vereinfachter, Java-ähnlicher Notation. Der TaskTrace wird sequentiell durchlaufen und in Teilsequenzen von aufeinander folgenden Aufgaben zerlegt, die den gleichen Vaterknoten im Aufgabenbaum besitzen (Zeile 9f). Für jede Sequenz muss nun bestimmt werden, ob eine Aggregation durchzuführen ist.

Das notwendige Kriterium (Zeile 13-16) ist, dass weder die Aufgabe unmittelbar vor, noch unmittelbar nach der Sequenz ein Kind oder Kindeskind in direkter Hierarchie unter dem Elternknoten der zu prüfenden Sequenz ist. Dies ist deshalb wichtig, da eine Aggregation der zu prüfenden Sequenz an dem benachbarten Knoten „vorbeiführen“ würde ohne ihn einzuschließen. Daher wird zunächst keine Aggregation durchgeführt, bis der benachbarte

tiefere Knoten über eine oder mehrere Aggregationen ebenfalls auf höherem Niveau ist und dann gemeinsam mit der diskutierten Sequenz aggregiert werden kann.

```
01  ApplySemanticAggregation(↔ til:TaskInstanceList) {
02    currentSequence.clear()
03    boolean aggregationDone? := true
04    while (aggregationDone?) {
05      aggregationDone? := false
06      for each (ti in til) {
07        if (currentSequence is empty) {
08          currentSequence.add(ti)
09        } else if (currentSequence.first().parent().equals(ti.parent())) {
10          currentSequence.add(ti)
11        } else {
12          parent := currentSequence.first().parent()
13          if ((til.previousTi(currentSequence.first()).parent()
14               .isSubNodeOf(parent)) and
15              (til.nextTi(currentSequence.last()).parent)
16               .isSubNodeOf(parent)) and
17              (fitsBetterToLensFunction(parent, currentSequence))) {
18            til.replace(currentSequence, parent)
19            aggregationDone? := true
20          } else {
21            currentSequence.clear()
22            currentSequence.add(ti)
23          }
24        }
25    }
26  }
```

Abbildung 6.10: Basisalgorithmus der semantischen Linsenfunktion

Das hinreichende Kriterium (Zeile 17) ist, dass der TaskTrace nach der Aggregation näher an der semantischen Linsenfunktion liegt als vorher. Das Prinzip einer semantischen Linse beeinflusst den Prozess der Datenvisualisierung [Chi00] schon in den frühen Phasen. Im Gegensatz dazu kommen optische Linsen erst in den späteren Phasen zum Einsatz [GFS05]. An Hand der Abbildung 6.11 soll die Linsenfunktion erläutert werden. Unten ist eine Aufgabensequenz „A, B, C, …“ abgebildet, wobei V der Vaterknoten von A, B und D im Aufgabenmodell ist. Die Dauer der Aufgaben ist horizontal abgebildet, die Hierarchie vertikal. Die Tiefe einer Aufgabe im Aufgabenbaum bestimmt sich dabei über die Entfernung einer Aufgabe zur Wurzel. Die Linsenfunktion besteht aus zwei Teilfunktionen, die im

nun diskutierten Beispiel (Abbildung 6.11) die Form von Geraden haben. Die erste Teilfunktion verläuft von einer Anfangszeit (t_A) bis zu einer Zeit mit dem Fokus (t_F) von Punkt (t_A, 0) zu (t_F, maximale Tiefe). Die zweite Teilfunktion verläuft von der Zeit mit dem Fokus (t_F) bis zu einer Endzeit (t_E) von Punkt (t_F, maximale Tiefe) zu (t_E, 0). Damit ist für jeden Zeitpunkt eine angestrebte Tiefe nach der Aggregation gegeben. Da diese im Bereich der rationalen Zahlen liegt, die tatsächliche Tiefe einer Aufgabe hingegen im Bereich der natürlichen Zahlen, ist dieses angestrebte Ziel nur in Sonderfällen vollständig zu erreichen. Eine Aggregation von (A, B, C) zu (V) wird nur dann durchgeführt, wenn der Abstand der Vateraufgabe kleiner ist als die Summe der Abstände der Teilaufgaben.

Im Beispiel: $\Delta d(V) < \Sigma(\Delta d(A), \Delta d(B), \Delta d(C))$.

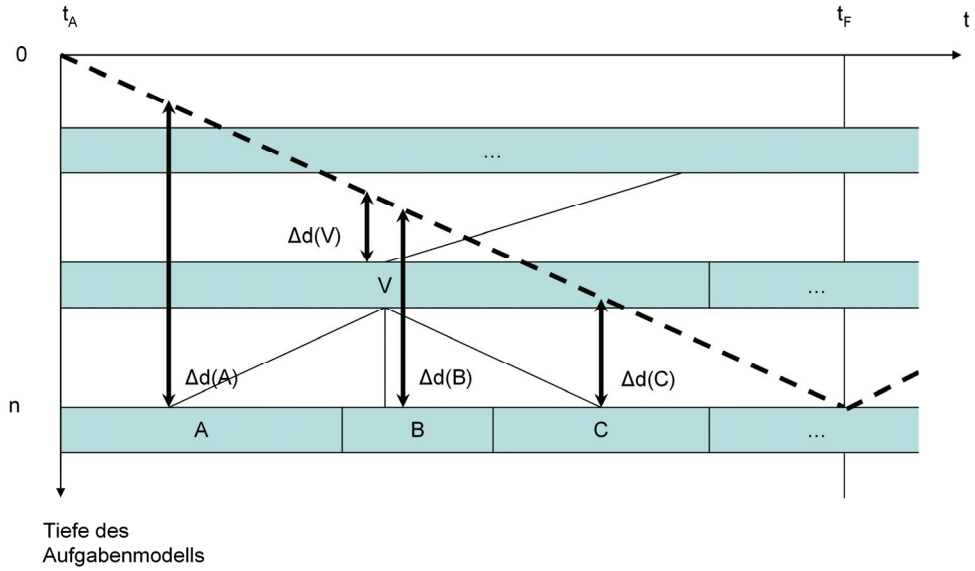

Abbildung 6.11: Schema einer einfachen Linsenfunktion

Ein Beispiel soll das Prinzip verdeutlichen. Abbildung 6.12 zeigt das Aufgabenmodell für die Rolle des Vorsitzenden einer Besprechung. Ein Beispiel-TaskTrace auf der Basis dieses Aufgabemodells ist in Tabelle 6.4 gegeben. Die angestrebte Assistenz im Raum sollte nach dem Eintreten der Teilnehmer das Licht entsprechend des Sonnenlichtes regeln und nach dem der Vorsitzende nach vorn gegangen ist, seine Folien von seinem persönlichen Gerät laden. Die Agenda wurde dabei nicht auf dem Hauptprojektor angezeigt, sondern von ihm aus gesehen auf einem Projektor an der rechten Wand. Nun soll die Sequenz der erkannten Aufgaben analysiert werden, um herauszufinden, ob das Problem in der Aufgabenausführung liegt, dass bspw. Aufgaben falsch erkannt wurden oder eine vorab nicht erwartete Aufgabenfolge auftrat, die eine besondere Assistenz verlangt.

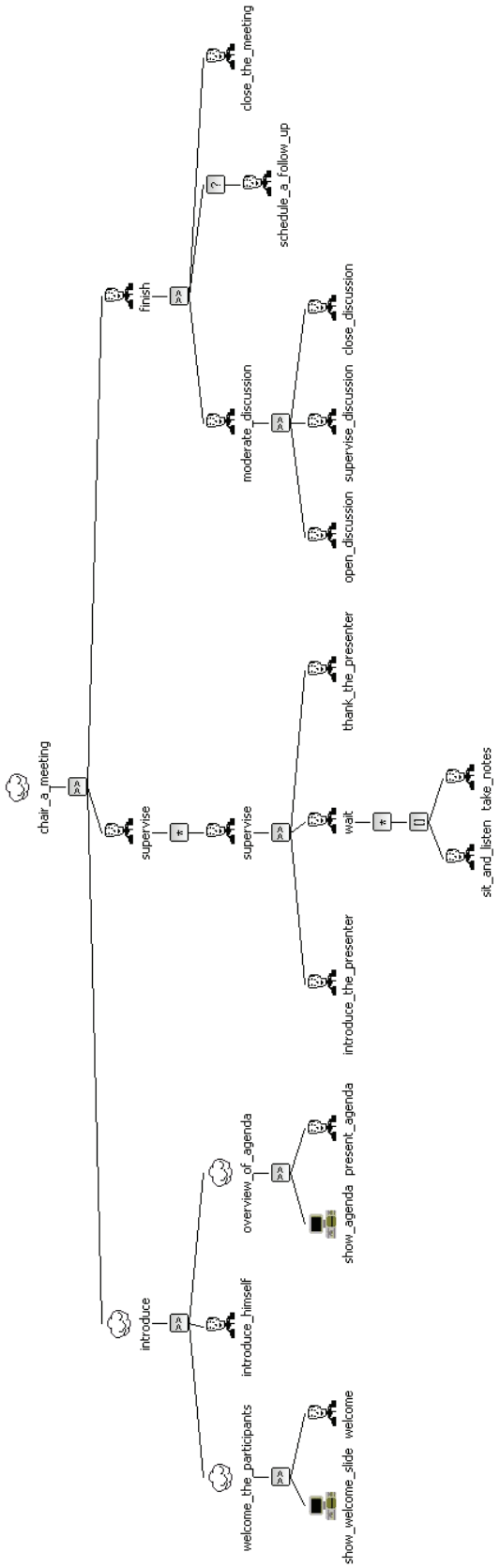

Abbildung 6.12: Aufgabenmodell Vorsitzender

Aufgabe	Tiefe im Baum	Blattaufgabe?	Anzahl der umfassten Blattaufgaben
show_welcome_slide	3	x	1
welcome	3	x	1
introduce_himself	2	x	1
show_agenda	3	x	1
present_agenda	3	x	1
introduce_the_presenter	3	x	1
sit_and_listen	4	x	1
take_notes	4	x	1
thank the presenter	3	x	1
open_discussion	3	x	1
supervise_discussion	3	x	1
close_discussion	3	x	1
schedule_a_follow_up	2	x	1
close_the_meeting	2	x	1

Tabelle 6.4: Beispiel-TaskTrace

Aufgabe	Tiefe im Baum	Blattaufgabe?	Anzahl der umfassten Blattaufgaben
welcome_the_participants	2	-	2
introduce_himself	2	x	1
show_agenda	3	x	1
present_agenda	3	x	1
supervise	2	-	4
finish	1	-	5

Tabelle 6.5: Beispiel-TaskTrace nach der Aggregation (Fokus auf: „show_agenda“)

Der TaskTrace in Tabelle 6.4 umfasst 14 Blattaufgaben des Aufgabenbaumes. Durch die Anwendung der Aggregation soll ein kleiner Bereich des Interesses detailliert angezeigt werden, alle übrigen Aufgaben weniger detailliert entsprechend ihrer Entfernung zum Fokus. Der Fokus wird auf einen Zeitpunkt gelegt, zu dem die Aufgabe „show_agenda“ ausgeführt wird. Der aggregierte TaskTrace wird in Tabelle 6.5 angezeigt. Drei Aufgaben rund um den Fokus werden detailliert angezeigt, alle übrigen wurden aggregiert, die Aufgaben „welcome_the_participants“ und „supervise“ auf eine Tiefe von 2 und die Aufgabe „finish“ auf Grund ihrer großen zeitlichen Entfernung zum Fokus auf eine Tiefe von 1.

Der Fokus lässt sich interaktiv vom Nutzer auf einer Zeitleiste verschieben. Das Ergebnis der Aggregation ist sofort in tabellarischer oder grafischer Form sichtbar.

6.2.2.2.4 Normalisierung

Für die Identifikation von Usability-Problemen aus dem aufgezeichneten Datenmaterial gibt es mehrere Möglichkeiten. Eine Möglichkeit ist der Vergleich der Ausführungszeit verschiedener Nutzer untereinander oder der Vergleich mit dem effizienten Verhalten eines Fachexperten. Braucht ein Nutzer deutlich länger für die Erfüllung einer Aufgabe, ist dies ein Indikator für ein Problem, was näher zu untersuchen ist. Braucht ein Nutzer für eine Aufgabe deutlich weniger Zeit als andere, sollte untersucht werden, ob die Aufgabe richtig erfüllt wurde oder ob die Intentionserkennung eine Aufgabe falsch erkannt hat. Bei einigen Konstellationen von Sensordaten gibt es aus Sicht der Intentionserkennung mehrere wahrscheinliche Deutungen, aus der eine gewählt wird. Stellt sich bei der weiteren Interaktion heraus, dass eine andere Aufgabe durchgeführt wird, wird nach kurzer Meldung der ersten Aufgabe eine weitere zurückgegeben. Dadurch entstehen im TaskTrace kurze Fragmente real nicht durchgeführter Aufgaben.

Filtert man im TaskTrace die besonders kurzen oder langen Aufgaben, kann man bereits manche dieser Fälle identifizieren. Allerdings findet man gleichzeitig auch Aufgaben, die naturgemäß sehr kurz sind, wie bspw. die Armbewegung zum Öffnen eines Fensters, oder sehr lang sind, wie bspw. eine mit längerer Dauer eingeplante Präsentation. Zudem hängt die Dauer der erkannten und aufgezeichneten Aufgaben auch stark von der Granularität der Modellierung ab. Verschiedene Teile des Aufgabenmodells können unterschiedlich feingranular spezifiziert worden sein. Gründe dafür können in der Aufgabencharakteristik liegen, wenn bspw. eine Aufgabe aus Sicht des Modellierers keine interessanten Unteraufgaben enthält, oder wenn sich aus der Perspektive der vorhandenen Sensoren Unteraufgaben detailliert erkennen lassen.

Durch das Normalisieren wird keine Aufgabe als absolut kurz oder lang bewertet, sondern relativ zu anderen Nutzern. Ein Ansatz ist, die Länge mehrerer TaskTraces zur Erfüllung derselben Aufgabe so zu strecken oder zu stauchen, dass sich die gleiche Gesamtdauer ergibt.

Ein Beispiel enthält Tabelle 6.6. Ein mit dem Raum vertrauter Experte bekommt die Aufgabe, in der Rolle eines Vorsitzenden eine Besprechung zu leiten. Eine Agenda gibt für den Usability-Test den Ablauf vor, Dauer und Inhalt der Vorträge, verwendete persönliche Geräte und einen Rahmen für die Interaktionen der Teilnehmer. Abweichungen von der geplanten Dauer sind möglich, bspw. wenn das persönliche Gerät mit den Folien sich nicht mit dem richtigen Projektor verbindet, Leinwände nicht heruntergefahren sind oder die Lichtverhältnisse im Raum schlecht sind. In solchen Fällen kann der Nutzer einen Augenblick warten, um zu sehen, ob das Smart Environment das Problem selbst erkennt und behebt, oder über eine manuelle Steuerung eingreifen, um das entsprechende Gerät per Hand zu justieren. Der Experte hat dabei den Vorteil, dass er schnell und gezielt mit der manuellen Steuerung Probleme bei der Assistenz korrigieren kann. Zudem kann er meist besser intuitiv abschätzen, ob die beginnende Aktionssequenz zur Ansteuerung der Geräte für die momentan gewünschte Assistenz zielführend sein wird und sich damit ein eventuelles Warten lohnt oder manuelles Eingreifen notwendig ist. Ein Nutzer mit wenig oder ohne Vorerfahrung

kann erst nach Abschluss der kompletten Sequenz erkennen, ob die Assistenz angemessen ist und braucht damit für solche Aufgaben mehr Zeit.

Tabelle 6.6 zeigt beispielhaft einen Verlauf einer Besprechung, die als Sequenz künstlicher Sensordaten im virtuellen Smart Environment erstellt wurde. Ein Nutzer und ein Experte führen Aufgaben aus dem Aufgabenmodell des Vorsitzenden aus. Ausführungszeiten und die zeitliche Differenz sind angegeben.

Aufgabe	Experte	Nutzer	Differenz
show_welcome_slide	00:00:18	00:00:31	- 00:00:13
welcome	00:01:12	00:02:09	- 00:00:57
introduce_himself	00:00:55	00:02:12	- 00:01:17
show_agenda	00:00:15	00:02:35	- 00:02:20
present_agenda	00:01:03	00:01:42	- 00:00:39
introduce_the_presenter	00:01:21	00:02:21	- 00:01:00
sit_and_listen	00:03:23	00:06:11	- 00:02:48
take_notes	00:01:42	00:03:06	- 00:01:24
thank the presenter	00:00:37	00:01:11	- 00:00:34
open_discussion	00:00:52	00:01:14	- 00:00:22
supervise_discussion	00:02:08	00:03:57	- 00:01:49
close_discussion	00:00:47	00:01:08	- 00:00:21
schedule_a_follow_up	00:01:35	00:03:03	- 00:01:28
close_the_meeting	00:00:31	00:00:49	- 00:00:18
Summe	00:16:39	00:32:09	

Tabelle 6.6: Zwei TaskTraces im Vergleich

Aufgabe	Experte	Nutzer	Differenz
show_welcome_slide	00:00:18	00:00:16	+ 00:00:02
welcome	00:01:12	00:01:07	+ 00:00:05
introduce_himself	00:00:55	00:01:08	- 00:00:13
show_agenda	00:00:15	00:01:20	- 00:01:05
present_agenda	00:01:03	00:00:53	+ 00:00:10
introduce_the_presenter	00:01:21	00:01:13	+ 00:00:08
sit_and_listen	00:03:23	00:03:12	+ 00:00:11
take_notes	00:01:42	00:01:36	+ 00:00:06
thank the presenter	00:00:37	00:00:37	+ 00:00:00
open_discussion	00:00:52	00:00:38	+ 00:00:14
supervise_discussion	00:02:08	00:02:03	+ 00:00:05
close_discussion	00:00:47	00:00:35	+ 00:00:12
schedule_a_follow_up	00:01:35	00:01:35	+ 00:00:00
close_the_meeting	00:00:31	00:00:25	+ 00:00:06
Summe	00:16:39	00:16:39	

Tabelle 6.7: Zwei TaskTraces normalisiert nach der Gesamtdauer des ersten TaskTraces

Der Nutzer braucht im Beispiel mehr Zeit als der Experte. Beim Normalisieren der Ausführungsdauer wird der TaskTrace des Nutzers mit dem Faktor 0,52 gestaucht. Dazu wird die Dauer jeder Aufgabe entsprechend einzeln verkürzt. In der Spalte Differenz der Tabelle 6.6 wird deutlich, dass jede Aufgabe des Nutzers länger ist, als die des Experten. Nach der Normalisierung in Tabelle 6.7 sind nur noch zwei Aufgaben länger. „introduce_himself“ ist etwas länger, während „show_agenda“ deutlich länger ist. Die Ursache dessen sollte an Hand weiterer Informationen, wie bspw. Videoaufzeichnungen und Sensordaten näher untersucht werden. Die Aufgaben „open_discussion“ und „close_discussion“ wurden vergleichsweise schnell gelöst. In solchen Fällen kann untersucht werden, ob die Aufgabe vollständig bearbeitet wurde oder ob ein alternativer Lösungsweg gefunden wurden, der eine effizientere Bearbeitung erlaubt, was zukünftig vom Assistenzsystem konsequent mit unterstützt werden soll.

Alternativ zur Normalisierung an Hand der Gesamtdauer kann auch die Dauer einer bestimmten Blattaufgabe oder inneren Aufgabe des Baumes herangezogen werden. Dies ist besonders dann geeignet, wenn Nutzer für Tätigkeiten systematisch länger brauchen als andere. Beispiele sind Aufgaben mit einer großen Anzahl an Tastatureingaben, bei denen die Anzahl der Anschläge je Minute zwischen Nutzern stark variieren kann, oder Räume mit Stufen, die Nutzer in unterschiedlicher Geschwindigkeit zu überwinden in der Lage sind. Eine zusätzliche Aufgabe vorab kann solche typischen Bewegungen enthalten und zur Berechnung der Normalisierung dienen, so dass bereits während der Aufgabenausführung die aufgezeichneten Aufgaben in normalisierter Dauer angezeigt werden können.

6.2.2.2.5 Prozess der Datenverarbeitung

Je nach Interesse der Evaluation kann der TaskTrace anders aufbereitet werden. Ein Beispiel für eine Kette von Operatoren für die Transformation ist in Abbildung 6.13 gegeben.

Abbildung 6.13: Beispielschema zur Transformation eines TaskTrace

Entsprechend des Architekturmusters „Pipes-and-Filters“ [BMR+00] besitzt jeder Operator die gleiche Schnittstelle: einen TaskTrace als Eingabe zusammen mit Parametern zur Steuerung der Transformation und als Ausgabe wieder einen TaskTrace. Dadurch können die Operatoren in beliebiger Reihenfolge durchgeführt werden. Beispielhaft wurden Filterung, Aggregation und Normalisierung beschrieben. Weitere Operatoren, wie bspw. spezielle Filter, können nach Bedarf implementiert werden.

Zur Steuerung der Reihenfolge der Operatoren wurde eine Koordinationssicht in der grafischen Oberfläche entwickelt. Sobald man einen neuen Operator parametrisiert, wird dieser am Ende des Transformationsprozesses angehängt. Die Koordinationssicht zeigt die Reihenfolge an und erlaubt diese zu verändern, also Operatoren vor und zurück zu verschieben. So ist es einerseits bspw. möglich, zuerst die Filterung durchzuführen, um sehr kurze

Aufgaben zu entfernen und diese anschließend nicht normalisieren zu müssen. Andererseits kann man auch zuerst die Normalisierung durchführen, um auf Basis der neu berechneten Zeiten die nun kurzen Aufgaben zu filtern.

6.2.2.3 Visualisierung

Nachdem die aufgezeichneten Daten aus der Perspektive der Aufgabenausführung aufbereitet wurden, soll ihre Visualisierung eine leichtere Interpretation ermöglichen.

Der in Kapitel 5.2 vorgestellte kooperative Aufgabenmodellsimulator wurde so erweitert, dass Sensordaten aus einem physischen Smart Environment aufbereitet und visualisiert werden können. Abbildung 6.14 zeigt das virtuelle Smart Environment. Es besteht aus einer Menge von Sichten, die sich beliebig anordnen und bei neuen Anforderungen modular erweitern lassen. Für verschiedene Arbeitsschritte, wie bspw. Modellierung und Evaluation, lassen sich sogenannte Perspektiven festlegen, die die Anordnung und Größe der Sichten speichern und einen schnellen Wechsel dazwischen erlauben. Die abgebildete Analyseperspektive teilt den Bildschirm in vier Bereiche:

(1) Der ContextView im linken Teil zeigt eine Draufsicht des Smart Environments als 2D-Darstellung der Sensordaten. Die betretenen Positionen der Nutzer können wahlweise alle als Punkte angezeigt werden oder reduziert auf aufeinanderfolgende Positionen mit einem gewissen Mindestabstand zueinander. Verbindungslinien zwischen den Punkten können den zurückgelegten Pfad abbilden. Jeder Nutzer erhält eine eigene Farbe. Immer wenn ein Nutzer die Ausführung einer Aufgabe beginnt oder beendet wird dies durch eine etwas vergrößerte Wegmarke dargestellt. Verweilt man einen Augenblick mit dem Mauszeiger darüber, werden der Aufgabename und das Ereignis angezeigt. Wählt man eine Menge solcher Punkte aus, bspw. in dem man ein Auswahlrechteck im Bereich des Interesses aufzieht, werden alle ausgeführten Aufgabeninstanzen gefiltert und im Ganttdiagramm (Abbildung 6.14) dargestellt. Der ContextView ist keine bloße Anzeige von Sensordaten, sondern zugleich ein Filter für die aufgezeichneten Interaktionssequenzen. So können Nutzer ausgewählt werden, um deren Aufgaben im TaskTrace zu filtern. Abbildung 6.15 zeigt dies für das Auswählen von Nutzer A. Auch Sensordaten lassen sich auf diese Weise filtern. Abbildung 6.16 zeigt die Sensordaten der Positionsbestimmung für Nutzer A und Statuswechsel der verwendeten Geräte. Durch die grafische Auswahl von Nutzern und Geräten können jeweils deren Sensordaten selektiert werden. Da die Anzahl der Positionsdatensätze in den getesteten Beispielen sehr groß war, wurden diese für eine bessere Übersichtlichkeit verblasst dargestellt. Die Anzeige im ContextView umfasst insgesamt die Positionen von Nutzern, Geräten und Gegenständen, die in den Händen von Nutzern befindlichen Geräte und die Videoverbindungen zwischen Geräten. Ergänzend zeigt eine Eigenschaftssicht weitere Eigenschaften der Entitäten, wie Größe oder Zustand von Geräten.

(2) Im oberen rechten Teil ist für jeden Nutzer und jede der Rollen die Aufgabenmodellinstanz mit dem Fortschritt der Aufgabenausführung angezeigt. Der Fortschritt wird wahlweise durch Intentionserkennung oder Beobachtung ermittelt. Wählt man mit der Maus im ContextView einen Nutzer aus, werden alle Modellinstanzen zugeklappt und nur für den ausgewählten Nutzer angezeigt.

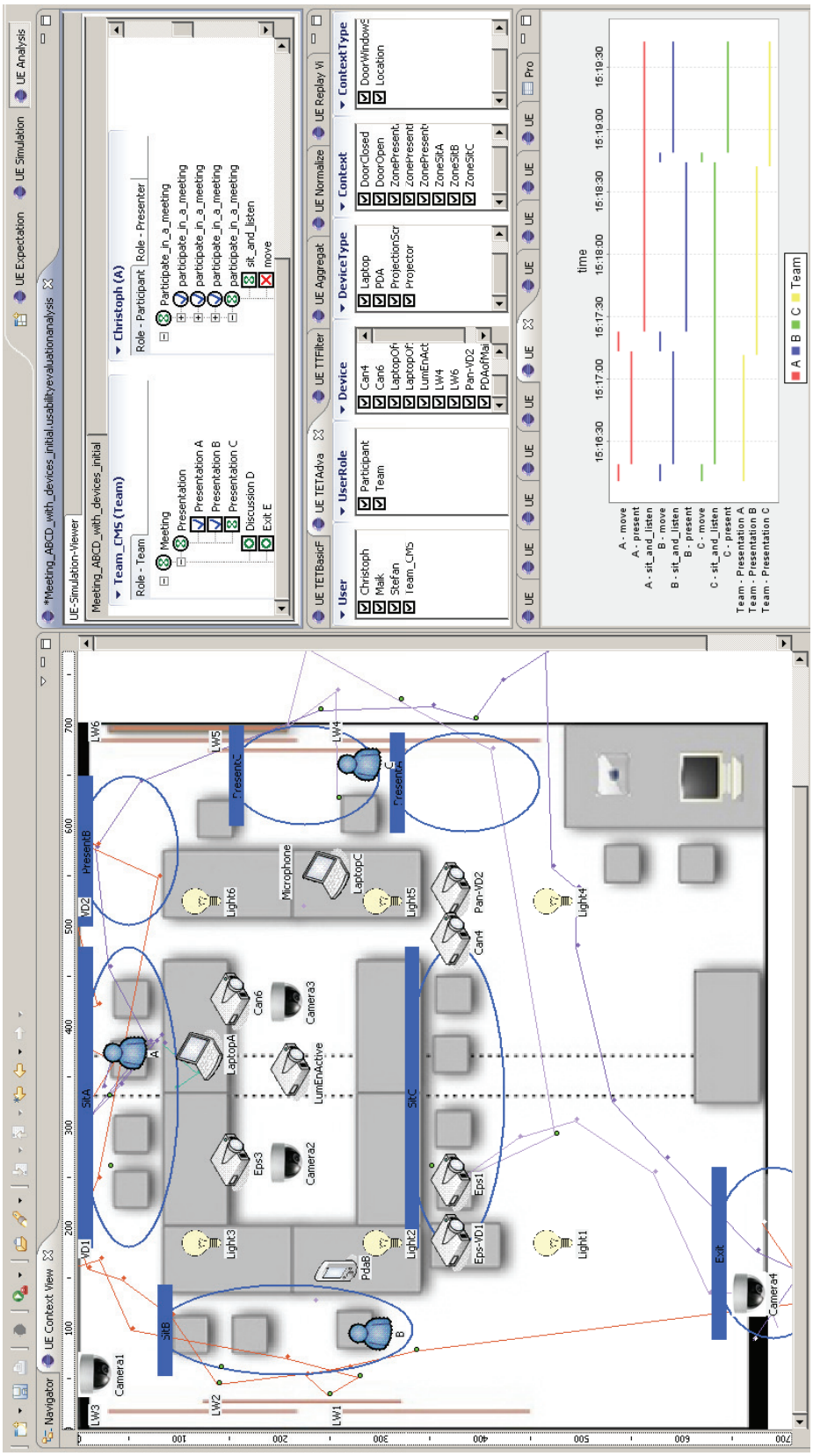

Abbildung 6.14: Visualisierung der Interaktionen im Smart Environment

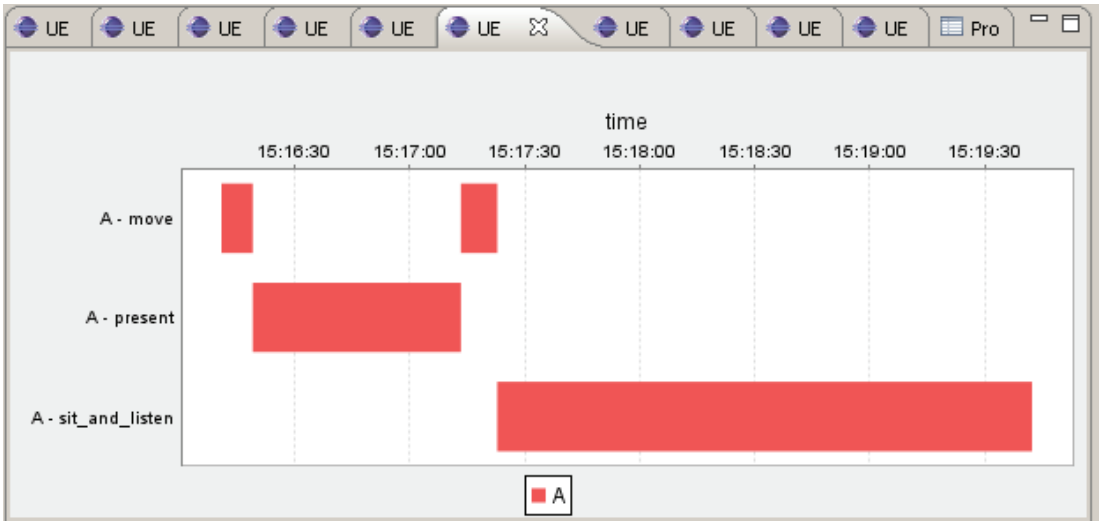

Abbildung 6.15: Detailsicht: Interaktion von Nutzer A als Ganttdiagramm

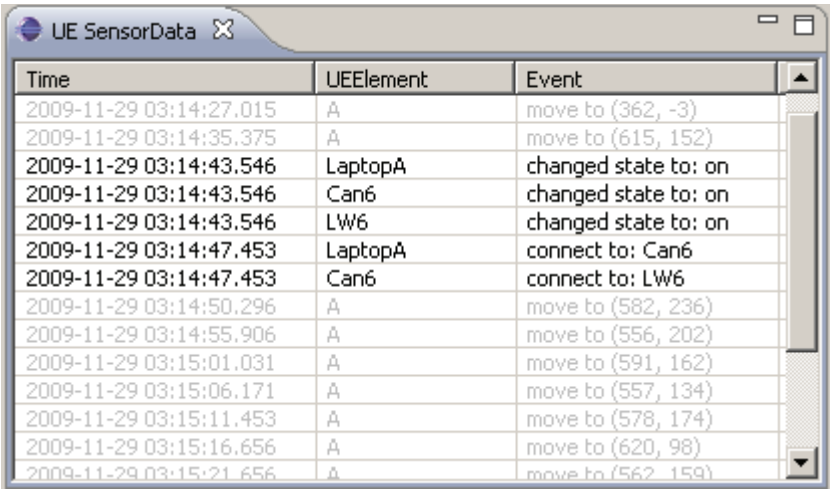

Abbildung 6.16: Detailsicht: Sensordaten des Nutzers A und verwendete Geräte

(3) Im rechten Teil der mittlere Bereich beinhaltet die Sichten zur Aufbereitung der aufgezeichneten Daten. Für Filterung, Aggregation und Normalisierung können die Parameter eingestellt werden. Jede dieser Sichten bildet einen grafisch dargestellten Schritt im Datenverarbeitungsprozess ab. Wenn sich der Inhalt einer dieser Sichten ändert oder die Aufgabenausführung voranschreitet, wird der TaskTrace in allen dargestellten Sichten aktualisiert.

(4) Im rechten Teil im unteren Bereich wird das Ergebnis der aufgabenbezogenen Datenaufbereitung angezeigt. Dafür stehen verschiedene Sichten bereit. In der Abbildung 6.13 ist ein Ganttdiagramm dargestellt, das die ausgeführten Aufgaben der einzelnen Nutzer und des Teams über den Zeitverlauf hinweg darstellt. Es wird während der Aufgabenausführung ständig aktualisiert und passt sich den aktuellen Filtern an. Alternative Sichten umfassen die Darstellung des TaskEventTrace als Tabelle, sowie verschiedene Diagramme des TaskTrace.

Die Visualisierung folgt dem Prinzip „overview first, zoom and filter, details on demand“ [CMS99 (S.625)]. Man kann sich bspw. durch einen Filter zunächst den TaskTrace mit einer geringen Modelltiefe ansehen. Die semantische Aggregation bietet dann die Möglichkeit von einer beliebigen Aufgabeninstanz eine detaillierte Ansicht zu bekommen.

6.2.2.4 Abspielen aufgezeichneter Sensordaten

Die bisher vorgestellte Aufbereitung der Sensordaten durch Filterung, Aggregation und Normalisierung erlaubt mit der anschließenden Visualisierung die Darstellung einer Momentaufnahme der Nutzerinteraktion. Darüber hinaus sollen aufgezeichnete Sensordaten aus dem physischen Smart Environment auch über den Zeitverlauf hinweg abgespielt werden.

Abbildung 6.17 zeigt das um diese Funktionalität erweiterte virtuelle Smart Environment. Auf der linken Seite befindet sich die bereits vorgestellte Draufsicht auf den Raum. Auf der rechten Seite oben wird für jeden Nutzer der Fortschritt der Aufgabenausführung innerhalb der animierten Aufgabenmodellinstanzen dargestellt. Rechts in der Mitte befinden sich die Steuerelemente zur Wiedergabe von aufgezeichneten Szenarien. Diese können entweder innerhalb der virtuellen Umgebung auf Basis der entworfenen Modelle erstellt worden sein oder von Sensoren in der physischen Umgebung stammen. Wenn man während des Abspielens selbst Interaktionen durchführt, werden diese aufgezeichnet und beim Speichern den abgespielten Daten hinzugefügt. So lassen sich Annotationen zu bestehenden Daten hinzufügen. Der Zeitstempel kann entsprechend der historischen Zeit generiert werden. Die Geschwindigkeit der Animation ist über einen Schieberegler stufenlos frei wählbar. So können die für die Evaluation weniger interessanten Interaktionsfolgen im Zeitraffer abgespielt werden. Aufgaben, bei denen Nutzer Probleme hatten, können verlangsamt abgespielt werden, um die Interaktionssequenz detailliert analysieren zu können. Als kleinste Einheit lässt sich ein einzelnes Sensordatum wiedergeben, wie bspw. die nächste Position eines Nutzers oder das nächste berührte Gerät, um eine Problemsituation möglichst genau rekonstruieren zu können. Die genannten Parameter zum Abspielen lassen sich auf das Vorwärts- und Rückwärtsabspielen anwenden. Für das Abspielen in Echtzeit können zudem die während der Evaluation aufgezeichneten Videodaten wiedergegeben werden. Durch das Umschalten zwischen den vorhandenen Kameras kann die Interaktion aus verschiedenen Blickwinkeln betrachtet werden.

Neben der Durchführung von Tests in einer Laborumgebung sind Evaluationen im realen Umfeld („in-situ“) wichtig, um authentische Kontextbedingungen sicherzustellen, da diese die Assistenz des Systems als Eingabeparameter direkt beeinflussen. Bei der Evaluation von Daten aus realen Aufzeichnungen ist allerdings der Datenschutz kritisch [NSK+08], da bspw. unternehmensinterne Besprechungen vertraulich bleiben sollen. Die bereitgestellte 2D-Ansicht in Abbildung 6.17 liefert eine anonymisierte Darstellung der anwesenden Personen und ihrer Interaktionen.

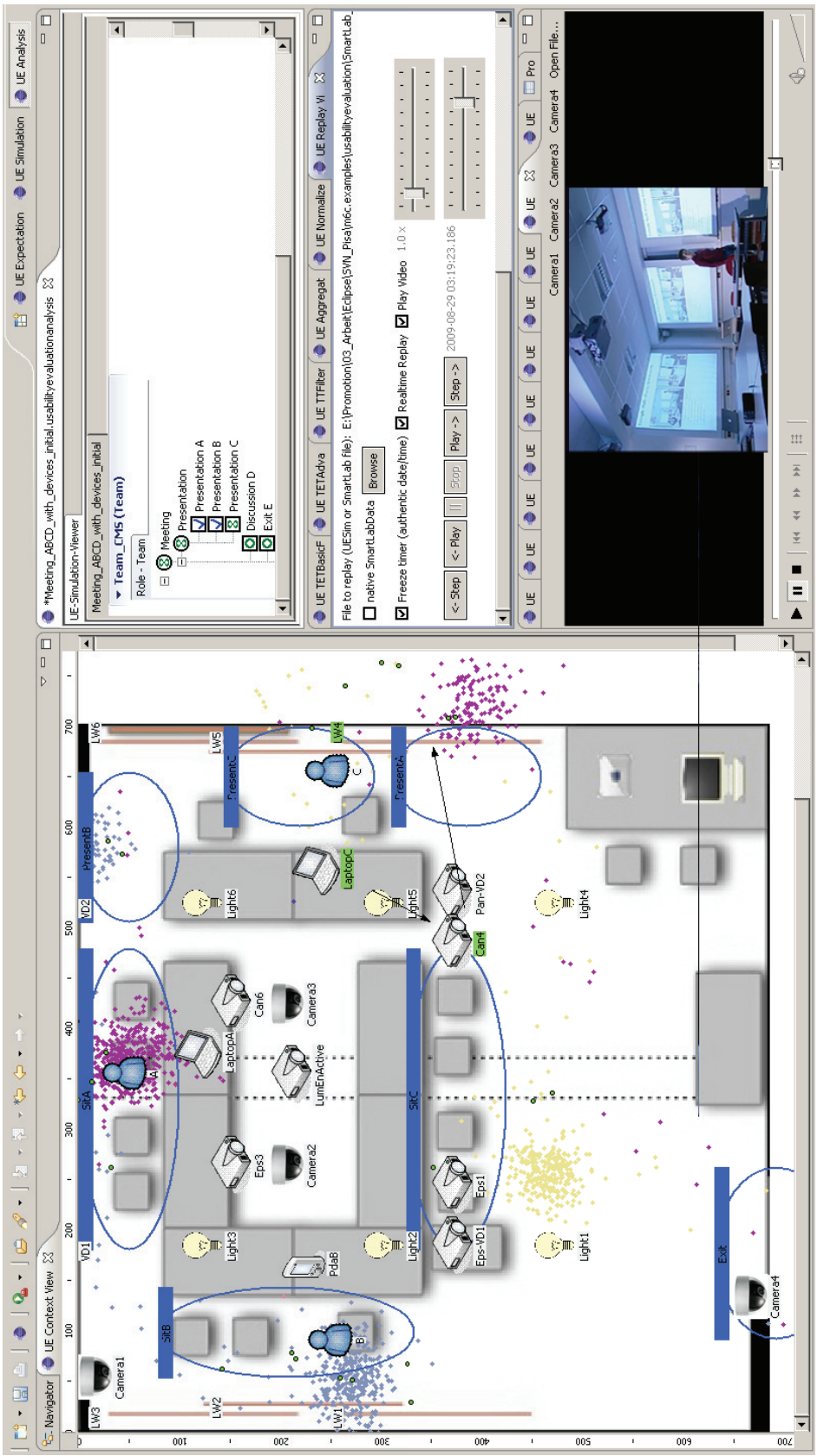

Abbildung 6.17: Visualisierung der Aufgabenausführung im Smart Environment

6.2.2.5 Annotation von Sensordaten

Während bereits aufgezeichnete Sensordaten im virtuellen Smart Environment abgespielt werden, können auch weitere Daten annotiert werden. Dies können textuelle Kommentare sein, die beobachtete Ausführung von Aufgaben oder ergänzende künstliche Sensordaten.

Textuelle Kommentare können an jede ausgeführte Aufgabe annotiert werden. Dadurch lassen sich besondere Beobachtungen während der Evaluation festhalten.

Wenn Daten verschiedener Sensoren vorliegen, wie bspw. Positionsdaten oder verwendete Geräte, aber keine Informationen über die dabei ausgeführten Aufgaben, lassen sich diese annotieren. Auf Basis von Aufgabenmodellen werden Beginn und Ende einzelner Aufgaben festgelegt. Das Ende einer Aufgabe kann dabei als erfolgreich (Ereignis „end“) oder misslungen (Ereignis „crash“) festgelegt werden. Aufgaben die außerhalb der erlaubten zeitlichen Abfolge ausgeführt werden, können trotzdem annotiert werden. Der Aufgabensimulator notiert dabei automatisch, dass eine Abweichung vorliegt.

Bei der Aufzeichnung von Sensordaten während eines Usability-Tests können Sensordaten fehlen, bspw. wenn ein Sensor durch Umgebungseinflüsse gestört ist und für ein Zeitintervall ungenaue Daten liefert oder wenn ein Transponder wegen zu geringen Batteriestandes ausfällt. Um dennoch komplette Datensätze (bspw. als Testdaten für das Maschinenlernen oder für die Evaluation) zu haben, besteht eine Möglichkeit darin, den gesamten Test zu wiederholen, was allerdings mit hohem Aufwand verbunden wäre. In diesen Fällen erlaubt das virtuelle Smart Environment die aufgezeichneten Daten bis zu diesem Zeitpunkt abzuspielen und anschließend künstliche Sensordaten auf der Basis der aufgezeichneten Videos zu erstellen. Während das Video wiedergegeben wird, kann man die gewünschten Interaktionen abkodieren. Dabei ist es zu empfehlen, das Video mehrfach abzuspielen und bei jedem Durchlauf nur einen Sensor einzubeziehen, um iterativ die gewünschten Daten zu erstellen. Dieses Vorgehen lässt sich auch anwenden, wenn bspw. nur eine begrenzte Zahl an UbiSense-Transpondern zur Verfügung steht oder wenn sich im Nachhinein ein Aspekt als besonders interessant herausstellt, der während der Evaluation nicht mit Sensoren erfasst wurde. Im „Smart Environment“-Labor an der Universität Rostock ist die RFID-Erkennung teilweise recht ungenau. Durch die Erstellung künstlicher Sensordaten lassen sich die betreffenden Algorithmen auch mit Daten höherer Genauigkeit testen, wobei die gewünschte Wahrscheinlichkeit für generierte Fehler frei festgelegt werden kann.

6.2.2.6 Vorgehen bei der Evaluation

Die vorgestellten Aspekte der Evaluation eines Smart Enavironments, wie die Spezifikation von Erwartungen, Aufbereitung der Daten, Visualisierung von Nutzerinteraktionen, Animation aufgezeichneter Szenarien und Annotation von Sensordaten, lassen sich jeweils separat oder in Kombination anwenden. Die Wahl der konkreten Techniken hängt bspw. ab von der Fragestellung des Tests, der verfügbaren Ressourcen (Zeit und Budget) und den eigenen Erfahrungen der beteiligten Usability-Experten.

Ein Beispiel soll nachfolgend den Prozess illustrieren. Ausgangssituation ist ein physisches Smart Environment, das implementiert und aufgebaut wurde. Während der Entwicklung wurden bereits die zu Grunde liegenden Modelle evaluiert. Nun soll die Assistenz im Raum mit allen physischen Aspekten durch eine Nutzerstudie in der Laborumgebung evaluiert werden. Dazu werden Testnutzer eingeladen, die jeweils zu erfüllende Aufgaben und persönliche Geräte erhalten, so dass jeder Nutzer eine bestimmte Rolle einnimmt. Währenddessen werden Nutzerinteraktionen als Sensorwerte und Videos aufgezeichnet.

(1) Abweichungen vom Aufgabenmodell

Nachdem der Test durchgeführt worden ist, werden zunächst die von der Intentionserkennung erkannten Ziele analysiert. Diese werden während des Tests an den kooperativen Aufgabensimulator übermittelt und dort zur Animation der Aufgabenmodelle verwendet. Parallel dazu werden die Ereignisse innerhalb des Simulators als TaskEventTrace protokolliert. Wird für eine Aufgabe das Ereignis „start“ oder „end“ geliefert, obwohl dies zu diesem Zeitpunkt laut Spezifikation nicht erlaubt ist, wird der Erfolg als „success = false“ aufgezeichnet. Ein Filtern des TaskEventTrace, nach Ereignissen, die nicht erfolgreich ausgeführt wurden, gibt Aufschluss über Abweichungen der erkannten Aufgabenausführung vom Aufgabenmodell.

Für Abweichungen zwischen Aufgabenmodell und erkannter Aufgabensequenz gibt es verschiedene Ursachen. Dies wird deutlich, wenn man sich die in Abbildung 6.18 dargestellte Verarbeitungsabfolge zur Intentionserkennung ansieht. Ein Fehler kann auf allen Stufen der Verarbeitung auftreten. Zunächst kann der Nutzer eine in dieser Situation unpassende Interaktion gewählt haben. Dies erkennt man durch einen Blick auf die Videoaufzeichnung. Wurde eine zielführende Interaktion ausgeführt, können die betreffenden Sensoren falsche Sensordaten liefern. So kann bspw. eine Lokalisierungssensor einen stark abweichenden Wert liefern. Diese Sensordaten werden im ContextView visualisiert und können mit dem Video verglichen werden. Stimmen diese Werte, kann die Intentionserkennung ein fehlerhaftes Ziel ermittelt haben. Dieses kann im kooperativen Aufgabensimulator eingesehen werden.

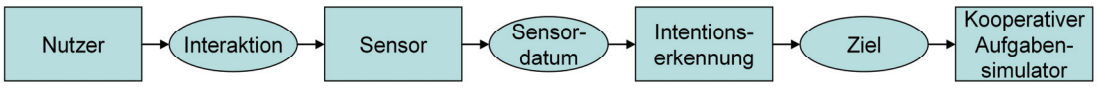

Abbildung 6.18: Interaktionsverarbeitung

Nachdem man den Fehler so eingegrenzt hat, ist die entsprechende Ursache dafür zu finden und zu beheben. Wenn der Nutzer eine andere Interaktion durchführt als im Aufgabenmodell beschrieben, kann dies tatsächlich nicht zielführend sein oder eine bisher nicht modelliert Alternative aufzeigen. Im Falle einer unpassenden Interaktion muss der aktuelle Kontext an Hand des Videomaterials und der Sensordaten detailliert analysiert werden. Führt die Interaktion auf einem alternativen Weg zum Ziel, ist das Aufgabenmodell zu überprüfen und so zu erweitern, dass das beobachtete Szenario ebenfalls unterstützt wird. Wenn die im

ContextView visualisierten Sensordaten von der Beobachtung in der realen Umgebung abweichen, ist der entsprechende Sensor zu untersuchen. So kann bspw. eine neue Kalibrierung oder ein Austausch das Problem beheben. Wenn das von der Intentionserkennung gelieferte Ziel fehlerhaft ist, ist dies zu überprüfen. Das zu Grunde liegende HMM enthält bspw. Informationen darüber, welche Sensordaten in welchem verdeckten Zustand vorliegen können. Entsprechend ist zu prüfen, ob bspw. die beschriebenen Orte, genutzten Geräte und anderen Kontextdaten korrekt sind.

(2) Abweichungen von der Erwartung für den Testfall

Im nächsten Schritt werden Aufgaben analysiert, die zwar dem Aufgabenmodell entsprechend ausgeführt werden dürfen, aber im konkreten Testfall nicht der Erwartung entsprechen. Der kooperative Aufgabenmodellsimulator vergleicht dazu bei jeder ausgeführten Aufgabe, ob diese im aktuellen Kontext begonnen oder beendet werden sollte. Für jedes Ereignis innerhalb des Aufgabensimulators wird eine eventuell vorhandene, spezifizierte Erwartung ausgewertet und das Ergebnis protokolliert. Ein Filtern des TaskEventTrace nach dem Attribut „Expectation“ erlaubt es, Interaktionen zu finden, die von der Erwartung abweichen. Es sollten dabei die Abweichungen zuerst untersucht werden, die in die höchste Fehlerkategorie eingeordnet werden, um Problemkandidaten mit kritischeren Auswirkungen zuerst zu beheben.

Eine Abweichung von der effizienten Bearbeitung der Aufgabe kann auf ein Usability-Problem hinweisen. Es kann aber auch ein äußerer Einfluss vorliegen, der dieses Vorgehen notwendig macht. Fällt bspw. ein Projektor aus, kann das Vortragen unter Benutzung einer Tafel eine Alternative sein. Eine detaillierte Analyse von Video- und Sensoraufzeichnungen gibt Aufschluss darüber, ob ein Kandidat für ein Usability-Problem ein tatsächliches Problem darstellt.

(3) Analyse besonders kurzer und langer Aufgaben

Ein weiterer Indikator für ein Problem während der Testdurchführung sind Aufgaben mit sehr kurzer oder sehr langer Ausführungsdauer. Über Filter lassen sich diese Aufgaben finden. Wenn die Ausführungsdauer der durch die Intentionserkennung ermittelten Aufgaben deutlich kürzer ausfällt als erwartet, kann dies zwei Ursachen haben. Einerseits kann ein Fehler bei der Intentionserkennung vorliegen, der nach kurzer Zeit durch neue Sensordaten korrigiert wird. Andererseits gibt es Fälle, in denen ein Nutzer eine Aufgabe beginnt, feststellt, dass dies nicht zum gewünschten Ziel führt und vorzeitig abbricht.

Fällt die Dauer für eine Aufgabe hingegen deutlich länger aus, können wieder beide Fälle vorliegen: eine Fehlerkennung der Intentionserkennung oder eine Ursache bedingt durch Nutzer. Fehlerkennungen dieser Art traten in praktischen Beispielen selten auf. In der größeren Zahl der Fälle liegt ein anderes Problem vor. Reagiert bspw. ein Gerät anders als erwartet, probieren Nutzer häufig manuelle Einstellungen am Gerät aus, wodurch Kontextinformationen (wie Position und berührte Geräte) gleich blieben und keine neue Intention erkannt wurde.

(4) Kombination mit anderen Verfahren

In den ersten drei Schritten wurden durch Analyse der aufgezeichneten Daten von Testfällen Kandidaten für Problemfälle ermittelt und anschließend die so gefundenen Situationen detailliert untersucht. Durch Heuristiken, wie die Analyse von Aufgabendauern, kann man schnell Kandidaten identifizieren und erste Probleme beheben. Dies ist aber kein Ersatz anderer Evaluationsmethoden. Um die Sicht von Nutzern stärker einzubeziehen, können ergänzend Fragebögen ausgegeben werden, um das subjektive Empfinden zu ermitteln. Bei welchen Aufgaben haben sich Nutzer wohlgefühlt? Wo kam die Assistenz ungelegen? Wo fehlte Assistenz? Oder worin bestanden Verständnisprobleme bei der Arbeit im Smart Environment? Das wiederholte Abspielen des Videomaterials kann eine Hilfe sein, um Nutzer an konkrete Situationen zu erinnern und die identifizierten Problemkandidaten zu analysieren und Verbesserungsvorschläge abzuleiten.

(5) Abspielen der aufgezeichneten Szenarien

Um einen Überblick über den gesamten Ablauf des Tests zu erhalten, können die aufgezeichneten Sensordaten im virtuellen Smart Environment wiedergeben werden. Dadurch können die in den Schritten zuvor quantitativ aus aufgezeichneten Daten ermittelten Kandidaten für Usability-Probleme und die qualitativ von Testnutzern ermittelten Eindrücke analysiert werden. Das interaktive Filtern bspw. nach Nutzern, Geräten und Orten erlaubt die Untersuchung von Problemen und deren Ursachen nach dem Prinzip „overview first, zoom and filter, details on demand“ [CMS99 (S.625)].

6.3 Verbesserung der Usability

6.3.1 Nach Evaluation erster Teilkomponenten

Zur Evaluation der Intentionserkennung werden künstliche Sensordaten erzeugt, die als Eingabe der Algorithmen dienen. Die daraus erkannten Ziele der Nutzer werden anschließend mit den tatsächlichen Zielen verglichen. Gibt es Unterschiede, können die aufgezeichneten Interaktionen vor- und zurückgespult werden, bis die letzte konsistente Situation gefunden ist. Animierte Aufgabenmodelle zeigen den momentanen Stand der Aufgabenausführung und zeigen die annotierten Kontextbedingungen der Aufgaben an. Stellt der evaluierende Experte fehlerhaft modellierte Kontextbedingungen fest, können diese aus dem virtuellen Smart Environment heraus verändert werden. Sind die betreffenden Aufgabenmodelle aktualisiert, kann die Generierung des Quellcodes der Intentionserkennung erneut durchgeführt werden. Ein anschließendes Abspielen der künstlichen Sensordaten gibt die Möglichkeit, den Erfolg der Änderungen zu kontrollieren.

Andere Komponenten des Smart Environments, wie bspw. die Strategieplanung, kann analog mit künstlichen Sensordaten getestet werden.

6.3.2 Nach Evaluation des kompletten physischen Smart Environments

Während eines Nutzertests werden vielfältige Sensordaten und Videos aufgezeichnet. Neben der Evaluation der Modelle können nun auch weitere physische Aspekte, wie visuelle und haptische Eigenschaften analysiert werden. Dabei können sehr unterschiedliche Probleme in allen Bereichen auftreten. Je nach Art des Problems, ist die Behebung individuell sehr verschieden. Zur Bearbeitung der Aufgabenmodelle und HMMs stehen Editoren bereit. Probleme mit konkreten Geräten, wie bspw. Defekte von Leinwänden nach zu starker Beanspruchung oder zu kleine Displayauflösungen, sind jeweils individuell zu beheben. Wie in den anderen Phasen der Entwicklung auch, ist nach jeder Verbesserung und Evaluation zu entscheiden, ob eine weitere Iteration notwendig ist oder die erreichte Qualität den Anforderungen entspricht.

6.4 Informationssystem für Nutzer in Smart Environments

6.4.1 Motivation

Das zu entwickelnde Nutzerinformationssystem soll sich drei Problemen bei der Interaktion in Smart Environments widmen:

(1) Smart Environments nehmen Nutzerverhalten über Sensoren wahr, versuchen daraus Nutzerintentionen abzuleiten und diese bestmöglich durch Assistenz zu unterstützen. Wenn es dabei in Ausnahmefällen zu Fehlinterpretationen und in Folge dessen zu einer ungeeigneten Assistenz kommt, kann dies zu Problemen bei der Benutzung des Systems führen. Cook und Das stellen fest: „If the environment makes a wrong automation decision, it will be at best an annoyance and at worst a hindrance to accomplishing everyday tasks.“ [RYC+07].

(2) Erkennt das Smart Environment die Nutzerintentionen richtig und beginnt mit der Ansteuerung der Geräte, ist möglicherweise die beginnende Änderung des Systemzustandes nicht sofort für alle Nutzer wahrnehmbar. Einige Nutzer sind sich dann nicht sicher, ob das System die geeignete Assistenz vorbereitet und damit ein kurzes Abwarten lohnend ist. Norman [Nor02 (S.49ff)] spricht vom „Gulf of Evaluation“ und stellt die Frage, ob ein Gerät eine physische Repräsentation seines Zustandes bietet und ob diese direkt für Nutzer interpretierbar ist. Eine Rückmeldung kann reichen von einer einfachen, akustischen Bestätigung, dass das System noch arbeitet, bis hin zu einer präzisen Information über den aktuellen Zustand.

(3) Darüber hinaus haben Ziefle et al. [ZAB06] für mobile Geräte untersucht, welche Bedeutung mentale Modelle haben. Die Studie zeigte, dass Nutzer mit einem guten mentalen Modell Aufgaben schneller und zielstrebiger lösen konnten, während ein falsches mentales Modell hinderlich war.

Das Nutzerinformationssystem soll den Zustand der Intentionserkennung des Raumes für Nutzer transparent machen, so dass die Nutzer in vereinfachter Form sehen, was der Raum über sie „denkt“. Die Nutzer sollen dadurch Hilfe bekommen, den „Gulf of Evaluation“ zu überwinden, indem die vom Raum erkannte Aufgabenausführung grafisch dargestellt wird.

Wenn die Erkennung mit der Realität übereinstimmt, kann dies beruhigend zur Kenntnis genommen werden. Im Falle einer fehlerhaften Erkennung soll es möglich sein, dass der Nutzer dies dem System mitteilt, damit die Erkennung korrigiert werden kann und eine passende Assistenz erfolgen kann. Gleichzeitig soll der Nutzer über die Darstellung eine Hilfe bei dem Aufbau eines mentalen Modells bekommen.

6.4.2 Konzept

Während Nutzer im Smart Environment interagieren, werden Sensordaten aufgezeichnet und die Intentionserkennung versucht daraus ausgeführte Aufgaben zu erkennen. Dadurch entsteht eine Sequenz ausgeführter Aufgaben. Diese soll für die Nutzer aufbereitet und visualisiert werden, so dass diese einerseits bei einer unerwarteten Assistenz des Raumes kontrollieren können, ob überhaupt die richtige Tätigkeit erkannt und unterstützt wurde. Andererseits bekommen Nutzer, die bei einer Tätigkeit durch eine andere Nebentätigkeit (bspw. einen Telefonanruf) unterbrochen wurden, eine Orientierung über den aktuellen Fortschritt.

Ein mögliches Problem dabei ist allerdings, dass solch eine Sequenz mit jeder neuen Aufgabe länger und unübersichtlicher wird. Eine Lösung besteht in der Anzeige der *n* letzten Aufgaben, was allerdings bedeutet, dass *n* Aufgaben mit gleichem, sehr hohem Detailgrad angezeigt werden und alles davor komplett entfällt. Eine bessere Lösung bietet die Anwendung der in Kapitel 6.2 entwickelten Fokus- und Kontexttechnik zur Aggregation eines TaskTrace. Hierbei wird ähnlich einer optischen Linse ein Fokus auf die aktuelle Aufgabe gelegt. Je weiter eine Aufgabe verglichen mit dem Fokus in der Vergangenheit liegt, mit desto geringerem Detailgrad wird sie dargestellt. Es werden also alle Aufgaben angezeigt, während eine kompakte, übersichtliche Darstellung über die Adaption des Detailgrades erreicht wird.

Die Darstellung soll auf dem persönlichen Gerät der Nutzer möglich sein, um die eigene Aufgabenausführung der einzelnen Rollen anzeigen zu können.

6.4.3 Implementierter Prototyp

Der implementierte Prototyp wurde als Eclipse-Plugin realisiert und ist in Abbildung 6.19 dargestellt. Im Beispiel wurde das Aufgabenmodell des Vorsitzenden einer Besprechung verwendet, wie es in Abbildung 6.12 gezeigt wurde. Der Vorsitzende kann sich damit über den aktuellen Verlauf der Besprechung informieren oder prüfen, ob das Smart Environment die ausgeführten Aufgaben richtig erkannt hat.

Dargestellt ist die Sequenz der ausgeführten Aufgaben entlang einer vertikalen Zeitleiste von oben beginnend. Jede Aufgabe ist als Rechteck dargestellt und die aktuell laufende Aufgabe „thank_the_presenter“ ist farblich gelb hervorgehoben. Bereits ausgeführte oder derzeit laufende Aufgaben sind von einer durchgezogenen Linie umschlossen. Die in der Vergangenheit liegenden Aufgaben „introduce“ und „introduce_the_presenter“ sind keine Basisaufgaben des Aufgabenmodells, sondern zum Einsparen von Platz auf dem Display aus mehreren Teilaufgaben aggregiert. Die laut Modell nächsten möglichen Aufgaben „introdu-

ce_the_presenter“ und „open_discussion“ sind durch gestrichelte Rechtecke dargestellt und geben eine Orientierung über nächste vom System durch Assistenz unterstützte Aufgaben. Im physischen Smart Environment kann der Nutzer beliebige Aufgaben ausführen, erhält aber nur bei den dargestellten Assistenz.

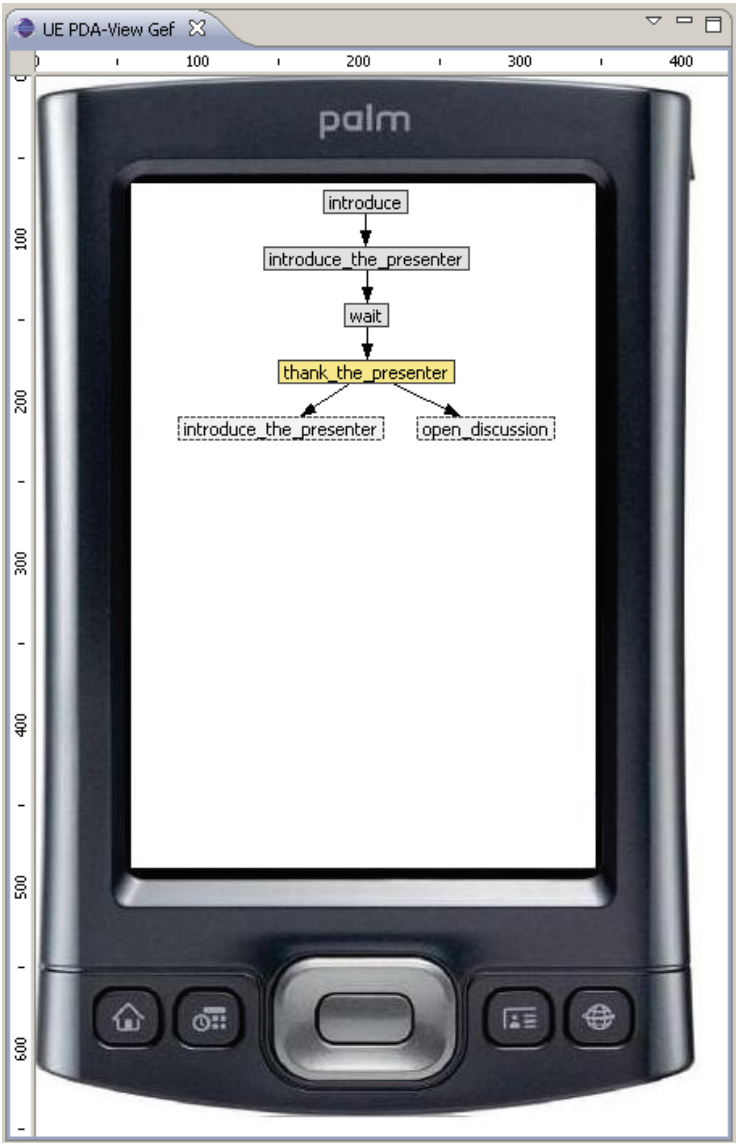

Abbildung 6.19: Prototyp für die Nutzerführung

Der Aggregationsalgorithmus wurde so adaptiert, dass als Parameter die je nach Ausgabegerät maximal vertikal darstellbare Anzahl von Aufgaben im Display dient.

Das Informationssystem wurde bisher funktional getestet. Eine künftige Nutzerstudie wäre wünschenswert, um die Anwendung entsprechend der Nutzerbedürfnisse anzupassen.

6.5 Zusammenfassung

Dem Prinzip der zielbasierten Interaktion folgend nimmt das Smart Environment Handlungen von Nutzern über Sensoren wahr, während eine Intentionserkennung daraus Ziele vorhersagt und eine Strategieplanung schließlich Aktoren steuert, um Assistenz anzubieten [AE06]. Die dafür notwendige Intentionserkennung kann modellbasiert entwickelt werden. Dazu werden die während der Designphase entwickelten und evaluierten Aufgabenmodelle in statistische Modelle und anschließend in Programmcode transformiert [GFF+07, BPK+09]. Währenddessen können Entwickler eine manuelle Feinjustierung von Parametern vornehmen.

Entwickelte Softwarekomponenten, wie Intentionserkennung oder Strategieplanung, können nicht losgelöst vom Anwendungskontext getestet werden. Daher wurde ein virtuelles Smart Environment entwickelt, das das Generieren künstlicher Sensordaten erlaubt. Diese werden als Eingabe verwendet und die Ergebnisse werden in der virtuellen Umgebung dargestellt. So kann bei der Intentionserkennung getestet werden, ob bei der Animation eines Szenarios unter Einbezug relevanter Orte, verwendeter Geräte und Gegenstände, sowie Stati von Geräten die erwarteten Ziele erkannt werden. Analog dazu werden zur Evaluation der Strategieplanung Ziele vorgegeben und die angebotene Assistenz wird dargestellt.

Nachdem das reale Smart Environment komplett eingerichtet ist, können Nutzertests durchgeführt werden, die alle Aspekte der Umgebung einbeziehen. Durch Aufzeichnungen von Sensor- und Videodaten entstehen große Datenmengen. Diese werden häufig manuell analysiert, um Nutzerverhalten zu identifizieren, das von der Erwartung an eine effiziente Aufgabenausführung abweicht. Um diesen aufwendigen Prozess teilweise zu automatisieren, wird eine Methode erläutert, um implizit bei Usability-Experten vorhandenes Wissen über Erwartungen explizit zu spezifizieren. In der Literatur zur Evaluation von Software für Einzelplatzrechner finden sich ebenfalls Ansätze zum Vergleich von tatsächlichem und erwartetem Verhalten. So visualisieren UsAGE [UW95] und Quip [HL99] Unterschiede zwischen Verhalten eines Testnutzers und eines Experten, um preiswert Kandidaten für Usability-Probleme zu identifizieren. Ebba [Bat95] hebt die Einschränkung auf den Vergleich mit einem einzelnen erwünschten Pfad auf und vergleicht mit einer allgemeineren Beschreibung, die allerdings sehr systemnah ist. In dieser Arbeit erfolgt die Spezifikation von Erwartungen in einer allgemeineren, aufgabenorientierten Weise. Zudem ist die Spezifikation auf Smart Environments zugeschnitten und erlaubt auch die Beschreibung von Abhängigkeiten zwischen Aufgaben verschiedener kooperierender Nutzer.

Zur Untersuchung der aufgezeichneten Daten wird eine aufgabenbasierte Analyse vorgeschlagen. Durch die entwickelten Werkzeuge können Aufgabensequenzen gefiltert, mit einer Fokus- und Kontexttechnik aggregiert, normalisiert und schließlich entlang einer Zeitleiste visualisiert werden. Die vergleichende Visualisierung der Aufgabenausführung mehrerer Nutzer wurde bereits von Maly et al. [MS06] vorgeschlagen, allerdings ohne dabei Aufgabenmodelle zu verwenden. Diese Arbeit greift die Idee von [MS06] auf, geht mit der Normalisierung einen Schritt weiter und kann durch die Verwendung von Aufgabenmodellen zusätzlich die Aggregation umsetzen.

Das entwickelte Smart Environment visualisiert neben den Aufgabensequenzen auch die Bewegungspfade der Nutzer. In der Literatur finden sich zahlreiche Arbeiten zur Darstellung ortsbezogener Daten. So färben Mikovec et al. [MMS+07] besonders häufig frequentierte Orte ein. Ieronutti et al. [IRC05] unterscheiden zwischen Einzel- und Mehrnutzeransichten und zeichnen bevorzugte Bewegungsrichtungen, was vor allem bei einem Netz alternativer Wege interessant ist. Holm et al. [HPS+02] zeigen ergänzende Daten über Kopfbewegung und medizinische Werte an (bspw. Puls). Die vorlegende Arbeit konzentriert sich auf die Visualisierung der reinen Bewegungspfade mit wählbarem Detailgrad. Die besondere Funktionalität besteht in der Verbindung von Orts- und Aufgabendaten, die es nach Auswahl von Wegmarken erlaubt, Informationen über dort durchgeführte Aufgaben zu bekommen.

Das virtuelle Smart Environment kann aufgezeichnete Sensordaten zu einem späteren Zeitpunkt wiedergeben. Dabei werden Nutzerinteraktionen als Draufsicht dargestellt, während der Fortschritt der Aufgabenausführung im kooperativen Aufgabenmodell abgebildet wird.

Ein entwickeltes Werkzeug erlaubt zudem die Annotation von aufgezeichneten Daten. Dies kann bspw. dann hilfreich sein, wenn zusätzliche Beobachtungen festgehalten werden sollen oder wenn während der Aufzeichnung Daten eines Sensors zeitweise fehlen.

Ein Nutzerinformationssystem verwendet den entwickelten Algorithmus zur Aggregation von TaskTraces, um die Arbeitsweise des Assistenzsystems transparenter zu machen.

In diesem Kapitel wurde das entwickelte virtuelle Smart Environment umfassend vorgestellt. In der Literatur finden sich eine Reihe weiterer Projekte, die teilweise ähnliche Ziele verfolgen. Ein Überblick soll den eigenen Ansatz einordnen.

Siafu [MN06] dient sowohl der Darstellung von aufgezeichneten Sensordaten, als auch zum Generieren von Kontextdaten für das Maschinenlernen. Dazu kann man für jeden Nutzer das Verhalten durch Quellcode beschreiben und anschließend in der Simulation frei konfigurierbare Kontextdaten generieren lassen. Das interaktive Animieren von Nutzerverhalten über die grafische Benutzungsschnittstelle ist allerdings nicht möglich.

UbiWise („UBIquitous Wireless Infrastructure Simulation Environment“) [BV03], Tatus [One04] und UbiReal („UBIquitous application aimulator with REAListic environments“) [NYT+06] sind sehr technisch orientiert und an der Entwicklung von Produkten ausgerichtet. Die Simulatoren dienen als Testumgebung für die zu entwickelnde Hard- und Software. So schließt UbiReal auch einen Netzwerksimulator und einen Simulator physischer Eigenschaften ein. Die vorliegende Arbeit legt mehr Wert auf die Untersuchung der Interaktion zwischen Mensch und Maschine, besonders darauf, wie Nutzer diese empfinden. UbiWise und Tatus verwenden für die Interaktion mit der Umgebung und für die Visualisierung Spiele-Engines, die eine Ich-Perspektive (die des Spielers) verfolgen und damit nur die Steuerung eines einzelnen Nutzers erlauben. Ein Wechsel des Nutzers ist nicht nahtlos möglich.

3DSim („3D-based rapid prototyping and SIMulation environment system“) [SH05] und APEX („Agile Prototyping for usEr eXperience“) [SCH09] sind dem Fokus dieser Arbeit näher. Beide bieten eine virtuelle Umgebung, die eine Interaktion von Nutzern erlaubt, wobei Systemreaktionen in der virtuellen Umgebung dargestellt werden. APEX verwendet

CPNs („colored petri nets“), um das Systemverhalten zu beschreiben. Die vorliegende Arbeit verwendet kooperative Aufgabenmodelle, um den Nutzer stärker in den Vordergrund zu stellen. APEX verwendet wie auch UbiWise und Tatus eine externe Simulationsumgebung, die nur eine Ich-Perspektive bietet und kein Editieren der Umgebung nach Start der Simulation, um bspw. relevante Orte nachträglich in die Simulation einzufügen oder Gegenstände zu verändern. Im Gegensatz dazu erlaubt das in dieser Arbeit entwickelte virtuelle Smart Environment in der Designphase den Grundriss während einer Experteneva-luation zu durchlaufen und Positionen von Möbeln und Ortsmarkierungen interaktiv anzupassen.

Mundo [AM07] ist eine Entwicklungsmethode für die rasche Entwicklung von Smart Environments. Zur Fehlersuche implementierter Komponenten werden Werkzeuge bereitge-stellt. Daneben besteht auch die Möglichkeit als Ersatz für ein reales Positionierungssystem entsprechende Sensordaten künstlich zu generieren und Positionsdaten grafisch darzustellen. Mundo ist vor allem auf die Entwicklung ausgelegt und bietet sehr technisch orientierte Testmethoden.

ResiSim [RYC+07] erlaubt die Visualisierung des aktuellen Umgebungszustandes im Smart Environment und die Steuerung der darin befindlichen Geräte. Zudem zeigt ResiSim den Umgebungszustand auf mobilen Geräten von Nutzern an, so dass die Nutzer selbst Aktionen des Assistenzsystems rückgängig machen und bewerten können. Die Bewertung des Systemverhaltens als positiv oder negativ wird für das Maschinenlernen genutzt, um die Assistenz besser an Nutzerwünschen auszurichten. ResiSim wird zudem während des Designs genutzt, um Assistenzfunktionen zu testen.

MoDie [LBV+06] verwendet, so wie die vorliegende Arbeit, ebenfalls Aufgabenmodelle, während eine Ontologie weitere Kontextinformationen abbildet. Allerdings ist MoDie auf das Testen von modellbasiert entwickelten Benutzungsschnittstellen beschränkt, was bei der vorliegenden Arbeit nur einen Teilaspekt darstellt.

USEd („User Scenario EDitor“) [MCK+08] verfolgt explizit das Ziel der Usability-Evaluation. Dazu bietet es die Möglichkeit, Nutzerverhalten grafisch zu animieren und dabei künstliche Sensordaten zu erzeugen, jedoch ohne dabei Sensorfehler zu erzeugen. Aufgaben können als begonnen oder beendet gekennzeichnet werden, wobei aber keine Aufgabenmo-delle zum Einsatz kommen. Das Abspielen von Daten erfolgt im verwandten Werkzeug SitCom („SITuation COMposer for smart environments“) [FCK07, CKF09], dass die Intentionserkennung gewährleistet. Hierbei werden wie in dem dieser Arbeit zu Grunde gelegten Ansatz von Burkhardt et al. [BPK+09] ebenfalls statistische Modelle eingesetzt.

Kapitel 7

Evaluation

Nachdem das Vorgehensmodell und das virtuelle Smart Environment in ihrer Funktionsweise erläutert wurden, sollen in diesem Kapitel erste Erfahrungen mit deren Anwendung berichtet werden. Eine Nutzerstudie wird durchgeführt, um zu untersuchen, wie gut Nutzer mit dem virtuellen Smart Environment umgehen können. Die Probanden bekommen dazu ein kooperatives Aufgabenmodell der Designphase und sollen dies interaktiv durchlaufen, um Abweichungen von den Anforderungen zu finden und zu beheben. In einem weiteren Experiment wird eine Komponente der Intentionserkennung während der frühen Implementationsphase evaluiert. Dazu werden künstliche Sensordaten mit der zuvor evaluierten virtuellen Umgebung generiert. Die Ergebnisse der Vorhersage werden untersucht.

7.1 Evaluation des Designs

7.1.1 Zielstellung

Für den praktischen Einsatz des entwickelten Vorgehensmodells wurde als Werkzeugunterstützung das virtuelle Smart Environment vorgestellt, das in einer Benutzungsschnittstelle vier Funktionalitäten vereint:

- F1: Animation von kooperativen Aufgabenmodellen („Walkthrough“)
- F2: Steuerung von „Wizard of Oz“-Experimenten
- F3: Generierung von künstlichen Sensordaten
- F4: Visualisierung von aufgezeichneten Daten aus Nutzertests

Für Anwendungen in allen vier Bereichen ist es wichtig, dass Nutzer sowohl die grafische Darstellung virtueller Objekte und Interaktionen gut interpretieren können, als auch gut mit der Steuerung der virtuellen Umgebung umgehen können. Folgende vier Thesen sollen durch eine Nutzerstudie untersucht werden:

These 1: Nutzer können sich in die Darstellung des virtuellen Smart Environments hineinversetzen.

Personen, Geräte und Gegenstände werden im virtuellen Smart Environment als vereinfachtes Abbild der Realität dargestellt. Die richtige Interpretation ist wichtig, um beim Abspielen von Sensordaten (F4) eine dargestellte Situation beurteilen zu können und um bei der Animation von Nutzerinteraktionen (F1, F3) und der Steuerung von Geräten (F4) die richtige Option auszuwählen. Je besser sich Nutzer unter den grafischen Elementen eine reale Situation vorstellen können, desto näher kommen die virtuellen an die realen Interaktionen.

These 2: Nutzer sind in der Lage, Interaktionen nach realem Vorbild zu animieren.

Für die Funktionalitäten F1 – F3 ist die Animation von Interaktionen bzw. die Steuerung von Geräten wichtig. Wenn ein Nutzer weiß, was er in der virtuellen Umgebung steuern möchte, soll es möglichst intuitiv und einfach möglich sein, dies umzusetzen.

These 3: Fehler im kooperativen Aufgabenmodell lassen sich durch Animation finden.

Mit dem virtuellen Smart Environment lassen sich verschiedene Artefakte evaluieren. Hier sollen speziell kooperative Aufgabenmodelle untersucht werden, die während der Phasen der Anforderungsanalyse und des Designs entstehen. Wenn Nutzer sich in die dargestellte virtuelle Umgebung hineinversetzen, soll darüber hinaus untersucht werden, wie gut Abweichungen zwischen gewünschter und tatsächlicher Funktionalität gefunden werden können.

These 4: Fehler im kooperativen Aufgabenmodell lassen sich direkt aus dem virtuellen Smart Environment heraus einfach beheben.

Wenn ein Fehler gefunden wird, soll dieser nicht lediglich dokumentiert werden, um den Fehler dann in einem separaten Entwicklungswerkzeug zu rekonstruieren, sondern es soll möglich sein, eine Teilmenge möglicher Fehler direkt in der virtuellen Umgebung zu beheben. Eine Untersuchung soll ergeben, wie leicht Fehler bei Vorbedingungen und Effekten im kooperativen Aufgabenmodell behoben werden können.

7.1.2 Testaufgabe

Zur Untersuchung der genannten Fragestellungen soll eine Nutzerstudie durchgeführt werden. Jeder der Probanden versetzt sich dabei in die Rolle eines Interaktionsdesigners bzw. Usability-Experten, der ein während der Designphase entworfenes kooperatives Aufgabenmodell vorgelegt bekommt. Der Proband soll prüfen, ob das Modell die erhobenen Anforderungen aus der Phase der Anforderungsanalyse erfüllt und bei Abweichungen die entsprechenden Fehler behebt.

Dazu wurde ein konkretes Szenario (vgl. Anhang A2) mit Nutzerinteraktionen entworfen, das die Ausführung einer Folge von Aufgaben im Smart Environment beschreibt und jeweils die erwartete Assistenz des Raumes nennt. Angelehnt an das Beispiel in Kapitel 3.4 wird eine Besprechung mit drei Teilnehmern durchgeführt. Die drei Personen betreten zunächst den Raum, wobei jeder ein persönliches Gerät bei sich trägt, auf dem sich Folien für einen Vortrag befinden. Nachdem sich alle gesetzt haben, tragen die Anwesenden nacheinander vor. Alle Vorträge laufen analog ab: Nachdem der Vortragende mit seinem mobilen Gerät vor das Auditorium tritt und die Präsentationsfernbedienung in die Hand nimmt, richtet die Rauminfrastruktur eine Videoverbindung vom Gerät des Vortragenden zu einem Projektor ein, der dann gedreht wird, bis er auf eine Leinwand neben dem Vortragenden projiziert. Die entsprechende Leinwand wird währenddessen heruntergefahren. Diese Assistenzfunktionen erfolgen später im komplett eingerichteten Smart Environment automatisch auf Basis der Intentionserkennung und der Strategieplanung. In dieser Studie legt der Proband den Wechsel von einer Aufgabe zur nächsten manuell fest, wobei das System prüft, ob alle Vorbedingungen erfüllt sind. Diese Vorbedingungen werden später zu Auslösern („Triggern“) der Assistenz transformiert und in der Intentionserkennung verwendet.

Das den Probanden vorliegende kooperative Aufgabenmodell enthält Fehler, liefert also nicht die im Szenario gewünschte Assistenz. Dazu wurden Vorbedingungen und Effekte präpariert. Zur Illustration sollen ein paar Beispiele genannt werden:

- Bei Beginn eines Vortrages erfolgt keinerlei Assistenz, da als Vorbedingung ein Aufenthaltsort des Nutzers falsch definiert wurde.
- Ein Vortragender tritt mit seinem persönlichen Gerät vor das Auditorium. Auf der Leinwand neben ihm erscheinen allerdings nicht die Folien von seinem Gerät, sondern der Bildschirminhalt eines Zuhörers aus dem Publikum.
- Die Folien eines Vortragenden werden versehentlich direkt auf die Wand projiziert, da die falsche Leinwand heruntergefahren wurde.

Ergänzend zum kooperativen Aufgabenmodell wurden für den Testfall Erwartungen (vgl. Kapitel 6.2.2.1) festgelegt, die während der Aufgabenausführung geprüft werden. Damit sollen grobe Abweichungen der Nutzerinteraktionen bzw. der Assistenz vom erwarteten Verhalten automatisch identifiziert und gemeldet werden, so dass der Proband diesen Kandidaten für ein Problem näher untersuchen kann. Beispiele sind:

- Bei jedem Vortragenden sollen Folien vom persönlichen Gerät geladen werden. Die Videoverbindung soll also dort beginnen.
- Bei jedem Vortragenden sollen die Folien nach einer beliebigen Anzahl von Videoverbindungen (bspw. über mehrere Rechner, die das Bild weiterleiten oder konvertieren) auf einer Leinwand angezeigt werden.
- Bei Verlassen des Raumes soll jeder sein persönliches Gerät wieder bei sich haben.

7.1.3 Durchführung der Evaluation

Vor dem Test bekam jeder Proband einen Fragebogen zu persönlichen Angaben und Vorwissen im Bezug auf die zu bearbeitende Testaufgabe. Anschließend bekam jeder die gedruckte Aufgabenstellung und das zu animierende Szenario (Anhang A2). Da keiner der Probanden zuvor mit dem virtuellen Smart Environment gearbeitet hat, wurde zunächst kurz die Bedienung erläutert. In immer gleicher Folge wurden die einzelnen Funktionalitäten kurz gezeigt und es bestand die Gelegenheit, Fragen zu stellen. Danach begannen die Probanden mit der Bearbeitung der Testaufgabe, während das virtuelle Smart Environment die Interaktionen in der virtuellen Umgebung aufzeichnete und der Bildschirminhalt als Video aufgezeichnet wurde. Obwohl es von den Probanden nicht ausdrücklich erwartet wurde, begannen viele nach kurzer Eingewöhnungszeit mit dem Werkzeug ihr Vorgehen zu kommentieren. Die Kommentare reichten von Freude über behobene Fehler bis zu konkreten Verbesserungsvorschlägen. Kommentare der Probanden und Beobachtungen während des Tests wurden protokolliert. Nach dem Test war ein Fragebogen über die Erfahrungen bei der Lösung der Aufgabe auszufüllen und viele Probanden nutzten die Gelegenheit, weitere Eindrücke zu berichten. Da einige Probanden gleichzeitig während der täglichen Arbeit mit Smart Environments zu tun haben, nutzten manche die Diskussion, um zu erfahren welche Funktionalitäten das Werkzeug über die im Test verwendeten hinaus bietet und inwiefern Anknüpfungspunkte für weitere Arbeiten bestehen. Die Tests dauerten jeweils 30-45 Minuten.

7.1.4 Auswertung

An der Nutzerstudie nahmen 24 Mitarbeiter der Universität Rostock von 7 verschiedenen Lehrstühlen teil. So waren bspw. Mitarbeiter aus den Bereichen Usability, Softwareentwicklung, Computergrafik und Ubiquitous Computing vertreten. Die Probanden hatten einen sehr unterschiedlichen Bezug zum „Smart Environment“-Labor, dessen virtuelles Pendant im Test verwendet wurde: 5 Probanden kannten den Raum nicht, 5 hatten ihn lediglich betreten, 8 hatten ihn in Aktion gesehen und 6 hatten dort bereits selbst etwas installiert. Alle Probanden waren mit Programmierung vertraut. Im Bezug auf Usability hatten 3 fundierte Kenntnisse. Die Kenntnisse im Bereich Aufgabenmodellierung waren sehr verschieden. Ein Aufgabenmodell gesehen haben 9 Probanden während der täglichen Arbeit, 8 kennen sie prinzipiell und 7 haben noch nicht davon gehört. Aufgabemodelle erstellt haben 7 Probanden, während 17 dazu noch keine Gelegenheit hatten.

Die eingeladenen Probanden hatten unterschiedliche fachliche Hintergründe und damit verschiedene Perspektiven. So waren die Kommentare eines Probanden aus der Computergrafik zur Art und Weise der Informationsdarstellung sehr aufschlussreich, während sich Probanden aus dem Bereich Ubiquitous Computing zu den einbezogenen Parametern der Geräte äußerten. Probanden aus dem Bereich Usability konnten Hinweise zu der Praktikabilität für die Durchführung von Evaluationen geben. Für weiterführende Studien wäre es interessant auch Usability-Experten aus der Wirtschaft einzubeziehen.

So konnten Hinweise aus verschiedenen Perspektiven gewonnen werden, die anschließend in die Verbesserung des Werkzeuges einfließen konnten. Darüber hinaus konnten Erfahrungen gewonnen werden, inwiefern verschiedene Kenntnisse den Umgang erleichtern.

Im Folgenden sollen die eingangs formulierten Thesen untersucht werden:

These 1: Nutzer können sich in die Darstellung des virtuellen SE hineinversetzen.

Alle Probanden konnten die gestellten Aufgaben lösen und damit zeigen, dass sie sich in das virtuelle Smart Environment hineinversetzen können. Sie konnten die virtuellen Entitäten und die realen gut miteinander in Verbindung bringen. So legten mehrere Probanden einen Laptop ziemlich am Rande eines Tisches ab, schoben ihn dann in die Mitte des Tisches und äußerten: „damit er nicht runterfällt“ oder „muss ja ordentlich aussehen“. Die Aufgabenstellung enthielt Freiheitsgrade und schrieb nicht vor, auf welchem Tisch ein Gerät abzulegen war oder wo genau auf einem Tisch es liegen soll. So wurden von den Probanden verschiedene Tische gewählt. Einige legten Laptop oder PDA während der Präsentation auf das Pult vor dem Auditorium, andere entschieden sich für das Ablegen auf einem der vorderen Tische, an dem gerade kein Zuhörer saß oder auf einem Beistelltisch an der Seite. Nicht alle dieser Möglichkeiten wurden bei der Modellierung des Testfalles in Betracht gezogen, erschienen aber für reale Situationen plausibel.

Einem Probanden gelang es darüber hinaus ein zusätzliches Usability-Problem zu entdecken, das bei Erstellung des Testfalles nicht beabsichtigt war. Während der Durchführung der Aufgabe „Diskussion“ wurden zur besseren Übersicht noch einmal die Folien mit dem Fazit aller Vortragenden auf verschiedenen Leinwänden gezeigt. Die Folie von Person B wurde dabei auf der Leinwand hinter dem Rücken von Person A angezeigt, wodurch A die Folie von B nicht sehen konnte. Als Lösung schlug der Proband eine sinnvollere Verteilung vor, so dass jeder Teilnehmer alle anderen Folien direkt sehen kann[3]. Der betreffende Proband hat das „Smart Environment“-Labor erst ein Mal während einer anderen Studie ein paar Monate zuvor betreten und damit nur einen geringen Bezug zum realen Labor. Dies kann als Indiz gewertet werden, dass die Darstellung des virtuellen Smart Environments intuitiv genug ist, um sich die animierten Situationen plastisch vorstellen zu können.

These 2: Nutzer sind in der Lage, Interaktionen nach realem Vorbild zu animieren.

Alle Probanden waren in der Lage, die Interaktionen des gegebenen Testszenarios vollständig und in der vorgesehenen Reihenfolge durchzuführen. Neben vielen sehr realistischen Interaktionen in der virtuellen Umgebung, führten Probanden auch vereinzelt Interaktionen aus, die in der Realität nicht möglich sind. So war bei der Animation durch Probanden vereinzelt zu beobachten, dass sie einen Vortragenden nach erfolgreicher Präsentation zurück auf den Platz gehen ließen und versehentlich die Fernbedienung zum Folienweiterschalten mitnahmen, obwohl der nächste Vortragende diese ebenfalls benötigen würde. Um dieses Versehen auszugleichen würde in der Realität der Vortragende nach vorn zum Pult

[3] Anmerkung: Ein Algorithmus, der Dokumente entsprechend den Präferenzen der Nutzer auf vorhandene Displays verteilt, wird in [HK07, Hei09] diskutiert und evaluiert.

gehen und das Gerät dem nächsten Vortragenden überreichen. Manche Probanden entschieden sich aber dafür, die Fernbedienung vom Sitzplatz aus direkt auf dem mehrere Meter entfernten Pult abzulegen. Mehrere Probanden kommentierten ihre Handlung schmunzelnd mit: „Jetzt schmeiß' ich das Gerät mal da hin." oder fragten: „Darf ich das jetzt einfach mal da hinten ablegen?". Dies zeigt, dass die Probanden sich der unrealistischen Interaktion häufig durchaus bewusst waren, aber an Stellen, die für den konkreten Testfall unrelevant waren, Vereinfachungen vornahmen. Für Testfälle, in denen eine sehr akkurate Durchführung der Interaktionen im virtuellen Smart Environment notwendig ist, sollten Nutzer vorab ausdrücklich darauf hingewiesen werden.

Während die Probanden Personen virtuell bewegten und ihnen Geräte in die Hand gaben, äußerten mehrere Probanden, dass ihnen die Arbeit mit der virtuellen Umgebung Spaß macht. Sie fassten die Testaufgabe nicht als eine notwendige Arbeit auf, die erledigt werden muss, sondern eher als spielerische Betätigung Figuren zu bewegen und mit ihnen in der virtuellen Umgebung etwas zu unternehmen.

Somit konnte gezeigt werden, dass die Probanden sich gut in die virtuelle Umgebung hineinversetzen konnten. Dort, wo unrealistische Interaktionen beobachtet werden konnten, waren viele Probanden sich dessen bewusst und äußerten, dass sie bei Testfällen die dies verlangen, auf Vereinfachungen bei der Animation verzichten würden.

These 3: Fehler im kooperativen Aufgabenmodell lassen sich durch Animation finden.

Um einen Fehler zu finden, muss zunächst festgestellt werden, in welcher Situation die dargebotene Assistenz von der gewünschten abweicht und anschließend muss die konkrete Ursache in der Spezifikation des kooperativen Aufgabenmodells gefunden werden.

Die Erkennung, dass die dargebotene Assistenz von der gewünschten abweicht, bewältigten die Probanden gut. Die Probanden verwendeten sowohl die grafische Prüfung im virtuellen Smart Environment, als auch die textbasierte Prüfung des generierten Protokolls. Im Protokoll wurden alle ausgeführten Aufgaben aufgezeichnet. Falls eine Vorbedingung für eine Aufgabe nicht erfüllt war, wurde dies angezeigt. Falls ein Effekt eine Assistenzfunktion auslöste, die gegen die spezifizierte Erwartung verstieß, wurde dies ebenfalls angezeigt. Die Auswertung der spezifizierten Erwartungen wurde durchgeführt, um grobe Fehler in der Assistenz zu finden. So wurde zu Beginn aller Vorträge geprüft, ob der Inhalt des persönlichen Gerätes des Vortragenden überhaupt auf einer Leinwand angezeigt wird, oder einfach unsichtbar bleibt. Über diese Prüfung ließ sich für die Probanden leicht der Fehler erkennen, dass die Folien von einem fremden Gerät projiziert wurden. Im Gespräch nach der Studie bewerteten die Teilnehmer das Protokoll mit Warnungen bei Abweichungen von der spezifizierten Erwartung als sehr hilfreich. Während der Studie war zu beobachten, dass manche Probanden anfangs die grafische Prüfung der Effekte vorzogen, bis sie erkannten, dass die Warnungen auf Basis der Erwartungen häufig gute Ergebnisse liefern und die grafische Kontrolle vernachlässigten. Bei Verwendung des Programms sollten Nutzer daher darauf hingewiesen werden, dass nur Kandidaten für Probleme automatisch identifiziert werden, aber eine Überprüfung durch einen Experten notwendig ist. Es gab allerdings auch viele Nutzer, die diese eigene Prüfung der Effekte vornahmen.

Um die Ursache eines Fehlers der Assistenz zu identifizieren, gingen die Probanden verschieden vor. Manche prüften grafisch, wie sich der Effekt in der virtuellen Umgebung auswirkt in Änderungen der Gerätezustände. Andere prüften die Spezifikation des Effektes textuell. Für alle gefundenen Fehler konnten die Probanden die Ursache identifizieren. Um die textuelle Prüfung weiter zu vereinfachen schlugen Probanden vor, bei Auswahl einer Vorbedingung oder eines Effektes die beteiligten Nutzer und Geräte in der Draufsicht des Raumes farblich hervorzuheben, so wie Tabellenkalkulationssysteme Formelbezüge visualisieren. Zudem wurde der Wunsch geäußert, bei Verweilen mit dem Mauszeiger über einem Gerät die Kette der Videoverbindungen von der Quelle des Dokumentes bis zur Bildausgabe farblich hervorzuheben. Beide Punkte sollten in der nächsten Version des virtuellen Smart Environments berücksichtigt werden.

Alles in allem gelang es den Probanden aber gut, Fehler im Modell durch dessen Animation aufzudecken.

These 4: Fehler im kooperativen Aufgabenmodell lassen sich direkt aus dem virtuellen Smart Environment heraus einfach beheben.

Nachdem die Probanden die Fehlerursache identifiziert hatten, konnten sie alle gefundenen Fehler beheben. Um die Vorbedingungen und Effekte korrigieren zu können, bekamen sie eine Referenz der möglichen Befehle. Durch die Verwendung der Funktion „Autovervollständigen“ konnten die Probanden jedoch weitestgehend auf die Referenz verzichten. Die Probanden kamen gut mit dem Arbeitsablauf klar, einen fehlerhaften Effekt rückgängig zu machen, den Effekt zu verändern und anschließend die Aufgabe erneut auszuführen, die den Effekt auslöst. Um die Festlegung von Effekten noch effizienter zu gestalten wurde der Vorschlag geäußert, Effekte alternativ zur textuellen Eingabe auch grafisch eingeben zu können. Also wie bei Tabellenkalkulationen möglich, grafisch festzulegen und im Hintergrund aufzuzeichnen.

7.2 Evaluation der frühen Implementation

Künstliche Sensordaten können für vielfältige Zwecke verwendet werden, bspw. für:

- die Expertenevaluation von einzelnen Komponenten des Raumes (bspw. Intentionserkennung, Strategieplanung), die Kontextinformationen als Eingabe benötigen, während noch kein komplett ausgestatteter Raum zur Verfügung steht,
- die Nutzerstudien mit „Wizard of Oz“-Experimenten, wenn Kontextdaten für die Einbindung einzelner funktionsfähiger Geräte benötigt werden und
- das Maschinenlernen, entweder in sehr frühen Entwicklungsphasen, in denen noch keine realen Sensordaten zur Verfügung stehen, oder in späteren Entwicklungsphasen, in denen reale Daten auf Grund von Problemen mit der Laborinfra-

struktur Lücken aufweisen, die durch künstliche Sensordaten geschlossen werden können.

Zur Verwendung künstlicher Sensordaten soll beispielhaft ein Experiment vorgestellt werden. Das Ziel bestand darin, eine Komponente für die Intentionserkennung zu testen, unter der Annahme, dass noch kein komplett instrumentiertes Smart Environment vorliegt. Basierend auf dem fachlichen Wissen eines Domänenexperten wurde ein statistisches Modell erstellt. Das Vorgehen bei einem modellbasierten Entwicklungsprozess wird in [Gie09] und [BPK+09] erläutert.

Als zu untersuchendes Artefakt lag ein HMM („Hidden Markov Model“)vor, das aus Positionsdaten von Nutzern den Fortschritt einer Besprechung vorhersagen sollte. Aus der Beobachtung der Positionsdaten von drei Nutzern innerhalb des Grundrisses eines Smart Environments sollte das HMM abschätzen, welche der Aktivitäten „Presentation A“, „Presentation B“, „Presentation C“, „Discussion“ oder „Exit“ vorlag. Wie bereits in Abbildung 6.3 dargestellt, dürfen die drei Nutzer in beliebiger Reihenfolge vortragen, wobei anschließend eine Diskussion und das Verlassen des Raumes folgen.

Zur Evaluation wird das in Kapitel 6.2.1 vorgestellte virtuelle Smart Environment verwendet. Das Aufgabenmodell beschreibt insgesamt sechs Szenarien (alle Permutationen von A, B und C), um die Besprechung erfolgreich zu absolvieren. Alle sechs wurden im virtuellen Smart Environment animiert. Die Sensordaten wurden mit einer Abtastrate von 1Hz und einem normalverteilten Fehler (mit Varianz von 20cm) durch Box-Müller-Transformation [BM58] generiert. Ein Ausschnitt der generierten Sensordaten ist in Abbildung 7.2 dargestellt.

```
  Position       Zeitstempel         Transponder
…
p  99.000 -53.000 633923383291090000 "RAW102"
p 177.000 -29.000 633923383322180000 "RAW103"
p 270.000 -71.000 633923383343590000 "RAW104"
p 183.000 103.000 633923383463120000 "RAW102"
p  50.000 139.000 633923383473900000 "RAW102"
p 162.000  52.000 633923383484680000 "RAW103"
p 148.000 125.000 633923383491400000 "RAW103"
p 234.000  45.000 633923383500310000 "RAW104"
p 334.000 105.000 633923383511710000 "RAW104"
p 429.000 186.000 633923383521400000 "RAW104"
p 570.000 236.000 633923383528120000 "RAW104"
p  48.000 256.000 633923383540310000 "RAW102"
p  59.000 189.000 633923383551250000 "RAW103"
p  37.000 369.000 633923383561400000 "RAW102"
p  53.000 293.000 633923383575150000 "RAW103"
p 638.000 353.000 633923383587650000 "RAW104"
p 627.000 458.000 633923383600620000 "RAW104"
…
```

Abbildung 7.2: Ausschnitt eines Stromes künstlich generierter Sensordaten

Die generierten Positionsdaten umfassen:

- ein „p“ zur Kennzeichnung des Sensordatentyps als Positionsdaten,
- die x- und y-Koordinaten, wobei der Koordinatenursprung sich an der Wandecke links von der Eingangstür des Raumes befindet (vgl. Abbildung 3.3) und negative Koordinaten sich dadurch erklären, dass die Sensoren den Aufenthalt einer Person als außerhalb des Raumes bestimmen (bspw. wenn Personen vor der Tür im Flur warten),
- einen Zeitstempel und
- die identifizierende Kennung des Transponders des Positionserkennungssystems. Im Beispiel hat Person A „RAW102“, Person B „RAW103“ und Person C „RAW104“.

Lädt man die generierten Sensordaten in das HMM, versucht dieses daraus die Aktivitäten der Nutzer zu erkennen. Ein Beispiel für das Ergebnis solch einer Erkennung ist in Abbildung 7.3 dargestellt. Die Diagramme der anderen fünf Szenarien finden sich in Anhang A.3.

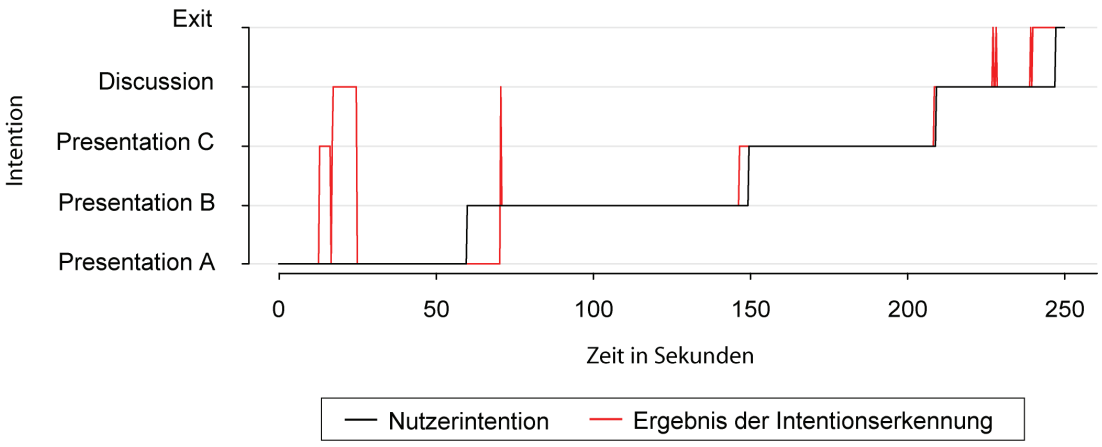

Abbildung 7.3: Beispiel für die erkannten Intentionen (ABCD-Besprechung)

Das Diagramm zeigt eine Besprechung mit der Reihenfolge der Präsentationen der Nutzer A, B und C, gefolgt von einer Diskussion und dem Verlassen des Raumes. Die Gesamtdauer der Besprechung ist mit 4 Minuten sehr kurz gewählt. Dies ist für diesen Test aber ausreichend, da der Schwerpunkt auf dem Wechsel von Aufgaben liegt. Dargestellt ist die im Zeitverlauf wechselnde Nutzerintention (schwarz), entsprechend der laut Testplan durchzuführenden Aufgabenfolge. Zusätzlich dargestellt sind die durch ein HMM vorhergesagten Intentionen (rot). Obwohl es sich bei Intentionen um eine Skala mit diskreten Werten handelt, wurden Wechsel zwischen Intentionen mit vertikalen Linien verbunden, um Wechsel optisch hervorzuheben und den zeitlichen Verlauf besser nachverfolgbar zu machen.

Betrachtet man die Diagramme der sechs durchgeführten Tests (vgl. auch Anhang A.3), zeigt sich, dass die vom HMM gelieferten Intentionen zu Beginn stärker von der Realität abweichen, aber nach kurzer Zeit eine hohe Übereinstimmung zeigen. Diese Abweichungen treten auch bei der Interpretation realer Sensordaten auf [Gie09]. Das HMM interpretiert die

künstlichen Sensordaten mit einer gewissen Toleranz richtig. Eine 100-prozentige Übereinstimmung ist nicht zu erwarten, da Übergänge zwischen Handlungen fließend sind und auch ein Beobachter Beginn und Ende einer Aufgabe nicht exakt benennen kann.

Beim ersten (hier nicht grafisch dargestellten) Testlauf mit den künstlichen Sensordaten wurden die Szenarien, die mit A beginnen richtig erkannt, während die mit B und C beginnenden Szenarien falsch erkannt wurden. Nach einer Analyse des HMMs konnte ein Fehler in den Initialwahrscheinlichkeiten der Zustände B und C identifiziert und durch die entsprechenden Kollegen behoben werden. Somit konnte durch die Verwendung der künstlichen Sensordaten aus dem virtuellen Smart Environment ein realer Fehler entdeckt werden, der bei der Planung dieses Tests noch unbekannt war. Die Sensordaten wurden anschließend erneut durch das verbesserte HMM interpretiert und die Ergebnisse wurden in den Diagrammen dargestellt.

Die Nützlichkeit des Evaluationsansatzes erhöht sich für komplexere HMMs, die weitere Kontextvariablen, wie das Berühren von Geräte und Gegenständen berücksichtigen.

7.3 Zusammenfassung

Zunächst wurde in einer Nutzerstudie untersucht, wie gut Testpersonen mit dem virtuellen Smart Environment umgehen können. Dabei standen 4 Thesen im Mittelpunkt:

- 1. Nutzer können sich in die Darstellung des virtuellen Smart Environments hineinversetzen.
- 2. Nutzer sind in der Lage, Interaktionen nach realem Vorbild zu animieren.
- 3. Fehler im kooperativen Aufgabenmodell lassen sich durch Animation finden.
- 4. Fehler im kooperativen Aufgabenmodell lassen sich direkt aus dem virtuellen Smart Environment heraus einfach beheben.

Die Studie konnte die 4 Thesen mit empirischen Daten qualitativ belegen. Sowohl Nutzer, die das reale „Smart Environment“-Labor kannten, als auch solche, die es nicht kannten, konnten ein kooperatives Aufgabenmodell animieren und dabei ein konkretes Szenario durchspielen. Mehreren Nutzern bereitete der Test in der virtuellen Umgebung Freude und es konnten Vorschläge zur Optimierung des Programmes gesammelt werden. Die Probanden fanden die präparierten Fehler und ein Proband erkannte darüber hinaus ein weiteres Usability-Problem.

Basierend auf diesen Erfahrungen, wurde ein weiteres Experiment durchgeführt. Ein Experte generierte künstliche Sensordaten für 6 Besprechungen mit jeweils 3 Vorträgen. Diese wurden in ein HMM zur Intentionserkennung geladen, um dieses zu evaluieren. Dabei wurde ein Fehler entdeckt, der bis dahin nicht bekannt war, und von den entsprechenden Kollegen behoben.

Kapitel 8

Schlussbemerkungen

8.1 Zusammenfassung

Die vorliegende Arbeit setzt sich mit dem Thema der Usability-Evaluation in intelligenten Umgebungen auseinander. Dazu wird in Kapitel 2 zunächst der Begriff der Usability (dt. Gebrauchstauglichkeit) als eine zentrale Produkteigenschaft definiert, die eine effektive, effiziente und zufriedenstellende Arbeit ermöglichen soll. Ein Überblick zeigt, dass für die Evaluation klassischer Arbeitsplatzanwendungen zahlreiche etablierte Methoden zur Verfügung stehen. Smart Environments kennzeichnen sich jedoch durch besondere Eigenschaften, wie eingebettete Benutzungsschnittstellen, multimodale Interaktionen, eine starke Kontextabhängigkeit und die Erwartung der Nutzer, ohne Bedienungsanleitung direkt arbeiten zu können. Daraus resultieren neue Herausforderungen für die Usability-Evaluation. Entsprechend werden bestehende Methoden adaptiert und ergänzend neue Methoden eingeführt. Eine Untersuchung bestehender Projekte zeigt, wie diese Methoden angewandt werden.

Aus der Analyse des Vorgehens bei der Evaluation in bestehenden Projekten werden in Kapitel 3 Anforderungen an ein Vorgehensmodell abgeleitet:

- Unterstützung früher Phasen des Entwicklungsprozesses
- Konkrete Ausgestaltung des Paradigmas der „Benutzer-orientierten Gestaltung“
- Aufgabenbasierter Prozess
- Integration von Entwicklung und Evaluation
- Berücksichtigung des Datenschutzes

Das in dieser Arbeit entwickelte Vorgehensmodell unterstützt den Prozess der Benutzer-orientierten Gestaltung von Smart Environments durch iterative Evaluation. Als Werkzeugunterstützung wird ein virtuelles Smart Environment vorgestellt, das Entwicklungs- und Evaluationswerkzeuge integriert. Die Anwendung des Vorgehensmodells wird in drei Phasen des Entwicklungsprozesses erläutert: Anforderungsanalyse, Design und Implementation.

Das Vorgehen während der Anforderungsanalyse wird in Kapitel 4 beschrieben. Das Ziel besteht darin, Eigenschaften an das zu entwickelnde System zu gewinnen. Um ein Verständnis für Nutzer und deren Arbeitsabläufe zu gewinnen, wird eine Aufgabenanalyse durchgeführt. Ausgehend von Nutzerprofilen werden Aufgabenmodelle entwickelt. Diese können in einem Aufgabenmodellsimulator animiert werden, um sie im Rahmen der Evaluation interaktiv zu durchlaufen und Modellinkonsistenzen identifizieren und beheben zu können. Da Nutzer typischerweise keine Erfahrungen mit Smart Environments haben, fällt es ihnen schwer, Anforderungen an eine zukünftige Assistenz zu formulieren. Daher besteht die Herausforderung darin, Nutzern ein Gefühl für die Assistenz zu vermitteln. Das entwickelte virtuelle Smart Environment unterstützt daher die Durchführung von „Wizard of Oz"-Experimenten.

Basierend auf den Dokumenten der Anforderungsanalyse beschreibt Kapitel 5 wie das Design durchgeführt wird. Aufgabenmodelle werden iterativ verfeinert, wobei kooperative und kontextuelle Abhängigkeiten ergänzt werden. Die Herausforderung besteht in der frühzeitigen Evaluation dieser Artefakte, bevor das physische Smart Environment gebaut wird, da der komplette Aufbau sehr aufwendig und kostenintensiv ist. Das virtuelle Smart Environment erlaubt die Animation der kooperativen Aufgabenmodelle. Ergänzend werden Nutzer, Geräte und Gegenstände in einer Draufsicht visualisiert und können in Nutzerinteraktionen einbezogen werden. Bei Geräten (wie PDAs oder Mobiltelefonen), deren grafische Benutzungsschnittstelle aufgabenmodellbasiert entwickelt wurde, kann dieses Zusatzwissen einbezogen werden. Ein Prototyp der Schnittstelle wird visualisiert und ist in die Evaluation integriert.

Kapitel 6 erläutert das Vorgehen bei der Implementation. Aufgabenmodelle werden schrittweise in statistische Modelle und schließlich in Quellcode für die Intentionserkennung überführt. Die entwickelten Softwarekomponenten, wie Intentionserkennung und Strategieplanung, lassen sich allerdings nicht losgelöst vom Anwendungskontext testen. Das virtuelle Smart Environment wurde so erweitert, dass künstliche Sensordaten generiert werden können. Diese dienen als Eingabe für die zu evaluierende Komponente, während Ausgaben in der virtuellen Umgebung visualisiert werden, um fehlerhaftes Systemverhalten zu idenifizieren.

Der zweite Teil von Kapitel 6 thematisiert die Evaluation eines komplett eingerichteten physischen Smart Environments, in dem Nutzerstudien der vollständigen Funktionalität durchgeführt werden können. Durch Aufzeichnungen von Sensor- und Videodaten entstehen große Datenmengen. Diese werden häufig manuell analysiert, um Nutzerverhalten zu identifizieren, das von der Erwartung an eine effiziente Aufgabenausführung abweicht. Um diesen aufwendigen Prozess teilweise zu automatisieren, wird eine Methode erläutert, mit der implizit bei Usability-Experten vorhandenes Wissen über Erwartungen explizit spezifiziert werden kann. Darüber hinaus wird das Prinzip der aufgabenbasierten Evaluation, das bisher nur für traditionelle Applikationen und Webapplikationen etabliert ist, in dieser Arbeit auf Smart Environments ausgeweitet. Optionen zur Visualisierung von Interaktionen im virtuellen Smart Environment und zur Annotation runden das Spektrum angebotener Methoden ab. Ein Nutzerinformationssystem verwendet den entwickelten Algorithmus zur

Aggregation von TaskTraces, um die Arbeitsweise des Assistenzsystems transparenter zu machen.

Kapitel 7 berichtet über erste Erfahrungen bei der Anwendung des entwickelten Vorgehensmodells. In einer Nutzerstudie führten 24 Hochschulmitarbeiter in der Funktion als Interaktionsdesigner bzw. Usability-Experten eine Evaluation während der Designphase eines Smart Environments durch. Ein kooperatives Aufgabenmodell wurde interaktiv durchlaufen (im Sinne eines „Walkthrough"), um Abweichungen von Anforderungen zu identifizieren und zu beheben. Die Anwender konnten mit Hilfe des virtuellen Smart Environments die präparierten Fehler finden und damit empirisch zeigen, dass sie (1) sich in die virtuelle Darstellung hineinversetzen können, (2) mit der Animation von Interaktionen klar kommen, (3) dadurch Fehler identifizieren und (4) schließlich direkt beheben können. In der Studie wurde sogar ein zusätzliches Usability-Problem gefunden, das vorher nicht bekannt war. Anschließend wurde eine Expertenevaluation während der frühen Implementationsphase eines Smart Environments durchgeführt. Es wurden künstliche Sensordaten für 6 verschiedene Besprechungsabläufe generiert und in ein statistisches Modell der Intentionserkennung geladen. Auch hier konnte ein vorher unbekanntes Problem identifiziert werden, das den Kollegen der entsprechenden Arbeitsgruppe nicht bekannt war und daraufhin behoben werden konnte.

Vollzieht man einen Blick zurück auf die anfängliche Zielstellung aus Kapitel 1.2, so wurde folgendes erreicht:

Spezifikation von Anforderungen an die Usability-Evaluation von Smart Environments

Ausgehend von einer Untersuchung bestehender Methoden zur Evaluierung der Usability in Smart Environments (Kapitel 2), werden Herausforderungen aufgezeigt und Anforderungen abgeleitet (Kapitel 3.1).

Entwicklung eines Vorgehensmodells für die Usability-Evaluation von Smart Environments

Entsprechend der identifizierten Anforderungen wird in der vorliegenden Arbeit ein Vorgehensmodell erarbeitet, das Prinzipien der Benutzer-orientierten Gestaltung auf Smart Environments überträgt. Der iterative Prozess besteht aus den Phasen Planung, Entwicklung, Evaluation und Verbesserung der Artefakte. Die Ausgestaltung des Vorgehensmodells wird ausführlich für die Phasen der Anforderungsanalyse, des Designs und der Implementation erläutert.

Steigerung der Effizienz der Evaluation durch Werkzeugunterstützung

Um eine effiziente Evaluation zu ermöglichen, wurde ein virtuelles Smart Environment als Werkzeugunterstützung entwickelt. Dieses bietet Funktionalitäten in vier Anwendungsbereichen an:

- (1) Animation von kooperativen Aufgabenmodellen mit Kontextinformationen für Expertenevaluation durch interaktives Durchlaufen („Walkthrough“)
- (2) „Wizard of Oz“-Experimente, um Nutzern eine Vorschau auf die gewünschte Assistenz zu geben
- (3) Generierung von künstlichen Sensordaten zur Evaluation einzelner implementierter Komponenten
- (4) Anonymisierte Visualisierung von aufgezeichneten Daten aus Nutzertests des komplett eingerichteten physischen Smart Environments

Im Vergleich zu anderen Evaluationsmethoden, die einzelne Aspekte der Evaluation betrachten, wird hier ein Vorgehensmodell vorgestellt, das eine Integration zwischen Entwicklung und Evaluation über mehrere Phasen der Entwicklung beschreibt. Zudem sind nun Expertenevaluationen von Artefakten früher Entwicklungsphasen möglich, die es einem Experten erlauben, Szenarien aus Perspektiven verschiedener Nutzerrollen zu betrachten.

8.2 Ausblick

In der vorliegenden Arbeit wurden zahlreiche Fragestellungen und Problembereiche bearbeitet, manche weiterführenden konnten jedoch nicht berücksichtigt werden. Diese eröffnen Möglichkeiten für zukünftige Forschungsvorhaben.

Weiterführende Integration von Entwicklungs- und Evaluationswerkzeugen
Wenn ein Usability-Problem identifiziert wurde, erlauben die entwickelten Werkzeuge die direkte Verbesserung einiger Artefakte. So können Raumpläne, Vorbedingungen und Effekte während der Expertenevaluation direkt editiert und erneut getestet werden. Wünschenswert wäre es, wenn man auch die Aufgabenstruktur selbst während des Durchlaufens verändern und weiter verfeinern könnte.

Momentan müssen nach Änderungen in kooperativen Aufgabenmodellen die statistischen Modelle über verschiedene Editoren neu generiert werden. Hier wäre eine direkte Integration der Werkzeuge sinnvoll, die diesen Prozess automatisch gewährleistet.

3D-Visualisierung des virtuellen Smart Environments
Das virtuelle Smart Environment stellt die Umgebung und Interaktionen darin als 2D-Visualisierung dar. Eine 2D-Visualisierung hat sich bei anderen Simulatoren ebenfalls bewährt für die Darstellung der Historie von Interaktionen (bspw. [IRC05], [RYC+07]) und Manipulation der Umgebung (bspw. [SH05]), während die aktuelle Situation häufig in 3D dargestellt wird. Für das Abspielen von Sensordaten könnte im entwickelten virtuellen Smart Environment ebenfalls 3D verwendet werden.

Vorgehensmodell um weitere Evaluationsmethoden ergänzen
Das Vorgehensmodell schlägt derzeit für jede Phase bestimmte Evaluationsmethoden vor. Weiterführende Untersuchungen könnten ermitteln, welche Verfahren sich für die Phasen ebenfalls eignen und Kombinationen verschiedener Methoden auf ihr Potential hin analysieren.

Weiterführende Evaluationen
Nachdem erste Experimente mit dem Vorgehensmodell durchgeführt wurden, können weitere Evaluationen an komplexeren Smart Environments durchgeführt werden, um Erfahrungen in Projekten zu sammeln. Das Interesse besteht darin, zu sehen, wie die bereitgestellten Werkzeuge angenommen werden und wie sich das Vorgehen in verschiedenen Organisationen umsetzen lässt.

Erarbeitung von Heuristiken
Das entwickelte virtuelle Smart Environment erlaubt das interaktive Durchlaufen von kooperativen Aufgabenmodellen. Während der Evaluation animiert ein Experte verschiedene Szenarien, um zu überprüfen, ob das Modell den Anforderungen entsprechend reagiert. Die Identifikation allgemeiner Heuristiken kann helfen, immer wiederkehrende Probleme zu berücksichtigen. Beispiele solcher Heuristiken könnten sein, dass es zur Erfüllung einer Aufgabe immer mindestens zwei Alternativen geben soll oder dass ein Effekt einer Aufgabe maximal eine Sprachausgabe gleichzeitig vornehmen darf.

Evaluation und Erweiterung des Nutzerinformationssystems
Eine Nutzerstudie des Nutzerinformationssystems wäre wünschenswert, um Erfahrungen mit in die Weiterentwicklung einfließen zu lassen. Zudem sollte es für Nutzer möglich sein, bei fehlerhafter Assistenz dies dem System direkt mitzuteilen. So könnte der Nutzer einerseits die fehlerhaft erkannte Aufgabe direkt per „undo“-Mechanismus rückgängig machen und die stattdessen tatsächlich zu unterstützende Aufgabe auswählen. Als Basis könnten „undo“-Mechanismen für Aufgabenmodelle [CF06, RF08] in adaptierter Form dienen. Andererseits könnte das System diese Nutzerentscheidungen verwenden, um das Modell automatisch zu verbessern, indem bspw. die Wahrscheinlichkeit für den Übergang zwischen Zuständen verändert wird. Analog dazu schlagen Cook et al. [RYC+07] eine Bewertung der Assistenz durch den Nutzer vor, um das Maschinenlernen zu ermöglichen.

Anhang

A1 Methodenreferenz für TaskExpressions

TaskExpressions werden zur Spezifikation von Erwartungen in Kapitel 6.2.2.1 verwendet. Bei der Bezeichnung der Entitäten wurde zur Vereinfachung auf den Namenszusatz „Instance“ verzichtet. Alle Entitäten sind Instanzen in der Laufzeitumgebung.

SimpleElement

- getId() : String
  Liefert die ID der Entität
- getName() : String
  Liefert den Namen der Entität
- getModelElement() : Object
  Liefert das interne Objekt, das zusätzliche Verwaltungsinformationen enthält

Evaluation

- getUserByName(name : String) : User
  Findet Nutzer mit bestimmtem Namen
- getTasksByName(name : String) : TaskList
  Findet alle Aufgaben mit einem bestimmtem Namen

EvaluationElement

- isLocatedAt(location : Location) : Boolean
  Ermittelt, ob sich eine Entität an einem bestimmtem Ort befindet

User

- getRoleByName(name : String) : Role
  Findet von einem Nutzer eine Rolle an Hand des Namens
- getTaskByName(name : String) : Task
  Findet eine Aufgabe mit bestimmtem Namen
- isCarryingItem(item : EvaluationElement) : Boolean
  Ermittelt, ob ein Nutzer ein Gerät oder Gegenstand bei sich trägt.

Device

- isConnectedTo(device : Device) : Boolean
  Ermittelt, ob ein Gerät mit einem anderen direkt oder indirekt (über n Teilverbindungen) per Videoverbindungen verbunden ist
- isConnectedTo(type : DeviceType) : Boolean
  Ermittelt, ob ein Gerät mit einem Gerät eines bestimmten Typs direkt oder indirekt (über n Teilverbindungen) per Videoverbindungen verbunden ist
- isInState(state : String) : Boolean
  Ermittelt, ob ein Gerät einen bestimmten Zustand besitzt

Role

- getTaskByName(name : String) : Task
  Liefert eine Aufgabe mit bestimmtem Namen (In Aufgabenmodellen in CTT-Notation muss ein Aufgabenname eindeutig auf eine Aufgabendefinition verweisen. [Pat99])

Task

- getChildTaskByName(name : String) : Task
  Liefert eine Unteraufgabe mit bestimmtem Namen
- getIterations() : Integer
  Liefert die aktuelle Anzahl von Wiederholungen einer Aufgabe
- getDuration(iteration : Integer) : Long
  Liefert die Ausführungsdauer der aktuellen Aufgabe
- isPerformedByUser(user : User, iteration : Integer) : Boolean
  Ermittelt, ob die n-te Wiederholung der Aufgabe von einem bestimmten Nutzer ausgeführt wurde
- isPerformedWithDevice(device : Device, iteration : Integer) : Boolean
  Ermittelt, ob die n-te Wiederholung der Aufgabe mit einem bestimmten Gerät ausgeführt wurde
- isPerformedWithArtifact(artifact : Artifact, iteration : Integer) : Boolean
  Ermittelt, ob die n-te Wiederholung der Aufgabe von einem bestimmten Gegenstand ausgeführt wurde
- isPerformedInContext(context : Context, iteration : Integer) : Boolean
  Ermittelt, ob die n-te Wiederholung der Aufgabe in einem bestimmten Kontext ausgeführt wurde

TaskList

- count() : Integer
  Liefert die Anzahl der Aufgaben in der Liste
- countTasksWithState(state : TaskState) : Integer
  Liefert die Anzahl der Aufgaben eines bestimmten Zustandes
- isAllTasksInState(state : TaskState) : boolean
  Ermittelt, ob alle Aufgaben den gleichen Zustand haben
- isExistingTaskInState(state : TaskState) : booelan
  Liefert, ob mindestens eine Aufgabe einen bestimmten Zustand hat

A2 Szenario der Nutzerstudie (Kapitel 7.1)

1. Betreten des Raumes

- Alle Nutzer betreten den Raum und gehen jeweils in ihre Zone SitA, SitB und SitC
- Jeder legt sein persönliches Gerät vor sich auf den Tisch

2. Präsentation von Person A

- PersonA nimmt LaptopA
- PersonA geht in Zone PresentA
- PersonA legt LaptopA auf einem Tisch ab
- PersonA nimmt Presenter (die Fernbedienung zum Folienweiterschalten)
- Aufgabe PresentationA start
    - Effekt: Videoverbindung von LaptopA nach LW4
- Aufgabe PresentationA end
    - Effekt: Videoverbindung abgebaut
- PersonA legt Presenter ab
- PersonA nimmt LaptopA
- PersonA geht in Zone SitA
- PersonA legt LaptopA auf den Tisch

3. Präsentation von Person B

- PersonB nimmt PDA B
- PersonB geht in Zone PresentB
- PersonB legt PDA B auf einem Tisch ab
- PersonB nimmt Presenter (die Fernbedienung zum Folienweiterschalten)
- Aufgabe PresentationB start
    - Effekt: Videoverbindung von PDA B nach VD2
- Aufgabe PresentationB end
    - Effekt: Videoverbindung abgebaut
- PersonB legt Presenter ab
- PersonB nimmt PDA B
- PersonB geht in Zone SitB
- PersonB legt PDA B auf den Tisch

4. Präsentation von Person C

- PersonC nimmt LaptopC
- PersonC geht in Zone PresentC
- PersonC legt LaptopC auf einem Tisch ab
- PersonC nimmt Presenter (die Fernbedienung zum Folienweiterschalten)
- Aufgabe PresentationC start
    - Effekt: Videoverbindung von LaptopC nach LW6
- Aufgabe PresentationC end
    - Effekt: Videoverbindung abgebaut
- PersonC legt Presenter ab
- PersonC nimmt LaptopC
- PersonC geht in Zone SitC
- PersonC legt LaptopC auf den Tisch

5. Diskussion

- Alle Nutzer sind auf ihrem Platz (SitA, SitB, SitC)
- Aufgabe DiscussionD start
    - Effekt: alle Dokumente von A, B, C anzeigen
    - (wie oben: LaptopA - LW4, PDA B – VD2, LaptopC – LW6)
- Aufgabe DiscussionD end
    - Effekt: alle Videoverbindungen abgebaut

6. Verlassen des Raumes

- Alle Nutzer nehmen ihr persönliches Gerät und verlassen den Raum
- Aufgabe ExitE start
    - Effekt: -
- Aufgabe ExitE end
    - Effekt: -

A3 Weitere Ergebnisse zum Experiment (Kapitel 7.2)

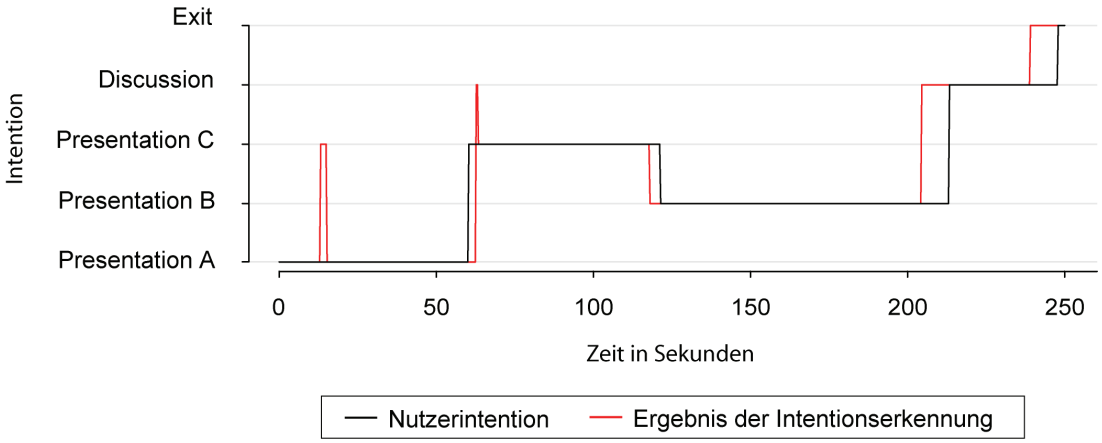

Abbildung A.1: erkannte Intentionen für Besprechung ACBD

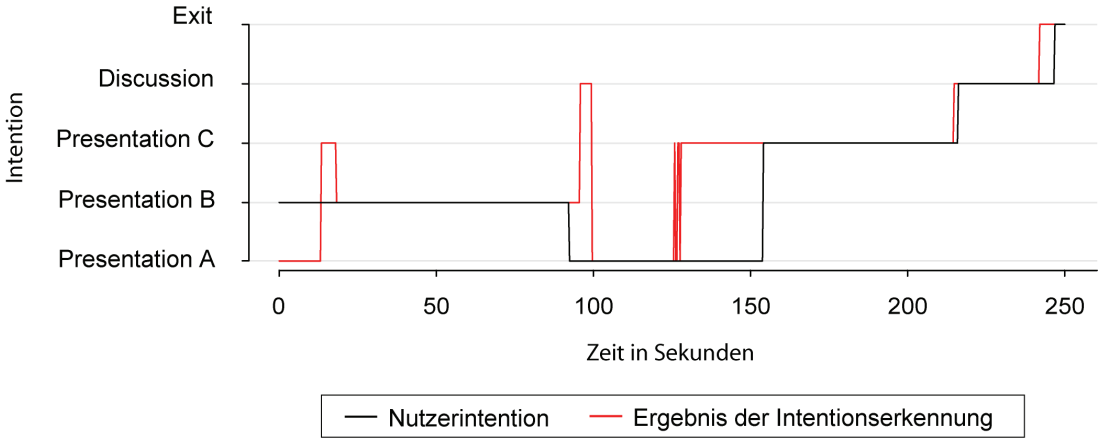

Abbildung A.2: erkannte Intentionen für Besprechung BACD

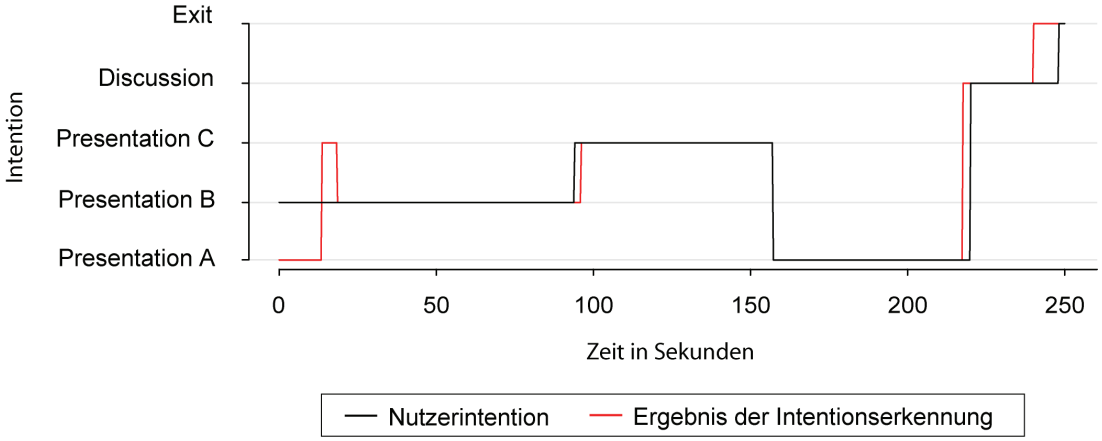

Abbildung A.3: erkannte Intentionen für Besprechung BCAD

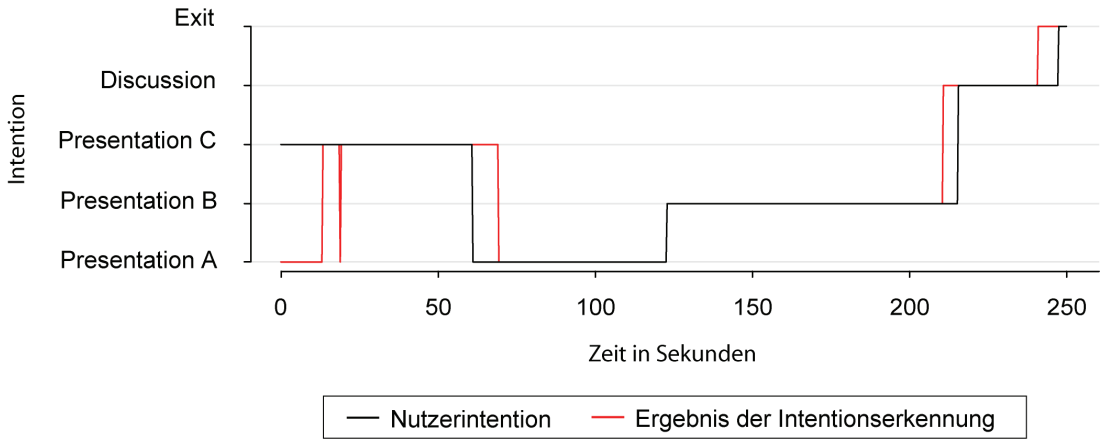

Abbildung A.4: erkannte Intentionen für Besprechung CABD

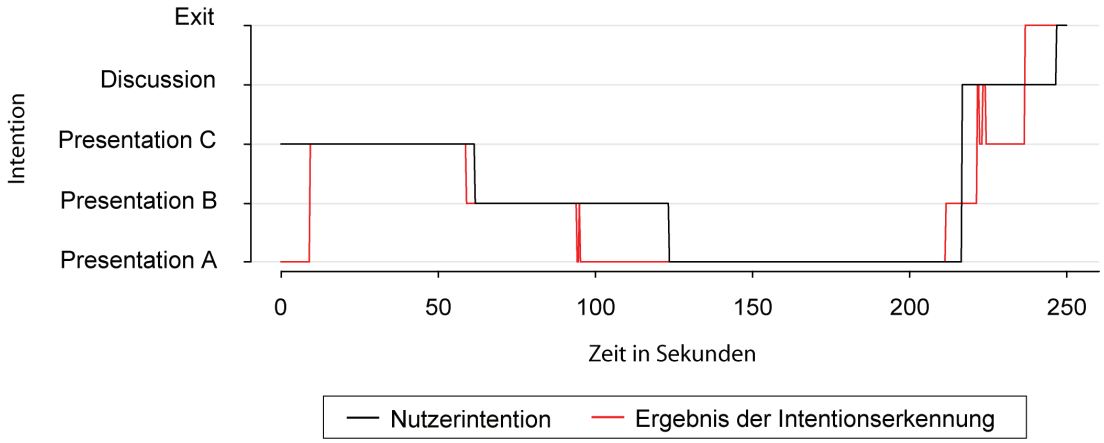

Abbildung A.5: erkannte Intentionen für Besprechung CBAD

Glossar

Aktoren: Typischerweise kleine, elektromechanische Geräte, die den physischen Zustand der Umgebung verändern können.

Aufgabe: „Zur Zielerreichung erforderliche Aktivitäten“. [DIN98]

Aufgabenmodell: Beschreibt wie Menschen durch die Ausführung von Aufgaben ein bestimmtes Ziel erreichen können. Aufgaben werden in Unteraufgaben zerlegt, für die eine zeitliche Abfolge festgelegt wird.

Bayes' Theorem: Eine mathematische Beziehung zwischen Wahrscheinlichkeiten, die die Aktualisierung von Wahrscheinlichkeiten auf der Basis neuer Informationen erlaubt. Es kann als formale Grundlage für das Maschinenlernen verwendet werden.

Durchlaufen (engl. „Walkthrough“): Eine Methode der Expertenevaluation, bei der Experten sich in die Lage späterer Nutzer versetzen und aus deren Perspektive die Arbeit mit dem frühen Konzept oder Prototyp des Produktes nachvollziehen. (vgl. [Rub94])

Explizite Interaktion: Der Nutzer bestimmt selbst wann was geschehen soll.

Grafische Benutzungsoberfläche (GUI – „Graphical User Interface“): Die Verwendung grafischer Symbole (bspw. Ordner und Fenster) zur Repräsentation von Objekten, mit denen der Nutzer am Bildschirm interagieren kann.

Implizite Interaktion: Auf Basis des Nutzerverhaltens und des Umgebungszustandes entscheidet das Assistenzsystem wann und was geschehen soll. (vgl. [Shi07])

Intentionserkennung: Algorithmen, die versuchen aus dem aktuellen Zustand der Umgebung die möglichen Ziele von darin befindlichen Nutzern oder Nutzergruppen abzuleiten.

Intelligente Umgebung: Synonym zu „Smart Environment“ verwendet.

Persona: Fiktive, archetypische Person, die Verhalten, Ziele und Motivationen beinhaltet und im Rahmen des nutzerzentrierten Designs (UCD) für Designentscheidungen dient, wenn keine realen Nutzer zur Verfügung stehen.

Sensoren: Ein technisches Bauteil, das qualitative und / oder quantitative Eigenschaften der Umgebung erfasst (bspw. Temperatur, Nutzerpositionen, Handhabung von Objekten durch Nutzer).

Smart Environment: Physische Orte, die in der Lage sind, auf Aktivitäten von Nutzern zu reagieren, um Nutzern Assistenz zu bieten und ihnen damit helfen, ihre Ziele in der Umgebung zu erfüllen. [AE06]

Szenario: Eine Sequenz von Aufgaben, die einen validen Durchlauf durch ein Aufgabenmodell beschreibt. Ein Aufgabenmodell beschreibt eine Menge von Szenarien.

Tätigkeit: „die Organisation und die zeitliche und räumliche Abfolge der Arbeitsaufgaben einer Person oder die Kombination der gesamten menschlichen Arbeitshandlungen eines Arbeitenden/Benutzers in einem Arbeitssystem“ [DIN04]

Usability (dt. „Gebrauchstauglichkeit“): „das Ausmaß, in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen“ [DIN98]

User-Centred Design (dt. „Benutzer-orientierte Gestaltung“): „Art der Entwicklung interaktiver Systeme, die sich darauf konzentriert, Systeme gebrauchstauglich zu machen.“ [DIN99] Grundlage hierfür ist der verstärkte Einbezug von Nutzern und die iterative Wiederholung von Entwicklung und Evaluation.

User Interface (Benutzungsschnittstelle): „den Benutzern angebotene und benutzbare Interaktionsmöglichkeiten eines Anwendungssystems sowie den zugehörigen Unterstützungssystemen“ [Her09]

Abkürzungsverzeichnis

CSCW	Computer Supported Cooperative Work
CTT	Concur Task Trees
EMF	Eclipse Modelling Framework
IDE	Integrated Development Environment
GEF	Graphical Editing Framework
GMF	Graphical Modelling Framework
GUI	Graphical User Interface
HMM	Hidden Markov Model
SE	Smart Environment
UCD	User-Centred Design
UI	User Interface
XML	Extensible Markup Language

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

[Aar04] Aarts, E.: Ambient intelligence: A multimedia perspective. IEEE Multimedia, Volume 11, Issue 1, pp. 12-19, 2004.

[Abo99] Abowd, G.D.: Classroom 2000: An Experiment with the Instrumentation of a Living Educational Environment. IBM Systems Journal, Vol. 38, No.4, pp.508-539, 1999.

[AB02] Arnstein, L., Borriello, G.: Labscape: The Design of a Smart Environment, Intel Research Seattle Technical Report IRS-TR-02-008, 2002.

[AD67] Annett, J., Duncan, K.D.: Task analysis and training design. Occupational Psychology, No.41, pp.211-221, 1967.

[AD03] Albert, W.S., Dixon, E.: Is this what you expected? The use of expectation measures in usability testing. Proc. of Usability Professionals Association, Scottsdale, USA 2003.

[AE06] Aarts, E., Encarnação, J.: True Visions. ISBN 3-540-28972-0, Springer, 2006.

[AHI+05] Abowd, G. D., Hayes, G. R., Iachello, G., Kientz, J. A., Patel, S. N., Stevens, M. M., Truong, K. N.: Prototypes and Paratypes: Designing Mobile and Ubiquitous Computing Applications. IEEE Pervasive Computing Vol.4, No.4, pp.67-73, 2005.

[Aik06] Aikio, K.-P.: Integrating Usability Engineering with Software Engineering: a preliminary view on aspects surrounding the topic of usability integration. IRIS 29, 2006.

[AM00] Abowd, G.D., Mynatt, E.D.: Charting past, present, and future research in ubiquitous computing. ACM Transactions on Computer-Human Interaction Issue 7, No.1, pp.29-58, 2000.

[AM07] Aitenbichler, E., Mühlhäuser, M.: Softwareentwicklung für Mundo Smart Environments. 15. ITG-GI-Fachtagung, Kommunikation in verteilten Systemen, Bern, Schweiz, 2007.

[AMR02] Abowd, G.D., Mynatt, E.D., Rodden, T.: The Human Experience. IEEE Pervasive Computing, Volume 1, Issue 1, pp.48-57, 2002.

[Bal98] Balzert, H.: Lehrbuch der Softwaretechnik. Band 2: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung. Spektrum-Verlag, ISBN 3-8274-0065-1, 1998.

[Bal00] Balzert, H.: Lehrbuch der Softwaretechnik. Band 1: Software-Entwicklung. Spektrum-Verlag, ISBN 3-8274-0480-0, 2000.

[Bat95] Bates, P.C.: Debugging heterogeneous distributed systems using event-based models of behaviour. ACM Transactions on Computer Systems (TOCS), Vol.13, No.1, pp.1-31, 1995.

[BB06] Bastide, R., S. Basnyat: Error Patterns: Systematic Investigation of Deviations in Task Models. Task Models and Diagrams (TAMODIA 2006), pp. 109-122, Hasselt, Belgium, 2006.

[BBS99] Biere, M., Bomsdorf, B., Szwillus, G.: The Visual Task Model Builder. Proceedings of the Third International Conference on Computer-Aided Design of User Interfaces, Louvain-la-Neuve, pp. 245-256, 1999.

[BGK+02] Burrell, J., Gay, G., Kubo, K., Farina, N.: Context-Aware Computing: A Test Case. UbiComp 2002, 2002.

[BLF+08] Blumendorf, M., Lehmann, G.,, Feuerstack, S., Albayrak, S.: Executable Models for Human-Computer Interaction. Design, Specification and Verification of Interactive Systems (DSV-IS 2008), Kinston, Canada, pp.238-251, 2008.

[BM58] Box, G. E. P., Muller, M. E.: A Note on the Generation of Random Normal Deviates. The Annals of Mathematical Statistics, Vol.29, No.2, pp.610-611, 1958.

[BMR+00] Buschmann, F., Meunier, R., Rohnert, H., Sommerslad, P., Stal, M.: Pattern-orientierte Softwarearchitektur. ISBN 3-8273-1282-5, Addison-Wesley, 2000.

[BMR+07] Ballagas, R., Memon, F., Reiners, R., Borchers, J: iStuff mobile: rapidly prototyping new mobile phone interfaces for ubiquitous computing. Conference on Human Factors in Computing Systems, San Jose, USA, (CHI '07), pp.1107-1116, 2007.

[Bom07] Bomsdorf, B.: The WebTaskModel Approach to Web Process Modelling. Task Models and Diagrams (TAMODIA 2007), pp.240-253, Toulouse, France, 2007.

[BPK+09] Burghardt, C., Propp, S., Kirste, T., Forbrig, P.: Rapid Prototyping and Evaluation of Intention Analysis for Ambient Environments. Intelligent in-

teractive assistance and mobile multimedia Computing (IMC 2009), Rostock, Germany, 2009.

[BV03] Barton, J.J, Vijayaraghavan: UBIWISE, a simulator for ubiquitous computing systems design. Technical Report HPL-2003-93, HP Laboratories, Palo Alto, 2003.

[CAF02] Consolvo, S., Arnstein, L., Franza, B.: User Study Techniques in the Design and Evaluation of a Ubicomp Environment. 4th international Conference on Ubiquitous Computing (UbiComp 2002), Göteborg, Sweden, pp.73-90, 2002.

[Car07] Card, S.: Information Visualization. pp.509-543, In: Sears, A., Jacko, J: The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications, Second Edition. Lawrence Erlbaum and Associates, ISBN 0-805-85870-9, 2006.

[CB05] Courage, C., Baxter, K.: Understanding Users: A Practical Guide to User Requirements Methods Tools and Techniques. Morgan Kaufmann, ISBN 1-558-60935-8, 2005.

[CBA+07] Coutaz, J., Balme, L., Alvaro, X., Calvary, G., Demeure, A., Sottet, J.:An MDE-SOA approach to support plastic user interfaces in ambient spaces. Human Computer Interaction International (HCII 2007), Beijing, China, pp.63-72, 2007.

[CBG+06] Crabtree, A., Benford, S., Greenhalgh, C., Tennent, P., Chalmers, M., Brown, B.: Supporting Ethnographic Studies of Ubiquitous Computing in the Wild. Designing Interactive Systems, pp.60-69, ACM Press, 2006.

[CBM04] V. Corama, J. Bohn, F. Mattern: Living in a Smart Environment: Implications for the Coming Ubiquitous Information Society, International Conference on Systems, Man and Cybernetics, pp. 5633-5638, IEEE Computer Society, 2004.

[CCT+03] Calvary, G. Coutaz, J. Thevenin, D. Limbourg, Q., Bouillon, L. Vanderdonckt, J.,: A unifying reference framework for multi-target user interfaces, Interacting with Computers, Vol. 15, No.3, pp. 289-308, 2003.

[CD04] Cook, D., Das, S.: Smart Environments. ISBN 0- 471-54448-5, John Wiley & Sons, 2004.

[CDM+00] Cheverst, K., Davies, N., Mitchell, K., Friday, A.: Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project. 6th Annual Conference on Mobile Computing and Networking, Boston, USA, 2000.

[CF06] Cass, A., Fernandes, C.: Using Task Models for Cascading Selective Undo. Task Models and Diagrams (TAMODIA 2006), pp.186-201, Hasselt, Belgium, 2006.

[CHB+07] Consolvo, S., Harrison, B., Smith, I., Chen, M., Everitt, K., Froehlich, J., Landay, J.: Conducting In situ Evaluations for and with Ubiquitous Computing Technologies. International Journal of HCI, Vol.22, No.1, pp.107-122, 2007

[Chi00] Chi, E.H.: A Taxonomy of Visualization Techniques using the Data State Reference Model. Information Visualization 2000, pp.69-75, Salt Lake City, USA, 2000.

[CKF09] Cuřin, J.; Kleindienst, J., Fleury, P.: CHIL Integration Tools and Middleware. In: [WS09], pp.353-364, 2009.

[CLC04] Clerckx, T., Luyten, K., Coninx, K.: The mapping problem back and forth: customizing dynamic models while preserving consistency. Task Models and Diagrams (TAMODIA 2004), pp. 33-42, ACM Press, 2004.

[CLS02] Chandler, C. D., Lo, G., Sinha, A. K.: Multimodal Theater: Extending Low Fidelity Paper Prototyping to Multimodal Applications. CHI 2002, pp. 874-875, 2002.

[CLV+03] Coninx, K., Luyten, K., Vandervelpen, C., Van der Bergh, J., Creemers, B.: Dygimes: Dynamically Generating Interfaces for Mobile Computing Devices and Embedded Systems. Mobile HCI, pp. 256-270, 2003.

[CM04] Carter, S., Mankoff, J.: Challenges for Ubicomp Evaluation. Technical Report UC Berkeley, 2004.

[CMK+07] Carter, S., Mankoff, J., Klemmer, S. R., & Matthews, T.: Exiting the cleanroom: on ecological validity and ubiquitous computing. Journal of Human-Computer Interaction, Issue 22, No. 1/2, 2007.

[CMN83] Card, S. K., Moran, T. P., Newell, A.: The psychology of human-computer interaction. ISBN 0-898-59859-9, Lawrence Erlbaum Associates, 1983.

[CMS99] Card, S., Mackinley,S., Shneiderman, B.: Readings in information visualization: using vision to think. Morgan Kaufmann, ISBN 1-558-60533-9, 1999.

[Coo99] Cooper, A.: The Inmates are Running the Asylum: Why High-tech Products Drive Us Crazy and How to Restore the Sanity. ISBN 0-672-31649-8, 1999.

[CRC07] Cooper, A., Reimann, R., Cronin, D.: About Face 3: The Essentials of Interaction Design. John Wiley & Sons, ISBN 0-470-08411-1, 2007.

[CRW07] Courage, C., Redish, J., Wixon, D.: Task Analysis. pp.927-947, In: Sears, A., Jacko, J: The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications, Second Edition. Lawrence Erlbaum and Associates, ISBN 0-805-85870-9, 2006.

[CW03] Consolvo, S., Walker, M.: Using the experience sampling method to evaluate ubicomp applications. IEEE Pervasive Computing, Vol.2, No.2, pp.24-31, ISSN 1536-1268, 2003.

[DA04] Dey, A.K., Abowd, D.G.: Support for the Adapting Applications and Interfaces to Context. In: Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces: Multi-devices, Cross-platform and Context-awareness, ISBN 0-470-85444-8, 2004.

[DF06] Dittmar, A., Forbrig, P.: Task-Action Consistency in Multi-Device Systems. Computer Human Interaction (CHI'2006) Workshop "The Many Faces of Consistency in Cross-Platform Design", pp.23-28, Montreal, Kanada, 2006.

[DFA+04] Dix, A., Finlay, J., Abowd, G., & Beale, R.: Human-Computer Interaction. 3. Auflage, Prentice Hall, ISBN 0-130-46109-1, 2004.

[DIN98] DIN EN ISO 9241-11: Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten: Anforderungen an die Gebrauchstauglichkeit – Leitsätze. Beuth-Verlag, Berlin, 1998.

[DIN99] DIN EN ISO 13407: Benutzer-orientierte Gestaltung interaktiver Systeme. Beuth-Verlag, Berlin, 1999.

[DIN04] DIN EN ISO 6385: Grundsätze der Ergonomie für die Gestaltung von Arbeitssystemen. Berlin: Beuth, 2004.

[DIN06] DIN EN ISO 9241-110: Ergonomie der Mensch-System-Interaktion – Grundsätze der Dialoggestaltung. Beuth-Verlag, Berlin, 2006.

[DJA93] Dahlbäck, N., Jönsson, A., Ahrenberg, L.: Wizard of Oz Studies - Why and How, Proceedings of 1st International Conference Intelligent User Interfaces, ACM Press, pp. 193–200, 1993.

[DLD+07] Davidoff, S., Lee, M.K., Day, A.K., Zimmerman, J.: Rapidly exploring application design through Speed Dating. Conference on Ubiquitous Computing (Ubicomp 2007), pp.429-446, Innsbruck, Austria, 2007.

[DR99] Dumas, J., Redish, J.: A Practical Guide to Usability Testing. ISBN 1-84150-020-8, 1999.

[EM09] Eckl, R., MacWilliams, A.: Smart Home Challenges and Approaches to Solve Them: A Practical Industrial Perspective. Intelligent interactive assistance and mobile multimedia Computing (IMC 2009), pp.119-130, Rostock, Germany, 2009.

[FBA07] Feuerstack, S., Blumendorf, M., Albayrak, S.: Prototyping of Multimodal Interactions for Smart Environments based on Task Models. AMI'07 Workshop on Model-Driven Software Engineering, Darmstadt, Germany, 2007.

[FCK07] Fleury, P., Cuřín, J., Kleindienst, J.: SitCom - Development Platform for Multimodal Perceptual Services. 3rd international Conference on industrial Applications of Holonic and Multi-Agent Systems: Holonic and Multi-Agent Systems For Manufacturing, Regensburg, Germany, 2007.

[FDM03] Forbrig, P., Dittmar, A., Müller, A.: Adaptive Task Modelling: From Formal Models to XML Representations. pp.169-192, Kapitel 12 in: Seffah, A., Javahery, H. (Eds.): Multiple User Interfaces, ISBN 0-470-85444-8, John Wiley & Sons, 2003.

[FDR+04] Forbrig, P.; Dittmar, A.; Reichart, D.; Sinnig, D.: From Models to Interactive Systems - Tool Support and XIML. IUI/CADUI 2004, Funchal, Portugal, 2004.

[Fer07] Ferscha, A.: Informative Art Display Metaphors. Human Computer Interaction International (HCII 2007), pp.82-92, Beijing, China, 2007.

[FKO+02] Fleck, M., Frid, M., Kindberg, T., O'Brian-Strain, E., Rajani. R., Spasojevic, M.: From Informing to Remembering: Ubiquitous Systems in Interactive Museums. IEEE Pervasive Computing, Vol. 1, No. 2., pp. 11-19, 2002.

[For07] Forbrig, P.: Objektorientierte Softwareentwicklungs mit UML. Carl Hanser Verlag, 2007.

[GDP99] Gaver, W.H., Dunne, A., Pacenti, E.: Cultural probes. ACM Interactions, Vol. 6, No. 1, pp. 21-29, 1999.

[GFF+07] Giersich, M., Forbrig, P., Fuchs, G., Kirste, T., Reichart, D., Schumann, H.: Towards an Integrated Approach for Task Modeling and Human Behavior Recognition. Human Computer Interaction International (HCII 2007), pp.1109-1118, Beijing, China, 2007.

[GFS05] Griethe, H., Fuchs, G., Schumann, H.: A Classification Scheme for Lens Techniques. WSCG'2005, Plzen, Czech Republic, 2005

[Gie09] Giersich, M.: Concept of a Robust & Training-free Probabilistic System for Online Intention Analysis in Teams. Dissertation, Universität Rostock, 2009.

[GGB+03] Gulliksen, J., Göransson, B., Boivie, I., Blomkvist, S., Persson, J., Cajander, Å: Key principles for user-centred systems design. Behaviour and Information Technology, Vol. 22, No. 6, pp. 397-409, 2003.

[GQM+09] Golovchinsky, G., Qvarfordt, P., van Melle, B., Carter, S., Dunnigan, T.: DICE: designing conference rooms for usability. Computer-Human Interaction (CHI 2009), p.1015-1024, 2009.

[Gre06] Greenfield, A.: Everyware: the dawning age of ubiquitous computing. ISBN 0-321-38401-6, New Riders Publishing, 2006.

[Hei09] Heider, T.: A Unified Distributed System Architecture for Goal-based Interaction with Smart Environments. Dissertation, Universität Rostock, 2009.

[Her09] Herczeg, M.: Software-Ergonomie: Theorien, Modelle und Kriterien für gebrauchstaugliche interaktive Computersysteme. ISBN 3-486-58725-0, 2009.

[HHF+06] Halloran, J., Hornecker, E., Fitzpatrick, G., Weal, M., Millard, D., Michaelides, D., Cruickshank, D., De Roure, D.: Unfolding understandings: co-designing UbiComp In Situ, over time. 6th Conference on Designing interactive Systems (DIS '06), pp.109-118, University Park, USA, 2006.

[HK02] Herfet, T., Kirste, T.: EMBASSI: Elektronische Multimediale Bedien- und Service Assistenz, White Paper. Online: http://www.embassi.de/publi/whitepaper/White_Paper-V1.1.pdf, 2002.

[HK07] Heider, T., Kirste, T.: Automatic vs. manual multi-display configuration: a study of user performance in a semi-cooperative task setting. 21st British HCI Group Annual Conference on HCI, pp.43-46, Lancaster, United Kingdom, 2007.

[HKB+06] Hartmann, B., Klemmer, S. R., Bernstein, M., Abdulla, L., Burr, B., Robinson-Mosher, A., Gee, J.: Reflective physical prototyping through integrated design, test, and analysis. Symposium on User interface Software and Technology (UIST '06), pp.299-308, Montreux, Switzerland, 2006.

[HL99] Helfrich,B., Landay, J. A.: QUIP: quantitative user interface profiling. Verfügbar unter: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1. 37. 2367 (Zugriff: 17.01.2009), 1999.

[HMW+03] Hutchinson, H., Mackay, W., Westerlund, B., Bederson, B. B., Druin, A., Plaisant, C., Beaudouin-Lafon, M., Conversy, S., Evans, H., Hansen, H., Roussel, N., Eiderback, B.: Technology probes: inspiring design for and with families. Proceedings of the ACM CHI'03 Conference on Human Factors in Computing Systems, CHI Letters, Vol. 5, No.1, 2003.

[Hol05] Holtzblatt, K.: Customer-centered design for mobile applications. Journal on Personal and Ubiquitous Computing Vol.9, No.4, pp 227-237, Personal and Ubiquitous Computing, 2005.

[HPS+02] Holm, R., Priglinger, M., Stauder, E., Volkert, J., Wagner, R.: Automatic data acquisition and visualization for usability evaluation of VR systems. Eurographics, 2002.

[HR98] Hackos, J. T., Redish, J. C.: User and task analysis for interface design. ISBN 0-471-17831-6, Wiley, 1998.

[HR00] Hilbert, D, Redmiles, D.: Extracting Usability Information from User Interface Events. ACM Computing Surveys, Vol.32, No.4, pp.384-421, 2000.

[HRR+08] Harper, R., Rodden, T., Rogers, Y., Sellen, A.: Being Human - Human Computer Interaction in the year 2020. Microsoft Research, ISBN: 0-9554761-1-2, 2008.

[IH01] Ivory, M., Hearst, M.: The State of the Art in Automating Usability Evaluation of User Interfaces. ACM Computing Surveys, Vol. 33, No. 4, 2001, pp. 470–516, 2001.

[ILB+05] Intille, S. S., Larson, K., Beaudin, J. S., Nawyn, J., Tapia, E. M., Kaushik, P.: A living laboratory for the design and evaluation of ubiquitous computing technologies. Computer-Human Interaction (CHI 2005), Portland, USA, pp.1941-1944, 2005.

[Int02] Intille, S. S.: Designing a Home of the Future. IEEE Journal on Pervasive Computing, Vol.1, No.2, pp.80-86, 2002.

[IR02] Ikonen, V., Rentto, K.: Scenario Evaluations for Ubiquitous Computing - Stories Come True?, Ubicomp 2002 - Workshop on User-Centered Evaluation of Ubiquitous Computing Applications, 2002.

[IRC05] Ieronutti, L., Ranon, R., Chittaro, L.: High-Level Visualization of Users' Navigation in Virtual Environments. INTERACT 2005, Berlin, Deutschland, 2005

[ISO88] ISO, "Information Processing Systems - Open Systems Interconnection - LOTOS - A Formal Description Based on Temporal Ordering of Observational Behaviour", ISO/IS 8807, ISO Central Secretariat, 1988.

[JJ89] Johnson, P., Johnson, H.: Knowledge Analysis of Tasks: Task analysis and specification for human-computer systems. In: Downton, A. (ed.): Engineering the Human Computer Interface, pp.119-144, McGraw-Hill, 1989.

[JJW95] Johnson, P., Johnson, H., Wilson, S.: Rapid Prototyping of User Interfaces Driven by Task Models. In: Scenario-Based Design, pp. 209-246, London, UK, Wiley, 1995.

[JLM+08] Jourde, F., Laurillau, Y., Moran, A., Nigay, L.: Towards Specifying Multimodal Collaborative User Interfaces: A Comparison of Collaboration Notations. Interactive Systems, Design, Specification and Verification (DSV-IS 2008), pp.281-286, Kingston, Canada, 2008.

[JS07] Jaimes, A., Sebe, N.: Multimodal human-computer interaction: A survey. Comput. Computer Vision and Image Understanding, Vol.108, No.1-2, pp.116-134, 2007.

[KA92] Kirwan, B., Ainsworth, L. K.: A guide to task analysis. ISBN 0-748-40058-4, Taylor & Francis, 1992.

[Kar97] Karat, J.: Evolving the scope of user-centered design. Communications of the ACM, Vol.40, No.7, pp.33-38, 1997.

[KCJ08] Kim, H., Choi, J., Ji, Y.: Usability Evaluation Framework for Ubiquitous Computing Device. Third International Conference on Convergence and Hybrid Information Technology, pp.164-170, 2008.

[KGC+08] Kimani, S., Gabrielli, S., Catarci, T., Dix, A.: Designing for Tasks in Ubiquitous Computing: Challenges and Considerations. Kapitel 7, In Mostefaoui, S. K. (ed.): Advances in Ubiquitous Computing: Future Paradigms and Directions. ISBN 1-599-04840-6, IGI Publishing, 2008.

[KV04] Koskela, T., Väänänen-Vainio-Mattila, K.: Evolution towards smart home environments: empirical evaluation of three user interfaces. Personal Ubiquitous Computing, Vol. 8, No. 3-4, pp. 234-240, 2004.

[LBV+06] Luyten, K., Van den Bergh, J., Vandervelpen, Ch., and Coninx, K.: Designing distributed user interfaces for ambient intelligent environments using models and simulations. Computers & Graphics Vol. 30, No. 5, pp. 702-713, 2006.

[LHL07] Li, Y., Hong, J. I., Landay, J. A.: Design Challenges and Principles for Wizard of Oz Testing of Location-Enhanced Applications. IEEE Pervasive Computing, Vol. 6, No. 2, pp. 70-75, 2007.

[Lin94] Lindgaard, G.: Usability and System Evaluation. ISBN 0-412-46100-5, Chapman & Hall, 1994.

[LLB07] Lewandowski, A., Lepreux, S., Bourguin, G.: Tasks Models Merging for High-Level Component Composition. In: Jacko, Julie A. (ed.) HCI International 2007 - 12th International Conference, Beijing, China, pp. 1129-1138, 2007.

[LM07] López-Jaquero, V., Montero, F.: Comprehensive Task and Dialog Modelling. Human Computer Interaction International (HCII 2007), pp.1149-1158, Beijing, China, 2007.

[Luy04] Luyten, K.: Dynamic User Interface Generation for Mobile and Embedded Systems with Model-Based User Interface Development, Dissertation, University Limburg, 2004.

[LV04] Limbourg, Q., and Vanderdonckt, J. Comparing Task Models for User Interface Design, pp. 135-154, In: Diaper, D., and Stanton, N.A. (eds.): The handbook of task analysis for human-computer interaction, Lawrence Erlbaum Associates, 2004.

[Mae94] Maes, P.: Agents that reduce work and information overload. Communications of the ACM, Vol.37, No.7, pp.30-40, 1994.

[MCK+08] Maly, I., Curin, J., Kleindienst, J., Slavik, P.: Creation and Visualization of User Behavior in Ambient Intelligent Environment. 12th Conference on Information Visualisation 2008, pp.497-502, 2008.

[MCW+02] McGee, D.R., Cohen, P.R., Wesson, R.M., Horman, S.: Comparing Paper and Tangible, Multimodal Tools. Computer Human Interaction (CHI 2002). pp.407-414, 2002.

[MDH+03] Mankoff, J., Dey, A. K., Hsieh, G., Kientz, J., Ames, M., Lederer, S.: Heuristic evaluation of ambient displays. Proceedings of the ACM CHI 2003 Conference on Human Factors in Computing Systems, CHI Letters, Vol. 5, No.1, pp.169-176, 2003.

[MJG08] Menon, V., Jayaraman, B., Govindaraju, V.: Biometrics Driven Smart Environments: Abstract Framework and Evaluation. 5th international Conference on Ubiquitous intelligence and Computing, Oslo, Norway, pp.75-89, 2008.

[MMS+07] Mikovec, Z., Maly, I., Slavik, P., Curin, J.: Visualization of users' activities in a specific environment. 39th Winter Simulation Conference, pp.738-746, Washington D.C., USA, 2007.

[MN06] Martin, M., Nurmi, P.: A generic large scale simulator for ubiquitous computing, Annual International Conference on Mobile and Ubiquitous Systems, pp. 1-3, 2006.

[Moo65] Moore, G. E.: Cramming more components onto integrated circuits. Electronics, Vol. 38, No. 8, pp. 114–117, 1965.

[MPH+97] Moran, T., Palen, L., Harrison, S., Chiu, P., Kimber, D., Minneman, S., van Melle, W., Zellweger, P.: “I’ll get that off the audio”: A case study of Salvaging Multimedia Meeting Records. CHI 1997, pp.202-209, 1997.

[MPS02] Mori, G., Paternó, F., Santoro, C.: CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. IEEE Transactions Software Engineering, Vol. 28, pp. 797-813, 2002.

[MPS03] Mori, G., Paternò, F., Santoro, C.: Tool Support for Designing Nomadic Applications. International Conference on Intelligent User Interfaces (IUI 2003), Miami, USA, pp.141-148, 2003.

[MS06] Malý, I., Slavík, P.: Towards Visual Analysis of Usability Test Logs. Task Models and Diagrams (TAMODIA 2006), Hasselt, Belgium, pp.25-32, 2006.

[MST+01] Makela, K., Salonen, E. P., Turunen, M., Haluinen, J., Raisamo, R.: Conducting a Wizard of Oz Experiment on a Ubiquitous Computing System Doorman. Proceedings of the International Workshop on Information Presentation and Natural Multimodal Dialogue, Verona, 2001.

[MY07] Mulder, S., Yaar, Z.: The User Is Always Right: A Practical Guide to Creating and Using Personas for the Web. New Riders Publishing, ISBN 0-321-434536, 2007.

[NSK+08] Neely, S., Stevenson, G., Kray, C., Mulder, I., Connelly, K., Siek, K.: Evaluating Pervasive and Ubiquitous Systems. IEEE Pervasive Computing, Volume 7, Issue 3, pp.85-88, 2008.

[Nie90] Nielsen, J.: Big paybacks from 'discount' usability engineering. IEEE Software Vol.7, No.3, pp.107-108, 1990.

[Nie93] Nielsen, J.: Usability Engineering. ISBN 0-12-518406-9, Morgan Kaufmann, San Francisco, 1993.

[Nie94] Nielsen, J.: Heuristic evaluation. In J. Nielsen & R. L. Mack (Eds.). Usability Inspection Methods, pp. 25-62, New York: John Wiley & Sons, 1994.

[Nie07] Nielsen, L.: Ten Steps To personas. Journal of HCI Vistas, Vol.3, 2007. Online available at: http://www.hceye.org/HCInsight-Nielsen.htm.

[NLB09] Nebe, K., Leuchter, S., Beck, A.: Integration von Software Engineering und Usability Engineering. In: S. Fischer, E. Maehle & R. Reischuk (Hrsg.), Informatik 2009, Im Focus das Leben, pp. 342-347, 2009.

[NM90] Nielsen, J., Molich, R.: Heuristic evaluation of user interfaces. Proceedings of ACM CHI'90 Conference on Human Factors in Computing Systems, Methodology, pp. 249–256, 1990.

[NL04] Niemelä, E., Latvakoski, J.: Survey of requirements and solutions for ubiquitous software. Mobile and Ubiquitous Multimedia, pp.71-78, 2006.

[Nor86] Norman, D. A.: Cognitive engineering. In D. A. Norman and S. W. Draper (Eds) User Centered Systems Design, Lawrence Erlbaum Associates Inc., 1986.

[Nor02] Norman, D.: The Design of everyday things. Perseus Books, ISBN 0-465-06710-7, 2002.

[NRG06] Nichols, J., Richter, K., Gajos, K.: The Many Faces of Consistency in Cross-Platform Design: A Whitepaper. Computer Human Interaction (CHI'2006) Workshop "The Many Faces of Consistency in Cross-Platform Design", pp.9-18, Montreal, Kanada, 2006.

[NW02] Nitschke, J., Wandke, H.: GUIDEAS: Guidance for Developing Assistance. Humboldt-Spektrum, Heft 1, pp.34-38, 2002.

[NYT+06] Nishikawa, H., Yamamoto, S., Tamai, M., Nishigaki, K., Kitani, T., Shibata, N., Yasumoto, K., Ito, M.: UbiREAL: Realistic Smartspace Simulator for Systematic Testing. Proc. of the 8th International Conference on Ubiquitous Computing (UbiComp2006), pp. 459-476, 2006.

[One04] O'Neill, E.: TATUS: A Ubiquitous Computing Simulator. Dissertation, The University of Dublin, Trinity College, 2004.

[PA06] Pruitt, J., Adlin, T.: The Persona Lifecycle: Keeping People in Mind Throughout Product Design. Morgan Kaufmann, ISBN 0-125-66251-3, 2006.

[Pat99] Paternò, F.: Model-Based Design and Evaluation of interactive applications. Springer, ISBN 1-852-33155-0, 1999.

[Pat08] Paternò, F.: Towards Universal Declarative User Interface Definition Languages. http://www.w3.org/2008/10/mbui/W3C%20Incubator-paterno.pdf, 2008.

[PBF09] Propp, S., Buchholz, G., Forbrig, P.: Integration of Usability Evaluation and Model-based Software Development. Journal on Advances in Engineering Software, Elsevier, 2009.

[PBW+04] Preuveneers, D., Van den Bergh, J., Wagelaar, D., Georges, A., Rigole, P., Clerckx, T., Berbers, Y., Coninx, K., Jonckers, V., De Bosschere, K.: Towards an Extensible Context Ontology for Ambient Intelligence. Second European Symposium, EUSAI 2004, Eindhoven, The Netherlands, pp.148-159, 2004.

[PF09] Propp, S., Forbrig, P.: Integrating Usability into Model-based Development of Smart Environments. Interact 2009, 4th Workshop on Software and Usability Engineering Cross-Pollination: Usability Evaluation of Advanced Interfaces, Uppsala, Sweden, 2009.

[Pos09] Poslad, S.: Ubiquitous Computing: Smart Devices, Environments and Interactions. ISBN 0-470-03560-9, 2009.

[PP02] Paganelli, L., Paternò, F.: Intelligent analysis of user interactions with web applications. 7th international Conference on intelligent User interfaces (IUI 2002), San Francisco, USA, 2002.

[PRD07] Poppe, R.W., Rienks, R., van Dijk, B.: Evaluating the Future of HCI: Challenges for the Evaluation of Emerging Applications. Artifical Intelligence for Human Computing, Springer, pp.234-250, 2007.

[PRS07] Paternò, F., Russino, A., Santoro, C.: Remote evaluation of Mobile Applications. Task Models and Diagrams (TAMODIA 2007), pp.155-168, Toulouse, France, 2007.

[PS01] Paternò, F., Santoro, C.: The ConcurTaskTrees Notation for TaskModelling. Technical Report of Guitar Project, 2001.

[Rab89] Rabiner, L.R.: A tutorial on Hidden Markov Models and selected applications in speech recognition. Proceedings of the IEEE, Vol.77, No.2, pp. 257-286, 1989.

[Rav03] Ravindranath Pandian, C.: Software Metrics: A Guide to Planning, Analysis, and Application. Auerbach Publications, ISBN 0-849-31661-8, 2003.

[RDW+05] Reilly, D., Dearman, D., Welsman-Dinelle, M., Inkpen, K.: Evaluating Early Prototypes in Context: Trade-offs, Challenges, and Successes. IEEE Pervasive Computing, Vol.4, No.4, pp.42-50, 2005..

[RF08] Reichart, D., Forbrig, P.: Transactions in Task Models. Task Models and Diagrams (TAMODIA 2008), pp.299-304, Pisa, Italy, 2008.

[RFD04] Reichart, D.; Forbrig, P.; Dittmar, A.: Task Models as Basis for Requirements Engineering and Software Execution. Task Models and Diagrams (TAMODIA 2004), pp. 51-58, Prague, Czech Republic, 2004.

[RHJ06] Ryu, H., Hong, G., James, H.: Quality assessment technique for ubiquitous software and middleware. Research Letters in the Information and Mathematical Sciences, Vo.9, pp.13-88, ISSN 1175-2777, 2006.

[RNB08] Rienks, R., Nijholt, A., Barthelmess, P.: Pro-active meeting assistants: attention please!. AI & Soiety, Special Issue: Social intelligence design: a junction between engineering and social sciences, Vol.23, No.2, pp.213-231, 2008.

[RKV+09] Rieffel, E., Kimber, D., Vaughan, J., Gattepally, S., Shingu, J.: Interactive Models from Images of a Static Scene. Computer Graphics and Virtual Reality (CGVR ‘09), Las Vegas, USA, 2009.

[RL77] Rodin, J., Langer, E.: Long-Term Effects of a Control-Relevant Intervention with the Institutional Aged. Journal Personality and Social Psychology, Vol. 35, No. 12, pp. 897–902, 1977.

[Rub94] Rubin, J., Handbook of usability testing, Wiley technical communication library, Hudson, T. (ed.), 1994.

[RYC+07] Rashidi, P., Youngblood, G. M., Cook, D., Das, S.:Inhabitant guidance of smart environments. Conference on Human-Computer Interaction, pp.910-919, 2007.

[SAK+02] Scholtz, J., Arnstein, L., Kim, M., Kindberg, T., Consolvo, S.: User-Centered Evaluations of Ubicomp Applications. IRS-TR-02-006, 2002.

[SC04] Scholtz, F., Consolvo, S.: Towards a Discipline for Evaluating Ubiquitous Computing Applications. Intel Research Technical Report, IRS-TR-04-004, 2004.

[Sch00] Schmidt, A.: Implicit Human Computer Interaction Through Context. Personal Technologies, Vol. 4, No. 3, pp. 191–199, 2000.

[SCH09] Silva, J. L., Campos, J. C., Harrison, M. D.: An infrastructure for experience centered agile prototyping of ambient intelligence. SIGCHI Symposium on Engineering interactive Computing Systems (EICS 2009), Pittsburgh, USA, 2009.

[Sea95] SEARS, A: AIDE: A step toward metric-basedinterface development tools. In Proceedings of the Eighth ACM Symposium on User Interface Software and Technology (Pittsburgh, PA, November), pp. 101–110. New York, NY: ACM Press, 1995.

[She02] Shneiderman, B.: Leonardo's laptop. ISBN 0-26269-299-6, MIT Press, 2002.

[Shi07] Shirehjini, A.A.N.: A Multidimensional Classification Model for the Interaction in Reactive Media Rooms. Human Computer Interaction International (HCII 2007), pp. 431-439, 2007.

[SH05] Shirehjini, A.A.N., Klar, F.: 3DSim: rapid prototyping ambient intelligence, Conference on Smart objects and ambient intelligence: innovative context aware services: usages and technologies, Grenoble, France, 2005.

[SL02] Sinha, A.K., Landay, J.A.: Embarking on Multimodal Interface Design. International Conference on Multimodal Interfaces (ICMI 2002), Pittsburgh, USA, pp. pp.355-360, 2002.

[SM03] Saha, D., Mukherjee, A.: Pervasive computing: A paradigm for the 21st century. IEEE Computer, pp. 25-31, 2003.

[SPG89] Scapin, D. L., Pierret-Golbreich, C.: Towards A Method For Task Description: MAD. In: Berlinguet, L., Berthelette, D. (editors), Work With Display Units 89, pp. 371-380, Elsevier Science Publishers B.V., 1989.

[STC] Society for Technical Communication, Usability and User Experience Community: Usability severity codes. www.stcsig.org/usability/resources/toolkit/usability%20severity%20codes.doc.

[SWF+07] Sinnig, D., Wurdel, M., Forbrig, P., Chalin, P., Khendek, F.: Practical Extensions for Task Models. Task Models and Diagrams (TAMODIA 2007), pp.42-55, Toulouse, France, 2007.

[SY98] Stanton, N., Young, M.: Is utility in the eye of the beholder? A study of ergonomics methods. Applied Ergonomics, Vol.29, No.1, pp. 41–54, 1998.

[Tec05] Technology Review: Usability Trend, 2005.

[TS06] Trapp, M., Schmettow, M.: Consistency in use through Model based User Interface Development. Workshop at CHI'2006, Montreal, Canada, 2006.

[Ubi09] Ubisense Echtzeit-Lokalisierungssystem. http://www.ubisense.net (Zugriff am: 17.01.2009).

[Uhl99] Uhl, A.: Evaluation. Stimmer, F. (Hrsg.): Lexikon der Sozialpädagogik und der Sozialarbeit, 4. Auflage, Oldenbourg, München, 1999.

[UW95] Uehling, D. L., Wolf, K: User action graphing effort (UsAGE). Conference on Human Factors in Computing Systems, pp.290-291, Denver, USA, 1995.

[VLB96] Van der Veer, G. C., Van der Lenting, B. F., Bergevoet, B. A. J.: GTA: Groupware task analysis-modeling completity, Acta Psychological, Vol.91, pp. 297–322, 1996.

[VMS+02] Vredenburg, K., Mao, J.-Y., Smith, P. W., Carey, T.: A survey of user-centered design practice. Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 471-478. ACM Press, 2002.

[Wal06] Waldner, J-.B.: Nano-informatique et Intelligence Ambiante. , ISBN 2-74-621516-0, Hermes Science, London, 2006.

[Wan08] Wandke, H.: Die Zukunft der Usability-Forschung im Lichte allgegenwärtiger Computertechnologie. artop Newsletter, 3/2008, ISSN 1613-8058, 2008.

[WDS08] Wurdel, M., Sinnig, D., Forbrig, P.: Task-based Development Methodology for Collaborative Environments, Engineering Interactive Systems 2008, Pisa, Italy, 2008.

[Wei91] Weiser, M.: The computer for the 21st century. Scientific American, pp. 94-104, 1991.

[WB96] Weiser, M., Brown, S.: The coming age of calm technology. http://www.johnseelybrown.com/calmtech.pdf, 1996.

[WFR05] Wolff, A., Forbrig, P., Reichart, D., Tool Support for Model-Based Generation of Advanced User Interfaces. MDDAUI 2005, Montego Bay, Jamaica, 2005.

[Wil03] Wilson, C.: Defining, Categorizing, and Prioritizing Usability Problems. UPA 2003 Idea Market, http://www.upassoc.org/conferences_and_events/upa_conference/2004/call/sample/ideamarket_usability%20problems_wilson.doc, 2003.

[WPF08] Wurdel, M., Propp, S., Forbrig, P.: HCI-Task Models and Smart Environments. Human-Computer Interaction Symposium (HCIS 2008), Mailand, Italy, pp.21-32, 2008.

[WRA05] Weber, W., Rabaey, J., Aarts, E.: Ambient Intelligence. ISBN 978-3-540-23867-6, Springer, 2005.

[WRL+94] Wharton, C., Rieman, J., Lewis, C. & Polson, P.: The Cognitive Walkthrough Method: A Practitioner´s Guide. pp. 105-140, In: Nielsen, J., Mack, & R.L. (Editors): Usability Inspection Methods. New York: John Wiley, 1994.

[WS09] Waibel, A. & Stiefelhagen, R. (ed.): Computers in the Human Interaction Loop, Springer, 2009.

[WSF08] Wurdel, M., Sinnig, D., Forbrig; P.: Task-based Development Methodology for Collaborative Environments, Engineering Interactive Systems 2008, Pisa, Italy, pp.118-125, 2008.

[WSH01] Woodruff, A., Szymanski M.H., Hurst A.: The Conversational Role of Electronic Guidebooks. International Conference on Ubiquitous Computing, Atlanta, USA, pp.187-208, 2001.

[WTN00] Whittaker, S., Terveen, L., and Nardi, B. A.: Let's stop pushing the envelope and start addressing it: a reference task agenda for HCI. Human-Computer Interaction, Vol.15, No.2, pp.75-106, 2000.

[Wur09] Wurdel, M.: Towards an Holistic Understanding of Tasks, Objects and Location in Collaborative Environments. Human Computer Interaction International (HCII 2009), San Diego, USA, 2009.

[YLL+08] You, F., Luo, H., Liang, Y., and Wang, J.: Prototyping and Evaluation for Smart Home Controller Based on Chinese Families Behavior Analysis. 8th Asia-Pacific Conference on Computer-Human interaction, Seoul, Korea, pp.437-445, 2008.

[ZAB06] Ziefle, M., Arning, K., Bay, S.: Cross-platform consistency and cognitive compatibility: The importance of users' mental model for the interaction with mobile devices. In: Richter, K., Nichols, J., Gajos, K., Seffah, A. (Eds.): The Many Faces of Consistency in Cross-Platform Design. CEUR-WS, Vol 198, pp. 75-81, 2006.

[Zim05] Zimmerman, J.: Video Sketches: Exploring pervasive computing interaction designs. Journal IEEE Computing, Vol.4, No.4, pp.91-94, 2005.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe. Die den benutzten Quellen oder Hilfsmitteln wörtlich oder inhaltlich entnommenen Stellen sind in der Arbeit als solche erkenntlich.

Ich erkläre weiterhin, dass ich mich zuvor weder an der Universität Rostock noch an einer anderen Universität um einen Doktorgrad beworben habe.

Rostock, 28.02.2010 Stefan Propp

Lebenslauf

Name:	Stefan Propp
Geburtsdatum:	10.02.1982 in Schwerin
09/1988 – 07/1992	Grundschule Friedrich-Engels-Schule, Schwerin
09/1992 – 07/2000	Gymnasium (Abschluss: Abitur) Johann-Wolfgang-von-Goethe-Gymnasium, Schwerin
08/2000 – 06/2001	Zivildienst Integrative Kindertagesstätte „Bärenkinder“
10/2001 - 10/2006	Studium der Wirtschaftsinformatik (Abschluss: Dipl.-Wirt.-Inf.) Wirtschafts- und Sozialwissenschaftliche Fakultät Universität Rostock
04/2006 – 10/2006	Diplomarbeit Thema: „Modellierung semantischer Beziehungen zwischen Entitäten der technischen Ebene und der Geschäftslogik einer Plattform für Geschäftsprozesse“ SAP AG (Abteilung: Research)
11/2006 – 01/2010	Promotionsstudium Thema: „Usability-Evaluation in intelligenten Umgebungen“ Lehrstuhl für Softwaretechnik / Graduiertenkolleg 1424 Universität Rostock

Thesen zur Dissertation

1. Die Planung und Entwicklung eines physischen Smart Environments sind teuer, insbesondere die Auswahl von Quantität und Qualität der Sensoren, Geräte und Softwarekomponenten.

2. Mit fortschreitender Entwicklung wird es immer aufwendiger ein bestehendes Usability-Problem zu beheben (1:10:100-Regel) So kann ein Problem, das bereits früh auf Modellebene gefunden wird typischerweise erheblich einfacher behoben werden, als wenn auf Basis des Modells Sensoren und Aktoren montiert wurden und diese neu montiert werden müssen. Daher sollte die Evaluation möglichst früh beginnen und eine iterative Entwicklung begleiten.

3. Die modellgetriebene Entwicklung von Smart Environments erlaubt die iterative Entwicklung und die schnelle Evaluation auf mehreren Stufen der Entwicklung, insbesondere der Anforderungsanalyse-, Design- und Implementierungsphase.

4. Das entwickelte virtuelle Smart Environment animiert je nach Entwicklungsfortschritt die jeweils entwickelten Aufgaben-, Orts- und Nutzermodelle. Damit ist nun ein interaktives Durchlaufen („walkthrough“) dieser Modelle möglich und erlaubt ein direktes iteratives Verbessern.

5. Viele Menschen haben keine Erfahrung mit der Interaktion in Smart Environments. Es fällt ihnen daher schwer, während der Anforderungsanalyse Wünsche an ein zu entwickelndes System zu benennen und gleichzeitig abzuschätzen, wie sinnvoll diese während der späteren Benutzung sein werden.

6. Ein Ausweg ist die Durchführung von „Wizard of Oz“-Experimenten, die den künftigen Nutzern in realer Umgebung demonstrieren wie sich die gewünschte Assistenz „anfühlt“. Das virtuelle Smart Environment kann zur Durchführung des Experimentes dienen.

7. Nachdem erste Softwarekomponenten implementiert sind, bspw. die Intentionserkennung, ist ein Testen nur dann möglich, wenn die notwenigen Kontextdaten der entsprechenden Sensoren vorhanden sind.

8. Um die Kosten des Aufbaus eines physischen Smart Environments zu vermeiden, können diese Kontextdaten im virtuellen Environment künstlich erzeugt werden, um Sensorkonfigurationen zu testen. Damit stehen Kontextdaten zur Verfügung, die das Testen der Intentionserkennung erlauben.

9. Nachdem das physische Smart Environment aufgebaut ist, können Nutzerstudien über die Modelle hinausgehende Facetten evaluieren, bspw. Ästhetik und Haptik der Lösung. Authentische Studien realer Interaktionen („in-situ“) sind dabei aufschlussreich. Die aufgezeichneten Sensordaten, einschließlich Video- und Audiomitschnitten beinhalten allerdings persönliche Daten und sind aus Sicht des Datenschutzes als kritisch zu bewerten.

10. Ein Ausweg ist die anonymisierte Darstellung der Nutzerinteraktionen im virtuellen Smart Environment, das von konkreten Nutzern und Geräten abstrahiert und nur einen anonymen Ausschnitt der Realität abbildet.