Institut für Informatik
der Technischen Universität München

Requirements Engineering kontextsensitiver Anwendungen

Wassiou Olarewaju Sitou

Institut für Informatik
der Technischen Universität München

Requirements Engineering kontextsensitiver Anwendungen

Wassiou Olarewaju Sitou

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. Hans Michael Gerndt
Prüfer der Dissertation:	
	1. Univ.-Prof. Dr. Dr. h.c. Manfred Broy
	2. Univ.-Prof. Gudrun J. Klinker, Ph.D.

Die Dissertation wurde am 20. November 2008 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 30. April 2009 angenommen.

How lucky Adam was.
He knew when he said a good thing, nobody had said it before.

Samuel Langhorne Clemens *alias* Mark Twain

30.11.1835 – 21.04.1910

Kurzfassung

Mit dem ständig wachsenden Umfang angebotener Funktionalität bei Software-Systemen und deren Verwendung in unterschiedlichen Situationen des täglichen Lebens, steigt auch der Bedarf an komfortabler Nutzung dieser Systeme. Im Bereich des Context-Aware Computing werden kontextsensitive Anwendungen (engl. context-aware applications) entwickelt, welche diesem steigenden Bedarf an Komfort Rechnung tragen und dem Nutzer die Verwendung dieser Systeme so unkompliziert wie möglich gestalten. Allerdings steigt dadurch die Komplexität der Entwicklung der Anwendungen zusätzlich. Derzeitige Entwicklungsmethoden sind den gestiegenen Anforderungen allerdings nicht zufriedenstellend gewachsen, da Konzepte zur Beherrschung der anfallenden Systemkomplexität fehlen.

Kontextsensitive Anwendungen reagieren auf die Eingaben der Nutzer unter Berücksichtigung relevanter Informationen der Nutzungssituation. Beispielsweise wird ein Nutzer automatisch über das Eintreffen einer Nachricht informiert. Für die Auswahl einer angemessenen Form der Benachrichtigung werden unter anderem seine Vorlieben, seine Fähigkeiten und die Gegebenheiten seines aktuellen Aufenthaltsorts berücksichtigt. Die relevanten Informationen, auch Kontext genannt, werden mit Hilfe von Sensorik gemessen. Um den Kontext strukturiert zu erfassen, wird ein Kontextmodell verwendet. Basierend auf den Ausprägungen des Kontextmodells werden Nutzungssituationen identifiziert. Dieses Vorgehen bietet die Grundlage dafür, dass Systemreaktionen automatisch an erkannte Nutzungssituationen angepasst werden können. Durch die Automatisierung der Anpassung wird der Nutzer von expliziten Interaktionen mit dem System entlastet. Eine Herausforderung des Requirements Engineering besteht darin, das zugehörige Anpassungskonzept zu entwerfen. Dafür sind diejenigen Informationen zu identifizieren, welche für eine Anpassung der Systemreaktion relevant sind. Die integrierte Spezifikation der Systemreaktion in Abhängigkeit der gegebenen Nutzungssituation ist im Requirements Engineering vorzunehmen.

In der vorliegenden Arbeit wird ein Rahmen entworfen, anhand dessen die relevanten Informationen über die Nutzungssituationen kontextsensitiver Anwendungen systematisch modelliert werden können. Grundlage hierfür ist eine Kategorisierung der charakteristischen Merkmale einer Nutzungssituation in die Bedürfnisse und Fähigkeiten der Nutzer, die durchgeführten Aktivitäten und die Gegebenheiten der Einsatzumgebung. Zwischen diesen Kategorien existieren Wechselwirkungen, welche ebenfalls für die Charakterisierung einer Nutzungssituation von Bedeutung sind. Das Konzept für die methodische Erarbeitung von Kontextinformationen der einzelnen Kate-

gorien und deren Integration in ein umfassendes Kontextmodell im Rahmen des Requirements Engineering bildet einen Schwerpunkt dieser Arbeit. Anhand einer integrierten Kontext-Szenarioanalyse erfolgt ausgehend von Nutzerbedürfnissen eine iterative Erhebung der Anforderungen an die zu entwickelnde Anwendung, und damit einhergehend die Modellierung des Kontextes.

Die Ergebnisse dieser Arbeit sind der Rahmen zur Modellierung des Kontextes und die Methodik zur integrierten Erhebung, Analyse und Spezifikation von Anforderungen und Nutzungssituationen kontextsensitiver Anwendungen. Die Konzepte werden anhand von zwei Fallstudien im Bereich des Context-Aware Computing demonstriert. Die Fallstudien behandeln zum einen einen kontextsensitiven Terminplaner und zum anderen ein kontextsensitives Scheibentönungssystem.

Dankesworte

An dieser Stelle möchte ich mich bei all jenen herzlich bedanken, die mich in den letzten Jahren unterstützt und mir das Gelingen dieser Arbeit ermöglicht haben.

Ein besonderer Dank geht an Prof. Manfred Broy, der es mir ermöglichte, an diesem aktuellen Thema zu arbeiten. Für die wissenschaftliche Betreuung, die großen Freiräume und das entgegengebrachte Vertrauen möchte ich mich besonders bedanken. Ein besonderer Dank ergeht ebenfalls an Prof. Gudrun Klinker für die inhaltlichen Diskussionen und die Übernahme der Rolle der Zweitgutachterin.

Großer Dank gebührt den Kollegen des Kompetenzzentrum für *Mobility & Context-Awareness*, Dr. Michael Fahrmair, Christian Leuxner und Bernd Spanfelner, die interessante Impulse für meine Arbeit lieferten, mir sehr viel Freiraum in den gemeinsamen Projekten einräumten und für ein angenehmes Arbeitsklima sorgten. Des Weiteren danke ich den Kollegen des Kompetenzzentrums für *Requirements Engineering*, insbesondere meiner Büro-Kollegin Dr. Eva Geisberger sowie den Kollegen Dr. Sabine Rittmann und Johannes Grünbauer für die wertvollen Diskussionen zum Thema Requirements Engineering. Bei dem gesamten Lehrstuhl für *Software & Systems Engineering*, insbesondere bei unserem Sekretariat, möchte ich mich für die gute Atmosphäre und die unkomplizierte Zusammenarbeit während den letzten Jahren bedanken.

Für das aufmerksame Durchlesen und die wertvollen Kommentare zu der endgültigen Fassung meiner Arbeit möchte ich mich bei Dr. Katharina Spies herzlich bedanken.

Nicht zuletzt danke ich sehr herzlich meiner großen Familie in aller Welt, insbesondere in Togo und Deutschland, für den finanziellen, seelischen und moralischen Beistand während meines Studiums und meiner Promotion. Ganz besonders möchte ich meiner Frau Sarah Pierrette sowie meinen Kindern Faruk und Malik, für ihre Geduld und Rücksicht während den letzten Jahren danken.

Inhaltsverzeichnis

Kapitel 1

Einführung

One can easily lose, if one goes to much for gain

Theodor Fontane

Inhaltsangabe

1.1 Motivation

Die rasant wachsenden technologischen Fortschritte und die damit verbundene Integration von Software-Systemen in viele Bereiche unseres täglichen Lebens, wie zum Beispiel Laptops, Mobiltelefone, PDAs, Uhren und Haushaltsgeräte, führen zu einem höheren Grad an Flexibilität und an Multifunktionalität der Systeme. In Zukunft werden Systeme benötigt, die sich den Bedürfnissen und Wünschen des Benutzers anpassen und dabei lediglich die minimal notwendige direkte Interaktion verlangen.

Mit dem Ubiquitous Computing [Weiser, 1991] und dem daraus motivierten Context-Aware Computing [Schilit et al., 1994] wird das Ziel verfolgt, die Nutzung eines Software-Systems in möglichst vielen Situationen eines Nutzers zu ermöglichen [Dey, 2000], [Fahrmair, 2005]. Dies beinhaltet auch Situationen, in denen der Nutzer aufgrund mangelnder Interaktionsfähigkeiten nicht die Möglichkeit hat, explizit mit dem System zu interagieren [Schmidt, 2002], [Samulowitz, 2002]. Hinter diesen Konzepten steckt die Idee eines Systemverständnisses, bei dem die Bedürfnisse und Wünsche des Benutzers stärker als bisher im Mittelpunkt stehen. Dem Benutzer wird suggeriert, das Verhalten des Systems stets an seine Bedürfnisse und die aktuelle Nutzungssituation anpassen zu können.

Das Ziel, die Systeme in möglichst vielen Situationen eines Benutzers zu nutzen, erfordert einen Mechanismus, der es erlaubt, die Systeme auch in Situationen zu nutzen, in

denen die Interaktionsfähigkeit des Benutzers beschränkt ist [Schilit, 1995]. Es wird eine situationsgerechte automatische Anpassung der Interaktionen des Systems mit dem Nutzer angestrebt. Das Konzept der *Kontextadaption* stellt eine Möglichkeit dar, situationsgerechte Anpassungen der Systeme technisch zu realisieren. Dabei wird die Interaktion zwischen System und Nutzer zunächst mit Informationen über den aktuellen Kontext der Systemnutzung angereichert. *Kontext* (in dieser Arbeit auch *Nutzungskontext* bzw. *Einsatzkontext*) bezeichnet in diesem Zusammenhang die hinreichend genaue Charakterisierung der Nutzungssituationen eines Systems anhand von Informationen, welche für die Adaption der Interaktionen zwischen dem System und dem Nutzer relevant und von dem System wahrnehmbar sind. Dabei ist unter *Adaption* die allgemeine Fähigkeit zu verstehen, sich in einer bestimmten Situation nach den durch die Umwelt vorgegebenen Verhältnissen zu richten. Folglich bezeichnet *Kontextadaption* die automatische Anpassung des beobachtbaren Verhaltens (bzw. des inneren Zustandes oder der Struktur) eines Systems an seinen Kontext.

Ein interaktives System wird dann kontextadaptiv (kurz adaptiv) genannt, wenn es über Sensorik relevante Informationen über die Einsatzumgebung aufnimmt, und diese zur Adaption seiner Interaktionen mit dem Nutzer verwendet. Demnach reagieren kontextadaptive Systeme nicht nur auf Eingaben, die explizit von dem Nutzer getätigt werden, sondern auch auf Ereignisse, die das System selbständig zur Laufzeit über ausgestattete Sensorik erkennt. Wir verwenden die Begriffe *kontextadaptive* und *kontextsensitive* Systeme (engl. context-aware systems[1]) in dieser Arbeit synonym. Dabei betrachten wir die Adaption stets aus einer gegebenen Nutzerperspektive.

Zum aktuellen Kontext der Nutzung des Systems gehören unter anderem die gegenwärtige Aufgabe des Nutzers, seine momentane Lage und Fähigkeiten sowie die Gegebenheiten der Einsatzumgebung des Systems. Diese Mischung erfordert eine systematische Erfassung des Kontextes anhand dessen die Adaption des Systems durchgeführt wird.

1.2 Analyse und Themenstellung

Kontextsensitive Anwendungen, wie sie in dieser Arbeit betrachtet werden, haben die Eigenheit, ein adaptives Verhalten abhängig von der Perspektive des Nutzers zu zeigen. Auf die gleiche Eingabe des Nutzers reagiert die Anwendung unterschiedlich. Sie verhält sich also aus Sicht des betrachteten Nutzers „nicht-deterministisch“. Dieser Nicht-Determinismus lässt sich dadurch auflösen, dass die Interaktion zwischen der Anwendung und dem Nutzer zusätzlich durch Sensoreingaben (dem Kontext der Anwendung) beeinflusst wird (siehe Abbildung 1.1).

Der Kontext, auf Basis dessen die Interaktion zwischen der Anwendung und dem Nutzer angepasst wird, hat allerdings unterschiedliche Aspekte, insbesondere *die Einsatzumgebung, den Nutzer und die Aktivitäten des Nutzers bei der Nutzung der Anwendung*, ihre Wechselwirkungen und Änderungen über die Zeit. Die Modellierung des Kontextes stellt eine zentrale Herausforderung bei der Entwicklung der Anwendungen dar. Durch das hohe Maß an Flexibilität und Multifunktionalität der Systeme sowie die

[1] *context-aware* bedeutet so viel wie Kontext-bewusst. In wissenschaftlichen Arbeiten werden allerdings die Begriffe kontextadaptiv bzw. kontextsensitiv verwendet.

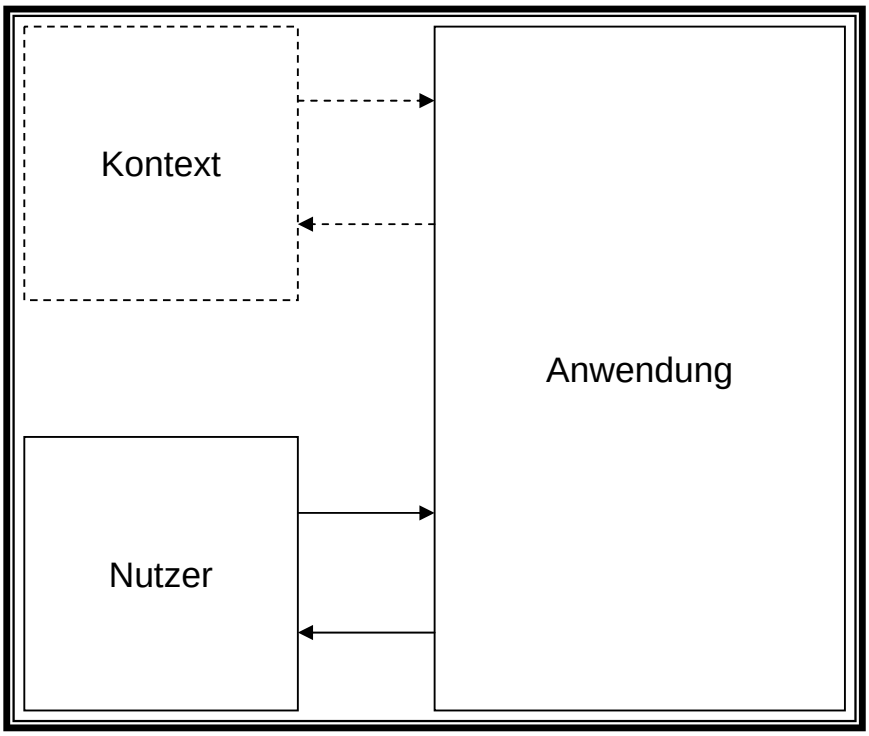

Abbildung 1.1: Nutzer, Kontext und Anwendung

dadurch bedingte Autonomie kontextsensitiver Anwendungen, erhöht sich außerdem die Gefahr, dass die Anwendungen ein aus Nutzersicht nicht-nachvollziehbares bzw. unerwünschtes Verhalten zeigen. Solche Divergenzen zwischen beobachteten und erwünschtem bzw. erwartetem Verhalten der Anwendungen sind hauptsächlich darin begründet, dass die relevanten Kontextinformationen – ein essentieller Bestandteil der Kontextadaption – aufgrund fehlender bzw. unzureichender Modelle nicht eindeutig bestimmt sind. Darüber hinaus erfolgt die Abbildung der Bedürfnisse und Wünsche des Nutzers sowie des Kontextes der Anwendungen auf ihr Verhalten nicht systematisch.

Obwohl mittlerweile die notwendige technische Infrastruktur ausgereift ist, existieren für die Entwicklung kontextsensitiver Anwendungen keine geeigneten Methoden [Davies et al., 2005]. Bei der Ermittlung, Analyse und Erfassung der Anforderungen kontextsensitiver Anwendungen werden ihre spezifischen Eigenheiten kaum berücksichtigt. Die verwendeten Methoden und Modelle sind nur bedingt geeignet. Notwendige Methoden existieren nur ansatzweise [Sutcliffe et al., 2005] und behandeln lediglich einzelne Aspekte wie zum Beispiel die Nutzung [Pham, 2001, Samulowitz, 2002], die Sammlung [Samulowitz, 2002] oder auch die technische Erfassung [Schmidt, 2002] von Kontextinformationen.

Das Requirements Engineering (RE), eine frühe Phase in der Software- und Systementwicklung, hat das Ziel, die Anforderungen an das zu entwickelnde System zu ermitteln, zu analysieren und eindeutig und vollständig zu spezifizieren. Es ist ein methodisch gestütztes Aufstellen von Anforderungen mittels Analyse der Problemstellung. Dabei werden unter Anforderungen sowohl qualitative als auch funktionale Aspekte des betrachteten Systems verstanden. Im Idealfall umfasst eine Anforderungsspezi-

fikation eine ausführliche Beschreibung des Zwecks, des geplantem Einsatzes in der Praxis sowie des geforderten Funktionsumfangs eines Systems.

Ziel dieser Arbeit ist es, bewährte Methoden für das Requirements Engineering zu untersuchen und sie auf die besonderen Herausforderungen kontextsensitiver Anwendungen anzupassen. Diese Methoden, eingebunden in eine RE-Methodik, werden eine systematische Entwicklung der Anforderungen kontextsensitiver Anwendungen ermöglichen.

Der in dieser Arbeit verfolgte Ansatz stellt die Bedürfnisse und Wünsche des Nutzers in den Vordergrund und liefert einen Rahmen zur Konstruktion des Modells des Nutzungskontextes einer kontextsensitiven Anwendung. Die Methodik zielt darauf ab, die Anforderungen einer kontextsensitiven Anwendung verknüpft mit ihren Nutzungssituationen zu spezifizieren. Hierbei werden insbesondere Erfahrungen aus praktischen Realisierungen derartiger Systeme abstrahiert und systematisiert.

1.3 Einordnung in die aktuelle Forschung

Requirements Engineering wird als iterativer, systematischer und interdisziplinärer Prozess verstanden, der mit allen beteiligten Stakeholdern abgestimmt wird. Dabei wird das Ziel verfolgt, Spezifikationen zu erarbeiten, welche die Ziele der Nutzer genügen [Nuseibeh und Easterbrook, 2000]. Die Ziele werden analysiert und verfeinert, und anschließend auf Anforderungen abgebildet, welche letztendlich durch das System realisiert werden. Obwohl Konzepte zum Erarbeiten von Anforderungen unterschiedlicher Gruppen von Stakeholdern beispielsweise in Form von User Studies oder Viewpoint-Ansätzen bereits existieren [Kotonya und Sommerville, 1998], [Robertson und Robertson, 1999], [Tuunanen, 2005], fehlen bislang dedizierte Konzepte zur Analyse von Anforderungen für Anwendungen aus dem Bereich des Context-Aware Computing.

Im Context-Aware Computing wird das Ziel verfolgt, die Bedürfnisse und Wünsche des Nutzers selbst in solchen Situationen zu erfüllen, in denen der Nutzer diese z.B. aufgrund äußerer Umstände (Steuern eines Fahrzeugs) nicht explizit äußern kann[2]. Die Befriedigung eines Bedürfnisses ist mit der Erfüllung der aus diesem Bedürfnis resultierenden Ziele verbunden. Ein Ziel bezeichnet hierbei ein in der Zukunft liegender angestrebter, vorgestellter bzw. beschriebener Zustand. Ein wichtiges Aufgabenfeld zielorientierter Ansätze im Requirements Engineering ist die Modellierung von Zielen sowie ihre Verfeinerung und Abbildung auf fein granuläre Anforderungen. Selbstverständlich sollte der Wert von zielorientierten Ansätzen im RE kontextsensitiver Anwendungen nicht vernachlässigt werden, zumal Ziele eine wichtige Rolle in der Grundidee solcher Anwendungen spielen. In [Kolos-Mazuryk et al., 2006] wurden drei repräsentative RE-Ansätze aus der Gruppe der zielorientierten Ansätze [van Lamsweerde, 2001] miteinander verglichen, und ihre Eignung für die Entwicklung kontextsensitiver Anwendungen untersucht. Bei den drei Ansätzen handelt es sich um dem KAOS-Ansatz [Dardenne et al., 1993], [Darimont et al., 1997],

[2] Die Begriffe *Bedürfnis*, *Wunsch* und *Ziel* sind eng miteinander verbunden. Ein Bedürfnis ist das Verlangen oder der Wunsch, einem empfundenen oder tatsächlichen Mangel Abhilfe zu schaffen. Zur Erfüllung von Bedürfnissen werden Strategien angewandt. Wünsche unterscheiden sich von Bedürfnissen dadurch, dass sie bereits eine Konkretisierungsstufe in Richtung auf Strategien darstellen.

den GBRAM-Ansatz [Antón, 1997] und den I*-Ansatz[Yu, 1997]. Diese Studie ergab, dass zielorientierte Ansätze für die Entwicklung kontextsensitiver Anwendungen von großer Bedeutung sind. Zugleich wurde jedoch festgestellt, dass sich die betrachteten Ansätze für eine systematische Gewinnung, Analyse und Erfassung von Anforderungen an kontextsensitive Anwendungen unzureichend sind. Diese Ansätze bieten in ihrer derzeitigen Form keine Systematik zur Behandlung der in dieser Arbeit betrachteten *kontextbezogenen* Anforderungen.

Erste Ansätze zur Behebung dieses Defizits sind die Ansätze zur Behandlung von so genannten *deferred requirements*[3] [Fickas et al., 2005] und von kontextabhängigen Einflüssen auf Anforderungen, angeführt von dem Ansatz des *Clinical Requirements Engineering* [Fickas, 2005]. Ferner werden szenariobasierte Analysemethoden zur Spezifikation von personenbezogenen Zielen und deren potentiellen Änderungen aufgeführt. Ein Vertreter dieser Gruppe von Ansätzen ist das *Personal and Contextual Requirements Engineering – PC-RE* [Sutcliffe et al., 2006]. PC-RE ist ein Ansatz zur Analyse individueller und persönlicher Ziele sowie der Wirkung von Zeit und Einsatzbedingungen auf die persönlichen Ziele. Der Ansatz soll es ermöglichen, die Funktionalitäten eines Systems zu spezifizieren, personalisierbare Features zu entwickeln und entsprechende Anpassung des Systems durchzuführen. Hinter der Personalisierung von Features verbirgt sich ein Benutzermodell, das jedoch nicht näher spezifiziert ist. In Zusammenhang mit PC-RE wird Kontext lediglich mit Ort (engl. *location*) und Zeitpunkt des Einsatzes gleichgestellt.

Mit dem steigenden Interesse an kontextsensitiven Anwendungen ist die Variation von Anforderung an ein System nicht mehr ausschließlich abhängig von den Nutzern, sondern auch von den durchgeführten Aufgaben und der Einsatzumgebung zu betrachten. In ortsbasierten (engl. location-based bzw. location-aware) Anwendungen hängt die Gültigkeit von bestimmten Anforderungen beispielsweise vom Ort der Nutzung ab [Cheverst et al., 2000, Abowd und Mynatt, 2000]. Die Auswirkung des Orts einer Nutzung auf die Anforderungen und somit auf das Verhalten der Anwendung wurde anfänglich in dem *Inquiry Cycle* Ansatz erforscht [Potts et al., 1994]. Dort wird aufgeführt, dass die Akzeptanz der Ausgabe eines Systems durch den Ort seines Einsatzes beeinflusst wird. Dabei werden Änderungen über die Zeit, abgesehen von Belangen der Evolution der Anforderungen, nicht explizit modelliert.

Die Verwendung des Kontextbegriffes lässt sich im Context-Aware Computing jedoch nicht auf den Ort der Nutzung reduzieren (“there is more to context than location”) [Schilit et al., 1994, Schmidt et al., 1999b]. Ein weiterer wichtiger Aspekt des Kontextes ist beispielsweise der kulturelle Hintergrund. Obwohl allgemein anerkannt ist, dass kulturelle Aspekte einen maßgeblichen Einfluss auf die Nutzung von Produkten haben, wissen wir bislang wenig über die kulturellen Einflüsse auf Systemanforderungen [Norman, 2004]. Datenschutz-Anforderungen an einen Bankautomat zum Beispiel sind ganz anders im Osten als im Westen [De Angeli et al., 2004].

Eine Herausforderung bei der Entwicklung kontextsensitiver Anwendungen ist die Festlegung, wie sich die Anwendung in welcher Situation verhalten soll. Ähnliche Herausforderungen treten bei der Bildung von Systemkonfigurationen in Produktli-

[3] *Defer* ist der englische Begriff für verschieben, verzögern, zurückstellen. *Defer* in Zusammenhang mit Anforderungen oder Zielen (*deferred requirement* oder *deferred goal*) wird verwendet, um Anforderungen oder Ziele zu bezeichnen, die während der Analyse gesammelt, deren Realisierung bzw. Erreichung jedoch bis zum Auftritt passender Ereignisse verzögert werden.

nienentwicklungen auf [Bayer et al., 1999]. Im Gegensatz zum Gebiet der Produktlinienentwicklung, in dem lediglich die Anpassung des Systems noch vor der Laufzeit erforscht wird, wird im Requirements Engineering kontextsensitiver Anwendungen der Fokus auf die Anpassung zur Laufzeit gelegt.

Die jeweiligen Einflüsse des Nutzungskontextes einer Anwendung (z.B. Benutzer bzw. Benutzergruppe, Aufgaben, Ziele, Einsatzumgebung) sollten bei der Entwicklung kontextsensitiver Anwendungen wohl durchdacht sein. Solange diese unterschiedlichen Aspekte von Kontext und deren Wechselwirkungen und Änderung über die Zeit nicht systematisch untersucht und erfasst werden können, werden kontextsensitive Anwendungen häufig mit der Problematik des unerwünschten Verhaltens (Unwanted Behavior – UB) [Fahrmair et al., 2006b] konfrontiert sein. Eine systematische Erfassung des Nutzungskontextes bleibt somit eine besondere Herausforderung bei der Gewinnung, Analyse und Spezifikation von Anforderungen an kontextsensitive Anwendungen.

1.4 Ergebnisse der Arbeit

In der vorliegenden Dissertation wird eine Methodik für das Requirements Engineering kontextsensitiver Anwendungen (RE-CAWAR – *Requirements Engineering for Context-Aware Applications*) entwickelt. Der Ablauf der Methodik ist in Abbildung 1.2 zusammengefasst – einschließlich der zugehörigen Artefakte, d.h. der Ergebnisse, die bei der Durchführung der methodischen Schritte erzielt werden.

Das Requirements Engineering kontextsensitiver Anwendungen beginnt mit einer Scoping-Aktivität, welche in drei Schritten durchgeführt wird. Im Rahmen der Scoping-Aktivitäten erfolgen die Festlegung der Anwendungsdomäne, die Identifikation der Stakeholder und ihrer Ziele sowie die Definition der Vision des zu entwickelnden Systems.

Nach der anfänglichen Scoping-Aktivität erfolgt im Rahmen der Modellierung eine integrierte Kontext- und Szenario-Analyse. Dabei werden Anwendungsszenarien, funktionale Anforderungen an das System und der Kontext der Nutzung der Anwendung modelliert. Die Modellierung des Kontextes im Rahmen der RE-CAWAR Methodik erfolgt auf Basis des in Abbildung 1.3 dargestellten Grundkonzeptes.

Anschließend erfolgt die Konsolidierung der erzielten Ergebnisse. Aus der Methodik resultieren ein umfassendes Kontextmodell der Anwendung und eine Liste der funktionalen Anforderungen an das System (*User Requirements Specification*). Die Anforderungen werden jeweils mit Kontextinformationen verknüpft, welche die einzelnen Nutzungssituationen der Anwendung charakterisieren. Zugleich wird für jedes Paar, bestehend aus einer funktionalen Anforderung und den charakteristischen Informationen einer Nutzungssituation, ein illustrierendes Szenario angegeben. Das Szenario beschreibt die Nutzung der Anwendungsfunktion, welche der betroffenen funktionalen Anforderung genügt – in der angegebene Nutzungssituation.

Die so beschriebenen Tupel aus funktionalen Anforderungen, charakteristischen Informationen der Nutzungssituationen sowie illustrierenden Szenarien werden in der RE-CAWAR Methodik *Contextual Requirements Chunks (CRCs)* genannt.

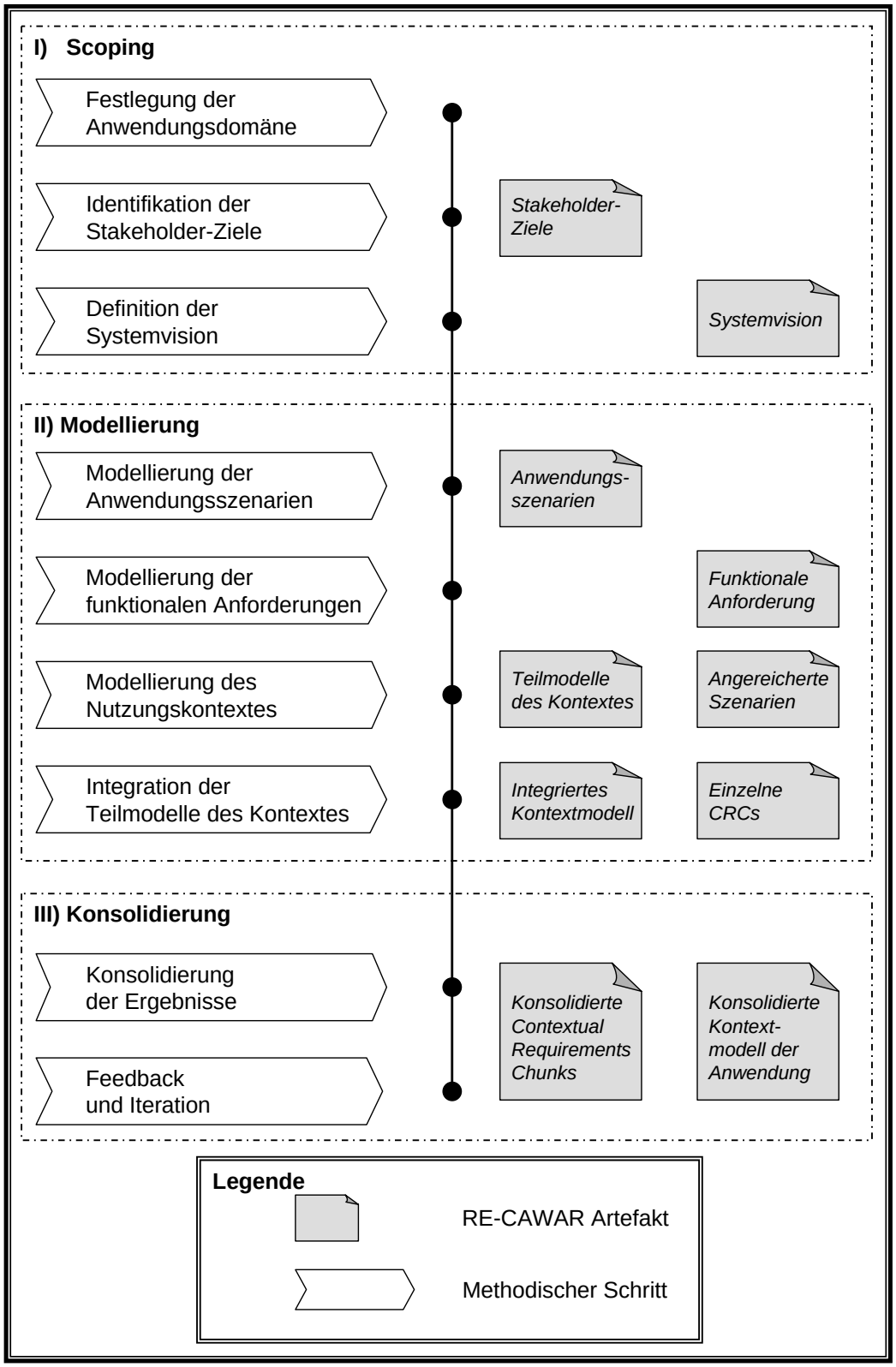

Abbildung 1.2: Schritte und Artefakte der RE-CAWAR Methodik

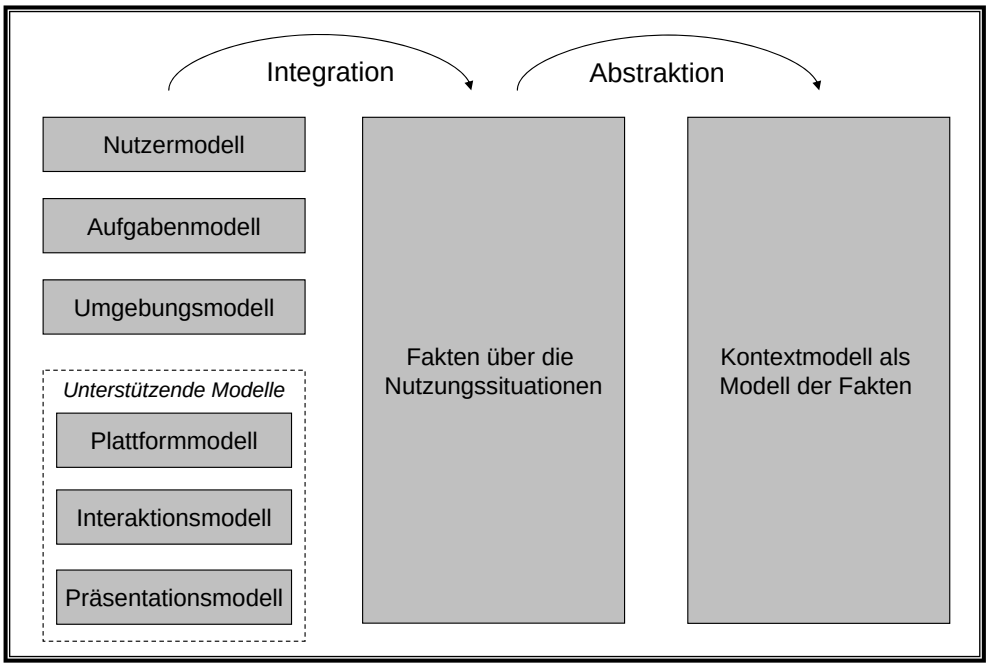

Abbildung 1.3: Grundkonzept der Kontextmodellierung in RE-CAWAR

Zusammenfassend sind die Ergebnisse der vorliegenden Arbeit:

1. **Analyse der Herausforderungen kontextsensitiver Anwendungen**
   Kontextsensitive Anwendungen aus Sicht des betrachteten Nutzers auf die gleiche Eingabe des Nutzers unterschiedlich reagieren. Sie weisen demnach aus Sicht des Nutzers ein nicht-deterministisches beobachtbares Verhalten auf. Grund hierfür ist die Hinzunahme relevanter Informationen über die Nutzungssituation der Anwendung – mit dem Ziel zu entscheiden, welches beobachtbare Verhalten als Reaktion auf die Nutzereingabe angebracht ist. Diese relevanten Informationen über die Situationen der Nutzung der Anwendung werden Kontextinformationen genannt. Eine Herausforderung des RE kontextsensitiver Anwendungen ist das Vermeiden von unerwünschtem Systemverhalten, welches aufgrund unzureichender Kontextmodelle, überzogener Nutzererwartungen oder auch unpassender Adaptionsentscheidungen auftritt.
2. **Konzeption eines Frameworks zur Kontextmodellierung**
   Das Kontextmodell einer Anwendung setzt sich potenziell aus einem Nutzermodell, einem Aufgabenmodell, einem Umgebungsmodell, sowie einer Reihe unterstützender Modelle (Plattformmodell, Interaktionsmodell und Präsentationsmodell) zusammen. Eine detaillierte Beschreibung dieser Modelle sowie geeignete Modellierungstechniken zu deren Erstellung werden in der vorliegenden Arbeit aufgeführt. Ein Konzept zur Integration der aufgeführten Teilmodelle des Kontextes in ein umfassendes Kontextmodell wird vorgeschlagen.
3. **Entwicklung einer modellbasierten RE-Methodik (RE-CAWAR)**
   Ein wesentlicher Bestandteil der RE-CAWAR Methodik ist die integrierte Ziel-, Szenario- und Kontext-Analyse. Sie dient der Ermittlung, Analyse und Modellierung von Anforderungen kontextsensitiver Anwendungen. Dabei werden die

Anforderungen mit den Situationen gekoppelt, in denen die realisierenden Anwendungsfunktionen ersterer genutzt werden. Jedes Paar (Anforderung, Nutzungssituation) wird mit einem Szenario illustriert. Das resultierende Tupel nennen wir *Contextual Requirements Chunk, CRC* (Anforderung – Kontext – Szenario). Konsolidierte CRCs dienen als Grundlage für die Spezifikation der Systemfunktionen bei dem Übergang zwischen Analyse und Design oder für die Bildung von Konfigurationen kontextsensitiver Anwendungen im Design. Bei der Entwicklung einer kontextsensitiven Anwendung auf Basis der mit der RE-CAWAR Methodik modellierten Anforderungen und des Kontextmodells wird das Auftreten unerwünschten Systemverhaltens vermieden; zumindest wenn die Ursprünge des Auftretens auf ein unzureichendes RE zurückzuführen sind.

Die Ergebnisse dieser Arbeit sind anhand von zwei begleitenden Fallstudien validiert und gesichert. Bei den Fallstudien geht es sich um einen kontextsensitiven Terminplaner und ein kontextsensitives Scheibentönungssystem.

Nicht behandelte Fragestellungen dieser Arbeit

Die Entwicklung kontextsensitiver Anwendungen stellt jenseits der in dieser Arbeit behandelten Fragestellungen eine Reihe weiterer interessanten Fragen, von denen genannt werden:

1. **Welche Probleme erfordern eine kontextsensitive Lösung?**
   Die Entscheidung, ob eine kontextsensitive Anwendung anstelle einer nicht-kontextsensitiven als Lösung eines Problems erwogen werden sollte, ist bislang nicht ausreichend untersucht. Es gibt eine Reihe von Vermutungen, die darauf hindeuten, dass diese Fragestellung hochgradig interdisziplinär ist. Es gibt allerdings keine Systematik, mit der entschieden werden kann, wann Kontextsensitivität einer herkömmlichen Lösung vorzuziehen ist. Das beobachtbare Verhalten einer kontextsensitiven Anwendung ist nicht nur von der Nutzereingabe abhängig, sondern auch von den Kontextinformationen über den Nutzungssituationen. Die Kontextsensitivität macht an dieser Stelle nur dann Sinn, wenn der vorgesehene Kontext der Nutzung der Anwendung variabel ist. [Welsh und Sawyer, 2008] postuliert, dass kontextabhängige Variationen in den annehmbaren Kompromissen zwischen nicht-funktionalen Anforderungen, Indikatoren für Probleme sind, die eine kontextsensitive Lösung verlangen. Dieses Postulat bleibt allerdings noch mit einer großen Auswahl von Anwendungen zu belegen.
2. **Wie werden die Kontextinformation zur Laufzeit gewonnen?**
   Nach der Modellierung des Kontextes, wie sie in dieser Arbeit geschieht, ist zu entscheiden, welche technischen Sensoren einzusetzen sind, um die gewünschten Kontextinformationen zu gewinnen bzw. zu sammeln. Dabei ist anzumerken, dass nicht jede gewünschte Kontextinformation direkt gesammelt werden kann. Eine Interpretation der gesammelten Kontext-Daten ist u.U. notwendig und sogar kostengünstiger, um die gewünschte Kontextinformation zu erhalten (siehe [Krause, 2006]). Damit verbunden ist auch die Fragestellung der technischen Realisierung des Kontextmodells der Anwendung (siehe [Henricksen, 2003]). Darüber hinaus bringt die Kontextsensitivität eine Reihe von Implikationen für die Interaktion zwischen dem Nutzer und der Anwendung (siehe [Schmidt, 2002]).

3. **Wie werden Systemkonfigurationen gebildet?**
   Damit verbunden ist die Fragestellung, *welche Funktionen der Anwendung gehören zu welcher Konfiguration?* Mit der gekoppelten Aufstellung der Anforderungen und der Nutzungssituationen einer kontextsensitiven Anwendung ist ein wichtiger Schritt in diese Richtung bereits in den frühen Phasen der Entwicklung kontextsensitiver Anwendungen gemacht worden. Eine weitere Herausforderung ist die Bildung von Systemkonfigurationen, mit deren Hilfe die Struktur der Anwendung flexibel gehalten wird (siehe [Schilit, 1995]). Hierzu existiert eine Reihe von Arbeiten (siehe u.a. [Dey, 2000, Samulowitz, 2002, Fahrmair, 2005, Trapp, 2005]), welche die Fragestellung aus unterschiedlichen Perspektiven behandeln.

1.5 Aufbau der Arbeit

Die Arbeit gliedert sich in sechs Kapitel: *Einführung*, *Grundlagen*, *Herausforderungen*, *Kontextmodellierung*, *RE-CAWAR Methodik* und *Resümee*. Die Gliederung gestattet dem Leser das Überspringen bereits bekannter Kapitel. Relevante Begriffe werden bei ihrer erstmaligen Verwendung kurz erklärt. Sie können bei Bedarf aber auch im Glossar nachgeschlagen werden. Die Anwendung der vorgestellten Methodik des Requirements Engineering kann am Beispiel eines kontextsensitiven Scheibentönungssystems ebenfalls im Anhang nachgeschlagen werden. Leser, die mit der Thematik kontextsensitiver Anwendungen, deren Architektur und Herausforderungen bereits vertraut sind, können die entsprechenden Teile in den Kapiteln Grundlagen und Herausforderungen überspringen und den Schwerpunkt auf die Kapitel Kontextmodellierung und RE-CAWAR Methodik legen.

Kapitel 1: Einführung

In diesem einführenden Kapitel wird neben der Motivation der Arbeit und der Themenstellung eine Einordnung in die aktuelle Forschung inklusive einer Abgrenzung zu anderen Ansätzen vorgenommen. Darüber hinaus sind die wichtigsten Ergebnisse dieser Arbeit kurz zusammengefasst.

Kapitel 2: Grundlagen

Die Arbeit beschreibt eine Requirements Engineering Methodik zur Entwicklung kontextsensitiver Anwendungen, wie sie in den Bereichen Context-Aware Computing Anwendung finden. Hierzu wird in diesem Kapitel ein kurzer Überblick über die Themenschwerpunkte vorgestellt, in welche sich diese Arbeit eingliedert. Die nötigen Grundlagen zu den Themen Methoden und Methodik, Requirements Engineering und Context-Awareness sind an dieser Stelle aufgeführt. Darüber hinaus werden die begleitenden Fallstudien (der kontextsensitive Terminplaner und das kontextsensitive Scheibentönungssystem) in diesem Kapitel eingeführt.

Kapitel 3 Herausforderungen

Im Kapitel 3 werden die Herausforderungen, die sich bei der Entwicklung kontextsensitiver Anwendungen stellen, diskutiert und analysiert. Eine solche Herausforderung

besteht beispielsweise darin, den aus Sicht des Nutzers nicht-deterministischen Aspekt des Systemverhaltens aufzulösen. In der Tat können die betrachteten Anwendungen aus der Perspektive bestimmter Nutzer ein nicht-deterministisches Verhalten aufweisen. Dies liegt daran, dass das beobachtbare Verhalten der Anwendungen nicht nur von den expliziten Eingaben des Nutzers abhängt, sondern auch von Kontextinformationen, die über Sensorik gesammelt werden. Die Handhabung des dynamischen Konstrukts des *Nutzungskontextes* (kurz *Kontext*) stellt eine weitere Herausforderung dar. Kontext im Zusammenhang mit Adaptivität umfasst unterschiedliche Aspekte, deren ausführliche Untersuchung für ihre Modellierung von wichtiger Bedeutung ist. Zur Dynamik von Kontext gehört nicht nur seine Änderungen über die Zeit, sondern auch die Vielfältigkeit seiner Aspekte und ihre Beziehungen unter einander. Diese werden im einzelnen untersucht. Eine weitere Herausforderung kontextsensitiver Anwendungen liegt im Umgang mit dem gehäuften Auftreten *unerwünschten Verhaltens* (UB), welches aus automatischen Anpassungen an den Kontext resultiert. Beide Herausforderungen werden, zusammen mit dem *Nichtdeterminismus aus Sicht des Nutzers*, in diesem Kapitel diskutiert und analysiert. Eine genaue Charakterisierung von unerwünschtem Verhalten bei kontextsensitiven Anwendungen sowie eine Klassifizierung des Phänomens nach den Ursachen seines Auftretens basierend auf unterschiedlichen Beobachtungen werden aufgeführt.

Kapitel 4: Modellierung des Nutzungskontextes

Ein Ergebnis dieser Arbeit ist die Bereitstellung eines Rahmens zur strukturierten und systematischen Erfassung des Nutzungskontextes. Eine Voraussetzung für die strukturierte und systematische Erfassung des Nutzungskontextes ist die Erstellung eines Modells, in dem die relevanten Aspekte des Kontextes erfasst sind. Zu der Modellierung des Nutzungskontextes gehören die Identifikation der möglichen *Bestandteile des Kontextes* und die Auswahl *geeigneter Modellierungstechniken*. Die möglichen Bestandteile eines Kontextmodells sind: ein Benutzermodell (Benutzerdaten und Benutzungsdaten), ein Aufgabenmodell (Aufgaben, die der Nutzer durchführt), ein Umgebungsmodell (Umgebung, in der die Anwendung genutzt wird) und eine Reihe unterstützender Modelle (Plattformmodell, Interaktionsmodell, Präsentationsmodell). Angesichts der Vielfältigkeit des Kontextes erweist sich die Beschränkung auf eine einzige Modellierungstechnik als zu restriktiv. Wir schlagen daher eine Reihe von Techniken vor, anhand derer die einzelnen Bestandteile des Kontextes modelliert werden können. Auch die Integration dieser Modelle wird in diesem Kapitel behandelt.

Kapitel 5: RE-CAWAR Methodik

Im Kapitel 5 wird die Requirements Engineering Methodik für die Entwicklung kontextsensitiver Anwendungen (RE-CAWAR) aufgeführt. Die RE-CAWAR Methodik ist ein modellbasierter Ansatz für das RE kontextsensitiver Systeme. Kern der Methodik ist die Entwicklung eines integrierten Modells des Kontextes unter Berücksichtigung seiner unterschiedlichen Aspekte. Die Entwicklung des Modells erfolgt iterativ mittels einer integrierten Ziel-Szenario-Kontext-Analyse. Ausgehend von Geschäfts- und Nutzerzielen werden Anforderungen an das System anhand ziel- und szenariobasierter Ansätze erarbeitet. Gleichzeitig werden Szenarien beschrieben, um die Nutzungssituationen zu identifizieren und das Kontextmodell zu generieren. Ein Ergebnis der Anwendung der Methodik ist eine konsolidierte Liste funktionaler Anforde-

rungen, welche mit den jeweiligen Nutzungssituationen und illustrierenden Anwendungsszenarien verknüpft sind (*Contextual Requirements Chunks*). Ein weiteres Ergebnis der Methodik ist das umfassende Kontextmodell der zu entwickelnden Anwendung. Die einzelnen Schritte der RE-CAWAR Methodik sowie deren Zusammenspiel werden in diesem Kapitel aufgeführt und anhand geeigneter Beispiele aus den begleitenden Fallstudien illustriert.

Kapitel 6: Resümee

Im abschließenden Kapitel dieser Arbeit wird ein *Resümee* gezogen. Dabei werden die wichtigsten Ergebnisse noch einmal aufgegriffen und deren Vorteile diskutiert. Ein Ausblick, welcher offene Punkte und Weiterentwicklungen skizziert, rundet dieses Kapitel ab.

Anhang: Fallstudie, Gliederung eines Anforderungsdokuments kontextsensitiver Anwendungen und Glossar

Im Anhang A wird die praktische *Anwendung* der erzielten Ergebnisse in Form einer prototypischen Realisierung anhand einer der begleitenden Fallstudien demonstriert. Bei dieser Fallstudie handelt es sich um ein kontextsensitives Scheibentönungssystem[4].

Im Anhang B wird eine *Gliederung* vorgeschlagen, wie Anforderungsdokumente kontextsensitiver Anwendungen aufgebaut werden. Bei der vorgeschlagene Gliederung wird die Grundidee verfolgt, sowohl Anforderungen als auch deren zugehörige Nutzungssituationen gemeinsam (integriert) zu entwickeln. Der Vorschlag konzentriert sich – im Gegensatz zu Gliederungsvorschlägen wie etwa dem *Volere-Template* [Robertson und Robertson, 2007] oder dem IEEE-Standard 830 [IEEE, 1998] – auf die spezifischen Fragestellungen kontextsensitiver Anwendungen, und lässt Themen wie Qualitätseigenschaften oder noch allgemeiner nicht-funktionale Anforderung außen vor.

Des Weiteren ist ein Glossar der wichtigsten Begriffe dieser Arbeit im Anhang aufgeführt.

[4] Die zweite Fallstudie behandelt einen kontextsensitiven Terminplaner (Context Aware Task Scheduler). Sie wird abwechselnd mit der Fallstudie des Scheibentönungssystems für die Zwecke der Illustration der vorgestellten Konzepte und der erzielten Ergebnisse in der Arbeit verwendet.

Kapitel 2

Grundlagen

Only a wise person can solve a difficult problem.

Popular Akan Proverb

Inhaltsangabe

2.1 Einleitung

Die vorliegende Arbeit beschreibt eine Requirements Engineering (RE) Methodik zur Entwicklung kontextsensitiver Anwendungen, wie sie in dem Bereich Mobile und Context-Aware Computing Anwendung finden. In diesem Kapitel geben wir einen Überblick über die Themenschwerpunkte, in die sich diese Arbeit eingliedert.

Eine Methodik liefert Vorgaben für ein systematisches Vorgehen bei der Softwareentwicklung und besteht aus zusammengesetzten Methoden. Der Begriff Methodik umfasst sowohl einzelne Aktivitäten der Softwareentwicklung wie Analyse, Entwurf oder Programmierung als auch die Softwareentwicklung insgesamt [Rechenberg et al., 2002]. In der Softwareentwicklung spielen Methoden eine wichtige Rolle. Sie beziehen sich dabei auf einen Einsatzbereich und geben Richtlinien in Form von Techniken, Mitteln und Organisationsformen an. Auf Vorgehensmodelle, die die Grundlage zu den Methoden bilden und Entwicklungsprozesse in überschaubare Abschnitte aufteilen, gehen wir in Abschnitt 2.2 ein.

Die in dieser Arbeit entworfene Methodik ist speziell auf die Phase des Requirements Engineering im Prozess der Entwicklung kontextsensitiver Anwendungen zugeschnitten. In Abschnitt 2.3 gehen wir also auf die Grundlagen des RE ein. Dort werden, nach einem kurzen Rückblick über die Entstehung des RE, wichtige Begriffe erklärt. Darüber hinaus wird ein Überblick über Aktivitäten, die im RE durchgeführt werden, sowie aktuelle Ansätze des RE gegeben.

Ein drittes zentrales Themengebiet der Arbeit neben RE und Methoden ist das Themengebiet der Kontextadaption im Umfeld kontextsensitiver Anwendungen. Kontextsensitivität (*engl. Context Awareness*), wie sie in dieser Arbeit verwendet wird, impliziert nicht nur eine Wahrnehmung eines veränderten Kontextes, sondern auch eine entsprechende Anpassung diesbezüglich. Um die Anpassungsfähigkeit der Anwendungen hinsichtlich des jeweiligen Kontextes zu betonen, wird der Begriff *Kontextadaptivität* als Synonym zu Kontextsensititviät verwendet. Unter Kontextadaption verstehen wir die automatische Anpassung eines Systems an die aktuellen Gegebenheiten seiner Umgebung. Sie ist eine notwendige Technik zur Realisierung kontextsensitiver Anwendungen im Umfeld des Context-Aware Computing. Auf die Grundidee des Context-Aware Computing sowie die grundlegenden Begriffe rund um Kontextadaption gehen wir in Abschnitt 2.4 genauer ein. Dort beginnen wir mit einigen Erläuterungen zum Ubiquitous Computing, einem in der Forschung sehr verbreiteten Anwendungsgebiet der Kontextadaptivität. Wir gehen dann auf den Prozess der Kontextadaption ein, und präsentieren den konzeptuellen Aufbau kontextsensitiver Anwendungen, wie sie in dieser Arbeit betrachtet werden.

Die Illustration und die Evaluation der in dieser Dissertation entwickelten Konzepten erfolgen in Form von Fallstudien. Im abschließenden Abschnitt 2.5 werden die begleitenden Fallstudien eingeführt.

2.2 Methode und Vorgehensmodell

Wesentlicher Beitrag dieser Arbeit besteht darin, eine Methodik zu entwickeln, die Hilfestellung bei der Durchführung des Requirements Engineering kontextsensitiver Anwendungen bietet. Die Entwicklung einer Methodik erfordert die Untersuchung sowie die Zusammenstellung geeigneter Methoden. Ein mit dem Thema der Entwicklung von Methoden verwandtes Thema ist das der Vorgehensmodelle. In diesem Abschnitt werden wir uns mit entsprechenden Grundlagen zu den Themen Methoden und Vorgehensmodelle vertraut machen.

2.2.1 Methode

In dem Sprachgebrauch der Wissenschaft steht eine *Methode* für eine Art und Weise eines Vorgehens, eine Vorgehensweise zur systematischen Erlangung neuer Erkenntnisse. In der Softwaretechnik bezeichnet eine Methode eine Vorgehensweise bei der Erstellung (Planung und Entwicklung) von Software-Systemen. Der Entwurf, die Konstruktion und die Anpassung von Methoden, Techniken und Werkzeuge für die Entwicklung von Software-Systemen erfolgen auf einem Gebiet, das *Method Engineering* genannt wird (siehe [Brinkkemper, 1996]).

Eng verwandt mit dem Begriff Methode ist der Begriff Technik. Am häufigsten wird der Begriff Technik dazu verwendet, um allein Notationen zu bezeichnen. Beispiele für Techniken sind ER-Diagrammen zur Modellierung von Daten oder strukturierte Interviews-Schablonen zur Durchführung von Interviews. Techniken können beispielsweise danach klassifiziert werden, wie formal die zugrunde liegende Notation ist (z.B. natürlich sprachlich, strukturierte Graphiken) oder welche Entwicklungsaktivität sie unterstützen (z.B. Datenmodellierung, Prozessmodellierung, Interaktionsdesign).

Die eigentliche Entwicklung einer Methode kann in drei Phasen unterteilt werden, die iterativ durchlaufen werden (vgl. [Brinkkemper, 1996]):

1. *Verstehen des zu lösenden Problems*
   Es wird in erster Linie erwartet, dass das zu lösende Problem klar definiert und verstanden wird. Die Herausforderungen sind auf dieser Phase zu identifizieren. Die Phase wird auch Phase des *Problemverstehens* genannt.
2. *Erstellen einer Liste von Kriterien*
   Eine solche Kriterienliste dient dazu geeignete Methoden zur Lösung des Problems zu wählen bzw. zu bewerten. Eine Reihe definierter Kriterien werden z.B. dafür verwendet, eine initiale Menge von Methoden, die zur Lösung des Problems in Frage kommen, einzuschränken. Die Kriterien können aber auch dazu verwendet werden, um die Eignung der zu konstruierenden Methoden zu bewerten. Diese Phase der Methodenentwicklung wird auch Phase der *Problemanalyse* genannt.
3. *Entwurf einer Methode bzw. einer Sammlung von Methoden*
   In dieser Phase erfolgt der eigentliche Entwurf der Methode(n) zur Lösung des Problems. Diese Phase lässt sich wiederum in die Schritte *Methodenauswahl*, *Experimentieren* und *Validieren* unterteilen (siehe Abbildung 2.1). Der Schritt der Auswahl der initialen Methoden (Kandidaten von Methoden zur Lösung des Problems) wird mit der erstellten Kriterienliste unterstützt. Wenn eine Sammlung von Methoden erforderlich ist, werden mit den einzelnen Methoden zunächst separat und anschließend gebündelt experimentiert. Nach den Experimenten werden die Ergebnisse analysiert und die Methoden validiert. Der Validierung folgt die Aktualisierung bzw. Verfeinerung der initialen Mengen von Methoden. Die Phase des Entwurfs wird auch Phase der *Methodenkonstruktion* genannt.

Die Grenzen zwischen den einzelnen Phasen ist zwar nicht formal festgelegt, allerdings kann man leicht feststellen, in welcher Phase der Methodenentwicklung man sich gerade befindet. Der Übergang einer Phase zu einer anderen ist stufenlos; der Anfang einer neuen Phase kann dadurch festgestellt werden, dass der aktuelle Schwerpunkt geändert wird. Als Kriterium für den Übergang von der ersten auf die zweite

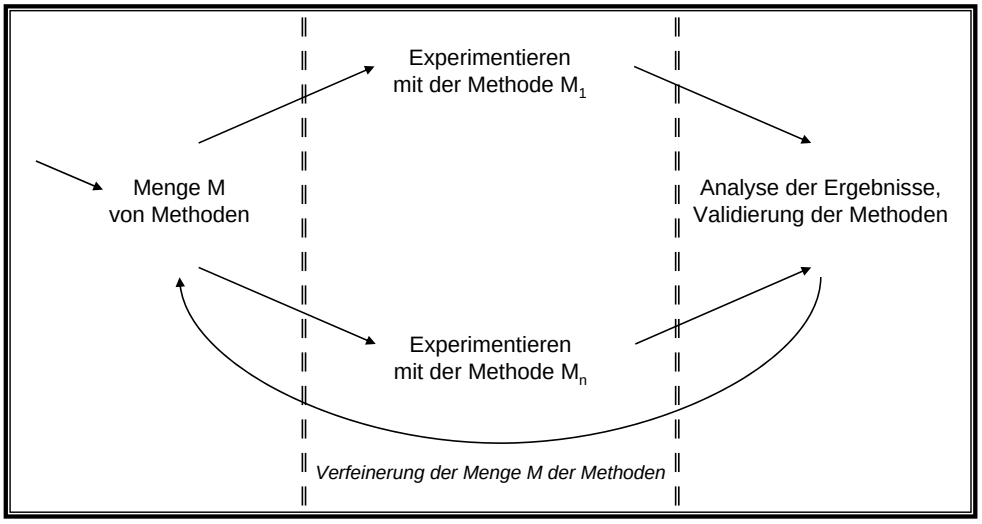

Abbildung 2.1: Strukturierter Entwurf von Methoden

Phase gilt, dass das Problem gut genug verstanden ist, um mit der Definition von Kriterien zur Auswahl bzw. Bewertung der Methoden zu beginnen. Spätestens bei der Durchführung von Experimenten befindet man sich in der Phase der Methodenkonstruktion.

2.2.2 Vorgehensmodell

Zur Lösung komplexer Aufgaben ist es notwendig, den Lösungsprozess in kleinere Einheiten zu gliedern. Ein Vorgehensmodell gibt eine Gliederung für den Prozess der Softwareentwicklung vor. Es regelt den Ablauf des Lösungsprozesses und unterteilt ihn in überschaubare Abschnitte, sodass eine schrittweise Planung, Durchführung, Entscheidung und Kontrolle ermöglicht wird (vgl. [Pomberger und Blaschek, 1996]). Diese Grundidee der Systemtechnik wird auch auf die Vorgehensweise bei der Entwicklung von Software angewandt. Demnach werden Softwareentwicklungsprojekte in Phasen unterteilt. In so genannten Phasenmodellen werden die einzelnen Phasen strukturiert hintereinander ausgeführt. In den objektorientierten Lebenszyklusmodellen stehen atomare Modellierungseinheiten mit Verhaltens- und Strukturbeschreibungen im Vordergrund.

Klassisch strukturierte Vorgehenweise

In den 70er Jahren ist eine Reihe unterschiedlicher Phasenmodelle und kombinierte Phase-/Tätigkeitsmodelle entstanden. Das bekannteste und älteste Modell dieser Reihe ist das Wasserfallmodell [Royce, 1970]. Die Grundidee bei diesen Modellen besteht darin, dass man ein Problem nur detailliert planen muss, um es bewältigen zu können. Eine sequenzielle Abarbeitung der Phasen ist ein typisches Merkmal dieser Modelle. Sie bewirkt allerdings, dass Fehler in den frühen Phasen, wie etwa in der Planung erst spät bemerkt werden. Außerdem werden Risiken zumeist erst spät erkannt. Änderungen in den frühen Phasen sind kostspielig und schwer zu managen.

Ende der 80er Jahre stellte Barry Boehm das Spiralmodell vor [Boehm, 1986]. In diesem Modell werden die Erkenntnisse umgesetzt, dass es einen natürlichen, nicht total vermeidbaren Wandel bei der Planung und den Anforderungen gibt. Das Spiralmodell ist zwar immer noch rein sequenziell, es beinhaltet jedoch eine iterative Planung mit integrierter Risikoanalyse. Nach [Pomberger und Blaschek, 1996] eignet sich dieses Modell gut für die Einbettung von Maßnahmen zur Qualitätssicherung in den Entwicklungsprozess. Fehler können früh erkannt und verschiedene Lösungsmöglichkeiten können mit einem vertretbaren Aufwand betrachtet werden.

Neben dem Wasserfallmodell und dem Spiralmodell sind weiteren strukturierten Vorgehensmodelle entstanden. Sie sind als erste Generation von Methoden zur Softwareentwicklung von großer Bedeutung (siehe [Balzert, 1996]). Im Vordergrund stehen dabei die Modellierung der Funktionen und die Entwicklung eines globalen Datenmodells. In einem eigenen Arbeitsschritt werden Funktions- und Datenmodell aufeinander abgestimmt. Bei einem solchen Top-Down-Verfahren, d.h., bei Entwicklungen von abstrakten Modellen hin zu programmiersprachlichen Realisierungen, ist der Übergang zwischen Modellebenen oft mit Modellbrüchen verbunden (vgl. [Rechenberg et al., 2002]). Die Top-Down-Vorgehensweise und die Modellbrüche haben in strukturierten Methoden immer wieder zu Problemen geführt, sodass die Entwicklung neuerer Ansätze vorangetrieben wurde.

Objektorientierte Vorgehensmodelle

Über die Jahre sind die zu entwickelnden Softwaresysteme immer größer geworden, und die damit verbundene Komplexität der Entwicklung ist stets angestiegen. Die Grenze der Top-Down-Vorgehensweise der strukturierten Vorgehensweisen wurde erreicht. Die Modellbrüche in der funktionsorientierten Softwareentwicklung und die Durchdringung der objektorientierten Programmierung führten dazu, dass erste Methoden zur objektorientierten Softwarekonstruktion, wie zum Beispiel [Horn und Schubert, 1993], entstanden. Dabei stehen die Gegenstände der Anwendungsdomäne im Vordergrund. Auf den Gegenständen können Operationen durchgeführt werden, die auf den gekapselten Daten arbeiten. Diese Operationen werden nicht zu standardisierten Abläufen zusammengesetzt, sondern als Dienstleistungen zur Benutzung angeboten.

Ein wichtiges Strukturierungsmerkmal für die Entwürfe und die Softwarearchitektur ist das Konzept der Hierarchiebildung, das programmiertechnisch durch Vererbung umgesetzt wird (vgl.[Rechenberg et al., 2002]). Die Objektorientierung bietet die Möglichkeit, Beschränkungen strukturierter Methoden zu überwinden, d.h. Daten zusammen mit Operationen zu modellieren, existierende Systeme flexibel zu erweitern und stückweise zu integrieren. Für alle Ebenen wird dabei eine einheitliche Modellbasis verwendet, Produkte können flexibel strukturiert und in die Umgebung integriert werden. Die objektorientierte Systemsicht unterstützt insbesondere die Erweiterbarkeit und die Wiederverwendbarkeit von Systemen (siehe [Breu, 2001]). Wichtige Grundideen objektorientierten Vorgehens sind die Konzepte der Datenkapselung, der Vererbung und der dynamischen Existenz von Objekten. Objekte sind die atomaren Strukturierungseinheiten eines objektorientierten Systems. Jedes Objekt hat einen internen Zustand, der durch die Ausprägung seiner Attribute bestimmt ist. Dieser interne Zustand ist gekapselt, d.h. er ist von außen nicht direkt zugänglich, sondern nur über die

Ausführung von Operationen. Sie bilden die Schnittstelle des Objekts. Objekte kommunizieren untereinander durch das Senden von Nachrichten. Weiterhin sind objektorientierte Systeme sehr dynamisch. Objekte können zur Laufzeit erzeugt werden und nach Abschluss der Interaktionen hören sie nach einer Weile auf, zu existieren. Objekte werden zumeist zu Klassen gruppiert. Klassen verkörpern das Typkonzept und beschreiben Mengen von Operationen mit ähnlicher interner Struktur, Schnittstelle und Verhalten. Eine weitere Gruppierung ähnlicher Objekte wird mit dem Konzept der Vererbung unterstützt.

In den 90er Jahren wurden eine Vielzahl von objektorientierten Softwareentwicklungsmethoden entwickelt. Bekannteste Ansätze sind die Methoden der OO-Analyse und OO-Design [Coad und Yourdon, 1991a] und [Coad und Yourdon, 1991b], OMT [Rumbaugh et al., 1991], die Methoden von Booch [Booch, 1993] und Jacobson [Jacobson, 2004] sowie die "Fusion Method" [Coleman et al., 1994]. Eine Vereinheitlichung und Standardisierung der dabei verwendeten unterschiedlichen Notationen wurde mit der Entwicklung der Unified Modeling Language (UML) [Booch et al., 1999] unternommen. Methodische Rahmenwerke, die auf Basis der UML definiert wurden, sind beispielsweise der Unified Software Development Process [Jacobson et al., 1999], der Catalysis-Ansatz [D'Souza und Wills, 1999] oder der Rational Unified Process [Kruchten, 2003].

2.3 Requirements Engineering – RE

Der Begriff "Requirement" (oder auch Anforderung) bezeichnet in Ingenieurwissenschaften ein dokumentiertes Bedürfnis nach etwas, was ein bestimmtes Produkt oder ein bestimmter Dienst sein, haben, aufweisen oder tun soll. In klassischen Engineering-Ansätzen dienen Mengen von Anforderungen als Eingabe zur Design-Phase der Produktentwicklung. Die Entwicklung von Anforderungen sollte erst nach einer Phase der Machbarkeitsanalyse oder der konzeptuellen Analyse stattfinden. Im Bereich des Software und Systems Engineering kann eine Anforderung eine Beschreibung dessen, was ein (Software-)System tun soll, bezeichnen. Diese Art von Anforderung spezifiziert etwas, was das System in der Lage sein soll, tun zu können. Sie werden funktionale Anforderungen genannt. Eine andere Art von Anforderung spezifiziert Eigenschaften, die besagen, wie qualitativ das System seine funktionalen Anforderungen genügen soll. Diese Art von Anforderungen werden meist "nicht funktionale", oder "Qualitätsanforderungen" genannt. Beispiele hierfür sind Anforderungen, die die Verfügbarkeit, Wartbarkeit oder die Usability[1] des Systems betreffen. Eine Sammlung von Anforderungen definiert die Eigenschaften oder Features des gewünschten Systems.

Bei der Entwicklung von Anforderungen werden gewisse Ziele verfolgt, die mit der Durchführung bestimmter Aktivitäten erreicht werden. Diese Ziele und die entsprechenden Aktivitäten werden in diesem Abschnitt beschrieben. Zuvor werden die Hintergründe des Requirements Engineering zusammen mit Kernbegriffen in seinem Umfeld erläutert. Am Ende des Abschnitts erfolgt eine Beschreibung unterschiedlicher Ansätze des Requirements Engineering.

[1] Usability wird des Öfteren mit dem Deutschen Begriff Gebrauchtauglichkeit übersetzt. Allerdings existieren über diesen Begriff unterschiedliche Meinungen. In dieser Arbeit wird diese Diskussion dadurch vermieden, indem der Englische Begriff Usability nicht übersetzt wird.

2.3.1 Hintergründe und Begriffe

Gegenstand des Requirements Engineering ist es, Anforderungen an ein zu entwickelndes System zu definieren. Somit beschäftigt sich Requirements Engineering mit der Entwicklung von Anforderungen. IEEE definiert den Begriff "Requirements Engineering" als Prozess des Studierens von Nutzerwünschen, um eine Definition der System-, Hardware- und Softwareanforderungen zu erhalten und als Prozess des Studierens und Verfeinerns von System-, Hardware- und Softwareanforderungen [IEEE, 1990]. Requirements Engineering wird somit nicht nur als Identifizierung von Anforderung zu Beginn einer Softwareentwicklung verstanden, sondern auch als ein iterativer, inkrementeller Prozess, der das Gewinnen, Dokumentieren, Konsolidieren und Verfolgung von Kunden- und/oder Nutzeranforderungen über den gesamten Entwicklungsprozess eines Systems hinweg umfasst.

Eine Anforderung ist nicht nur eine Bedingung oder Eigenschaft, die ein Nutzer benötigt, um ein Problem zu lösen oder ein Ziel zu erreichen, sondern kann auch eine Bedingung oder Eigenschaft sein, die ein System oder eine Systemkomponente aufweisen muss, um einen Vertrag zu erfüllen oder einem Standard, einer Spezifikation oder einem anderen formell auferlegten Dokument zu genügen. Vielfach wird auch die dokumentierte Form einer Bedingung oder Eigenschaft als Anforderung bezeichnet (vgl. [IEEE, 1990]). Um eine klare Unterscheidung zwischen den Bedingungen und den dokumentierten Bedingungen zu haben, wird eine Bedingung oder Eigenschaft, die ein Kunde fordert bzw. über die ein System verfügen soll als Anforderung bezeichnet. Die dokumentierte Bedingung oder Eigenschaft hingegen wird als Anforderungsartefakt bezeichnet.

Requirements Engineering ist eine Kernaktivität bei der Software- und Systementwicklung, die ursprünglich aus der Systemanalyse entstanden ist. Die Systemanalyse basiert auf der Analyse von bestehenden Systemen oder Prozessen, um diese durch das System, das entwickelt wird, (teilweise) zu unterstützen oder zu ersetzen. Bekannte Methoden der Systemanalyse sind die *Strukturierte Analyse* [DeMarco, 1978], die *Essential Systems Analysis* [McMenamin und Palmer, 1984] und die *Modern Structured Analysis* [Yourdon, 1989].

Das Requirements Engineering beschränkt sich aber nicht nur auf Entwicklung von Systemen zur Ersetzung (alter) Systeme oder Prozesse, sondern umfasst auch die Entwicklung vollkommen neuer Systeme. Für die Entwicklung solcher Systeme reicht die traditionelle Systemanalyse schlichtweg nicht aus, da es keine Systeme oder Prozesse gibt, die zuvor analysiert werden können. Das Wissen über die Anforderungen muss von den Stakeholdern, beispielsweise durch Interviews, gewonnen werden (vgl. [Gause und Weinberg, 1989], [Kotonya und Sommerville, 1998] und [Robertson und Robertson, 1999]).

Das moderne Requirements Engineering umfasst Methoden, die die Ingenieure dabei unterstützen, auch Anforderungen an neue Systeme zu gewinnen, zu konsolidieren und zu dokumentieren. Heutzutage gibt es verschiedene RE-Ansätze. Aktuelle RE-Ansätze sind die zielbasierten Ansätze (u.a. [Antón, 1996], [Darimont et al., 1997], [Mylopoulos et al., 1999] und [van Lamsweerde, 2001]), die szenariobasierten Ansätze (u.a. [Rolland et al., 1998] und [Kaindl, 2000]) sowie die modellbasierten Ansätze (u.a. [Heumesser und Houdek, 2003], [Geisberger et al., 2006] und [Broy et al., 2007]). Eine Beschreibung dieser Ansätze erfolgt weiter unten in diesem Kapitel.

2.3.2 Ziele und Aktivitäten

Drei wesentliche Ziele werden im Requirements Engineering verfolgt:

1. Probleme, Bedürfnisse und Wünsche der Kunden, Nutzer und weiteren relevanten Stakeholder gut genug verstehen, so dass sie mit höchster Wahrscheinlichkeit befriedigt werden.
2. Eine angemessene Teilmenge der Probleme, Bedürfnisse und Wünsche auswählen, so dass sie mit höchster Wahrscheinlichkeit im Rahmen der gegebenen Zeit und Budget gelöst, befriedigt bzw. erfüllt werden.
3. Externes Verhalten des gewünschten System gut genug dokumentieren, so dass das entwickelte System dem entspricht, was die Stakeholder (insbesondere das Marketing und der Kunde) erwarten.

Zur Erreichung dieser Ziele werden im Rahmen des Requirements Engineering die Aktivitäten der *Gewinnung*, *Analyse* und *Spezifikation* der Anforderungen durchgeführt.

Die Aktivität der Anforderungsgewinnung (engl. Requirements Elicitation) hat das Ziel, Anforderungen von den Stakeholders zu identifizieren, gewinnen bzw. zu erheben.

Die Aktivität der Anforderungsanalyse (engl. Requirements Analysis) hat das Ziel, die erhobenen Anforderungen zu analysieren und sie auf Konsistenz und Vollständigkeit zu prüfen. Bei der Anforderungsanalyse wird stets im Auge behalten, welche Anforderungen mit den gegebenen Ressourcen innerhalb der angegebenen Zeit erfüllt werden können. Ausgehend von bereits identifizierten Anforderungen können zusätzliche Anforderungen identifiziert werden. Während der Anforderungsanalyse entstehen drei Kategorien von Anforderungen, die kontinuierlich überarbeitet werden. Die erste Kategorie enthält alle Anforderungen, die das System erfüllen muss; die zweite Kategorie enthält alle Anforderungen, die das System nicht erfüllen braucht; und die dritte Kategorie enthält alle Anforderungen, die das System erfüllen kann, allerdings unter sorgfältiger Berücksichtigung der verfügbaren Ressourcen. Beim Eingang neuer Anforderungen werden die einzelnen Kategorien unter Berücksichtigung der verfügbaren Ressourcen und der angegebenen zeitlichen Einschränkungen stets überprüft und neu sortiert. Dabei können Anforderungen Kategorien wechseln. Der Prozess der Festlegung, welche Anforderungen ein System erfüllt soll, unter Berücksichtigung gegebener Zeit- und Ressourcen-Rahmen besteht grundsätzlich aus drei Teilaktivitäten: einer relativen Priorisierung der gesammelten Anforderungen, einer Schätzung der Ressourcen, die für die Erfüllung einer Anforderung benötigt werden, und einer Auswahl einer geeigneten Untermenge der gesammelte Anforderungen, die das System erfüllt wird.

Die Aktivität der Anforderungsspezifikation (engl. Requirements Specification) erfolgt mit dem Ziel, die Anforderungen für die Entwickler aufzuschreiben, und somit eine initiale Brücke zwischen Anforderungen und Entwurf zu schaffen. Während der Anforderungsspezifikation werden also die identifizierten und analysierten Anforderung dokumentiert.

Neben diesen Aktivitäten spielen auch die Aktivitäten der *Validierung* und *Verifikation* der dokumentierten Anforderungen gegen die Stakeholder-Bedürfnisse eine wichtige Rolle im Requirements Engineering. Bei der Validierung geht es darum, die Überein-

stimmung der Ergebnisse mit den Zielen und Bedürfnissen der verschiedenen Stakeholder festzustellen. Dabei werden Fragen der Art "Bauen wir das richtige System?" adressiert. Die Aktivität der Verifikation zielt darauf, die Angemessenheit (im Sinne der Korrektheit) der spezifizierten Anforderungen nachzuweisen. Dabei werden Fragen der Art "Bauen wir das System richtig?" adressiert.

2.3.3 Aktuelle Ansätze

Neuere Ansätze im RE verwenden Ziele um die Intentionen und Bedürfnisse der Stakeholder zu erfassen und diese durch exemplarische Beschreibungen (Szenarien - Use Cases) anzureichern. Neben den ziel- und szenariobasierten Ansätzen werden Anforderungen vermehrt modellbasiert dokumentiert, um einerseits Anforderungen besser kommunizieren zu können und andererseits die Komplexität der Anforderungen beherrschbarer zu machen. Ziel- und szenariobasiertes Requirements Engineering sowie modellbasierte Ansätze sind dabei keinesfalls konkurrierende Techniken, und sollten auch nicht unter diesem Blickwinkel betrachtet werden. Die natürlich sprachliche Dokumentation von Zielen und Szenarien kann mit der Dokumentation in Form von Modellen ergänzt werden.

Zielorientiertes Requirements Engineering

Ein Paradigma der aktuellen Requirements Engineering Ansätze ist der zielorientierte Ansatz. Dabei werden, ausgehend von den Absichten der Stakeholder, warum das System benötigt bzw. entwickelt wird, Anforderungen an das System abgeleitet, also was das System können bzw. leisten soll [Yu, 1997, van Lamsweerde, 2001].

Durch eine explizite und intensive Betrachtung der Ziele eines Systems können auch irrelevante Anforderungen identifiziert werden, d.h. Anforderungen, die nicht dazu beitragen, irgendwelche Stakeholder Ziele zu erreichen. Außerdem können mit Hilfe von Zielen die erfassten Anforderungen auf Vollständigkeit hin überprüft werden [Yue, 1987], indem hinterfragt wird, ob sämtliche Ziele des Systems durch die definierten Anforderungen erfüllt werden können. Ziele dokumentieren die Wünsche von Stakeholdern und können von diesen leichter benannt und nachvollzogen werden. Die Verwendung zielorientierter Ansätze insbesondere zur Anforderungsgewinnung und im Rahmen der Identifikation von Konflikten zwischen unterschiedlichen Absichten der Stakeholder wird als sehr vorteilhaft eingestuft [van Lamsweerde, 2001, Kavakli, 2002, Bleistein et al., 2006].

Bei zielorientierten RE Ansätzen liegt der Fokus besonders auf folgenden Fragestellungen:

- Identifikation von High-Level Zielen des zu entwickelnden Systems
- Verfeinerung und Operationalisierung der Ziele bis auf die Ebene der Anforderungen
- Identifikation und Resolution von Ziel- und Anforderungskonflikten
- Repräsentation von Zielsystemen als Kriterium für die Prüfung der Anforderungen auf ihre Vollständigkeit

In der Literatur wird eine Reihe zielorientierter Ansätze angegeben. Darunter zählen der NFR (Nonfunctional Requirements) Ansatz ([Chung et al., 1999],[Mylopoulos et al., 1999]), konzipiert für die Gewinnung und Dokumentation von Qualitätzielen bzw. -Eigenschaften eines Systems, der GBRAM (Goal-Based Requirements Analysis Method) Ansatz ([Antón, 1996, Antón, 1997, Antón et al., 2001]), die eine Methode zur hierarchischen Analyse und Verfeinerung von Zielen definiert, und auch der AGORA (Attributed Goal-Oriented Requirements Analysis) Ansatz [Kaiya et al., 2002]. Eine weitere zielorientierte Methode ist der KAOS (Knowledge Acquisition in autOmated Specification) Ansatz mit dem Schwerpunkt auf der Verfeinerung und Operationalisierung von abstrakten Zielen ([Dardenne et al., 1993, Darimont und van Lamsweerde, 1996, Darimont et al., 1997, Letier und van Lamsweerde, 2002]) sowie auf der Konfliktlösung ([van Lamsweerde et al., 1998]).

Die hohe Anspruchshaltung zielorientierter Ansätze, beispielsweise in Bezug auf Ziel-Gewinnung, stellt sich in der praktischen Umsetzung, trotz der breiten Literatur, immer noch als eine sehr komplexe Aufgabe dar. Den an der Systementwicklung beteiligten Praktikern ist in der Regel ein abstrakter Zielbegriff, in Bezug auf einzelne Bereiche ihrer Domäne, nicht ohne Weiteres zu vermitteln. Die Anwendung einzelner Methoden zur Identifikation von Zielen ist nicht immer selbstverständlich. Die in der Literatur immer wieder als notwendig gesehenen Eliminierung irrelevanter oder falscher Ziele stellt sich allerdings in der Praxis als sehr schwierig heraus. Eine partielle Beseitigung der benannten Defizite bei zielorientierten Ansätzen wird angestrebt durch eine Kopplung dieser Ansätze mit szenariobasierten Ansätzen.

Szenariobasiertes Requirements Engineering

Szenariobasiertes Requirements Engineering umfasst sowohl die Entwicklung von Use Cases als auch die Entwicklung modellgetriebener Szenarien, die z.B. in Form von Message Sequence Charts dokumentiert werden. Ein Überblick über szenariobasierte Ansätze (sowohl Use Cases orientiert und modellgetrieben) wird in [Amyot und Eberlein, 2003] gegeben.

Use Cases sind eine weit verbreitete Technik, die für die Gewinnung und Dokumentation von Anforderungen eingesetzt werden kann. Mittels Use Cases wird die Nutzung eines geplanten Systems aus Sicht des Nutzers beschrieben. Use-Case-basierte Ansätze werden beispielsweise in [Jacobson et al., 1992, Kulak und Guiney, 2000, Armour und Miller, 2001, Cockburn, 2001] sowie in [Bittner und Spence, 2002] ausführlich aufgeführt.

Als modellbasierte Szenarien werden szenariobasierte Ansätze bezeichnet, denen eine (teilweise) formale Semantik zugrunde liegen. Derartige Ansätze sind beispielsweise in [Dano et al., 1997] und in [Sutcliffe et al., 1998] beschrieben. Das Grundkonzept dieser Ansätze liegt in der Beschreibung von Interaktionen zwischen einer Menge von Entitäten. Je nach Domäne und Verwendungszweck werden die interagierenden Entitäten als Komponenten, Akteure, Prozesse oder Agenten bezeichnet; diese Begriffe werden jedoch teilweise synonym verwendet. Kennzeichnend für eine Entität ist, dass sie über ein Verhalten verfügt, d.h. sie kann durch den Austausch von Nachrichten mit ihrer Umgebung interagieren. Szenarien beinhalten also eine Menge von Entitäten

sowie eine Abfolge von Nachrichten.

Durch konkrete Interaktionsfolgen zwischen einem Benutzer und einem System dokumentieren Szenarien den Nutzen des Systems. Szenarien können daher Defizite oder positive Aspekte eines bestehenden oder zu entwickelnden Systems anhand konkreter Beispiele aufzeigen. Anhand der Defizite oder Vorteile aus den Szenarien lassen sich dann Ziele formulieren, die das zu erstellende System erfüllen soll. Ziele und Szenarien ergänzen sich gegenseitig. Während ein Szenario konkret ist, aber Intentionen oft nicht explizit beschreibt, lassen sich Ziele abstrakt formulieren und drücken Intentionen explizit aus [van Lamsweerde, 2001]. Zielorientierte Ansätze ermöglichen alleine nicht das zufrieden stellende Erheben von Anforderungen, und sollten durch Szenarien ergänzt werden. Demnach werden Szenarien verwendet, um die Entwicklung von Zielmodellen zu unterstützen (siehe z.B. [Potts, 1995, Antón und Potts, 1998, Rolland et al., 1998, Rolland et al., 1999, Kaindl, 2000] oder auch [Letier, 2001]). In diesem Zusammenhang wurden Ansätze vorgeschlagen, um Ziele aus Szenarien abzuleiten und vice versa (siehe [van Lamsweerde et al., 1998, Antón et al., 2001].

Modellbasiertes Requirements Engineering

Die Grundidee hinter modellbasierten Requirements Engineering Ansätzen beruht zum einen auf der Verwendung eines Artefaktmodells als konzeptuelles Modell aller im RE zu erstellenden bzw. zu berücksichtigenden Artefakte und der Beziehungen der Artefakte untereinander. Dabei bezeichnet ein Artefakt einen Inhalt, der im Rahmen des Requirements Engineering Prozesses entsteht oder dazu verwendet wird eine Spezifikation zu erarbeiten. Das Artefaktmodell definiert also die im RE zu entwickelnden Inhalte. Mit der Durchführung eines RE-Prozesses wird ein solches Artefaktmodell instanziiert, d.h. für die im Artefaktmodell definierten Artefakttypen werden konkrete Ausprägungen erzeugt. *Volere Templates* [Kotonya und Sommerville, 1998] stellen in gewisser Hinsicht ein Beispiel dieser Klasse modellbasierter RE Ansätze dar.

Zum anderen ist der Einsatz von Systemmodellen und Sichten, insbesondere Architekturmodellen und funktionalen Sichten, Teil des modellbasierten Ansatzes. Es wird auf eine Einbeziehung zugeschnittener Systemmodelle in das Requirements Engineering gezielt. Systemsichten werden für die Modellierung der erforderlichen Systemdienste, ihre Struktur und ihre Nutzung mittels unterschiedlicher Sichten definiert. Zu den Systemsichten gehören die Ablaufsicht für die Prozess- und Szenariomodelle, die Struktursicht für das Systemsumgebungsmodell, die logischen Systemgrenzen, die Diensthierarchie und die Komponentenstruktur, die Interaktionssicht für das Interaktionsmodell, die Verhaltenssicht für die Zustandsübergangsmodelle und die Datensicht für die Datenstrukturen zur Struktur-, Interaktions- und Verhaltensmodellierung.

Die Modellbasierung wird zwar als ein viel versprechendes Mittel zur Komplexitätsbeherrschung bei der Entwicklung von Software-Systemen gesehen ([Broy, 2006b]), ist aber bislang auf der Anforderungsebene nicht genügend ausgeprägt. Erste Grundlagen zum modellbasierten RE wurden in [Geisberger, 2005], [Geisberger und Schätz, 2007] gelegt. Der *REM-Ansatz (Requirements Engineering Reference Model)* z.B. (siehe [Geisberger et al., 2006], [Broy et al., 2007]) stellt ein Framework zum modellbasierten RE dar. Auch der *RAM-Ansatz (Requirements Abstraction Model)* [Gorschek und Wohlin, 2006] ist ein Vertreter modellbasierter RE Ansätze.

2.4 Context-Aware Computing

"Context Awareness is the ability of an applications to discover and react to changes in the context in which they are situated." – William N. Schilit [Schilit, 1995]

Konzepte für Softwaresysteme, die ihren Kontext berücksichtigen, werden seit Ende des 20. Jahrhunderts immer wichtiger für das Gebiet des Software Engineering und insbesondere im Bereich Context-Aware und Ubiquitous Computing. Dabei wird versucht, eine Anwendung abhängig von ihrem Kontext anzupassen, und die Interaktion zwischen dem Nutzer und dem System auf das minimal notwendige zu reduzieren.

Die Grundidee hinter diesen Konzepten ist ein flexibles Systemverständnis, wobei der Gedanke des Systems als Werkzeug nicht mehr allein im Vordergrund steht, sondern und vor allem auch die Erfüllung der Bedürfnisse und Wünsche des Nutzers in möglichst vielen Nutzungssituationen. Welche Informationen in einer gegebenen Nutzungssituation für die Erfüllung der Bedürfnisse und Wünsche des Nutzers von Bedeutung sind, hängt von vielen Faktoren ab. Die Feststellung dieser Faktoren stellt eine Herausforderung für das Forschungsgebiet des Context-Aware Computing, das sich als Ziel gesetzt hat, die Nutzung eines Systems, die Interaktion zwischen dem System und seinem Nutzer durch Bereitstellung von über einer Menge von Sensoren gesammelten relevanten Informationen zu erleichtern.

In diesem Abschnitt werden grundlegende Konzepte in Zusammenhang mit Context-Aware Computing aufgeführt. Erste Arbeiten auf dem Gebiet des Context Awareness [Want et al., 1992, Schilit et al., 1993, Schilit et al., 1994] sind inspiriert von Mark Weiser's Ubiquitous Computing Vision. So werden zuerst die Grundideen hinter Ubiquitous Computing zusammen mit den damit verbundenen Voraussetzungen zusammengefasst. Danach wird der Zusammenhang zwischen den Konzepten Context Awareness, Kontextsensitivität und Kontextadaptivität erläutert. Das für diese Arbeit besonders wichtige Konzept der Kontextadaption wird ausführlich diskutiert. Anschließend wird ausgehend von der Kontextadaption eine Architektur kontextadaptiver Anwendungen vorgestellt, wie sie in dieser Arbeit Anwendung findet.

2.4.1 Die Vision von Mark Weiser

"Ubiquitous Computing enhances computer use by making computers available throughout the physical environment, while making them effectively invisible to the user" – Mark Weiser (23.07.1952 - 27.04.1999)

1988 führte Mark Weiser der Begriff des Ubiquitous Computing ein, um eine Zukunft zu beschreiben, in der der *Personal Computer*, auch PC genannt, durch eine Menge "unsichtbarer" Rechner ersetzt wird, die in alltägliche Gebrauchsgegenstände integriert sind [XEROX, 1999]. Sein Artikel, der 1991 in *The Scientific American* unter dem Titel *The Computer of the Twenty-First Century* erschien, gilt als Initialpunkt der Forschung zu Ubiquitous Computing [Weiser, 1991]. In diesem Artikel beschreibt Weiser die Vision einer Welt, in der Rechner nicht mehr eigenständige Geräte sind, mit denen sich der Mensch explizit auseinander setzen muss. Während man sich heutzutage meist noch an den Schreibtisch setzen muss, um auf den Monitor des PC zu blicken, wenn man Ausgaben wie E-Mails oder Web-Seiten betrachten möchte, könnte dies in der von Weiser beschriebenen Zukunft stattdessen auf dem Fernseher, auf dem Display

des Auto-Bordcomputers oder durch Projektion auf beliebige geeignete Flächen in der Einsatzumgebung möglich sein.

Mit Ubiquitous Computing entsteht also ein Verhältniswechsel zwischen Benutzern und der Anzahl verwendeter Computersysteme (siehe Abbildung 2.2). Nun steht der Benutzer im Mittelpunkt der Interaktionen; er ist umringen von unterschiedlichen Rechnern, mit denen er teils direkt (und bewusst) interagieren kann, und teils indirekt (und ohne explizite Eingabe) interagieren kann.

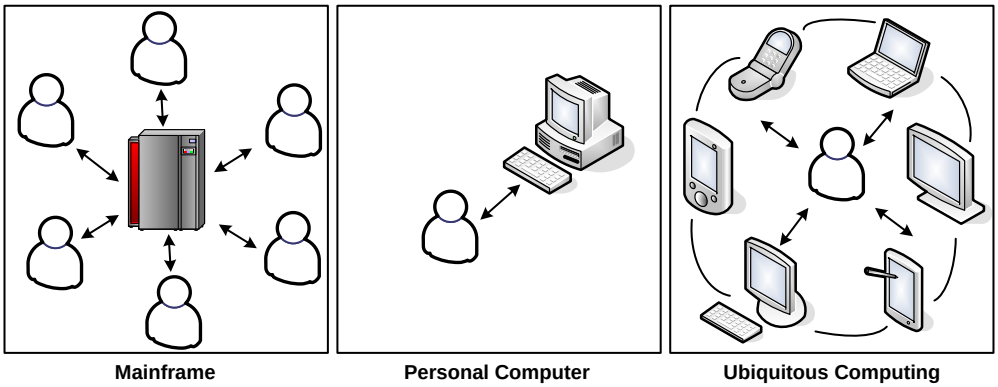

Abbildung 2.2: Ubiquitous Computing mit dem Benutzer als Mittelpunkt

Definition 2.1 Ubiquitous Computing

☞ Unter Ubiquitous Computing ist die direkte oder indirekte Nutzung computer-basierter Anwendungssysteme in möglichst vielen Situationen (z.B. Orte, Zeiten, Tätigkeiten) eines Benutzers zu verstehen. (Angelehnt an [Kephart und Chess, 2003, Fahrmair, 2005])

Ubiquität in Zusammenhang mit der Definition 2.1 bedeutet die Nutzbarkeit eines Systems im weitesten Sinne auf so viele Situationen wie möglich auszudehnen. Dazu gehören explizit auch Alltagssituationen eines Benutzers in denen seine Interaktionsfähigkeit beschränkt ist (etwa beim Autofahren) oder Situationen mit heterogener technischer Infrastruktur unsicherer Verfügbarkeit (etwa in heterogenen Funknetzen).

Die Verwirklichung einer solchen Vision des Ubiquitous Computing ist mit einer Menge von Voraussetzungen verknüpft, die zuvor noch geschaffen werden müssen. Im Folgenden geben wir dazu ein Überblick, ohne dabei einen Anspruch auf Vollständigkeit zu stellen.

1. *Hardware Infrastruktur*
   Notwendige Hardware Infrastruktur inklusive Sensoren und Aktuatoren verfügbar machen, Energiebedarf der Geräte optimieren, Wartungsaufwand in Grenze halten.
2. *Kommunikation*
   Kooperation und Informationsaustausch zwischen den Ressourcen im Sinne ihrer Interoperabilität fördern, nahtloser Übergang zwischen den Übertragungstechnologien im Falle von Mobilität ermöglichen (*Seamless Handover* [Dillinger et al., 2004, Mohyeldin et al., 2004b, Mohyeldin et al., 2004a]).

3. *Mensch-Maschine-Interaktion*
   Primäres Ziel der Anwendungen ist eine Unterstützung des Benutzers durch eine kooperative und automatische Reaktion auf den jeweiligen Kontext. Umso wichtiger wird dies, da Geräte, Dienste und ihr Zusammenspiel stetig komplexer werden. Intuitive und durchschaubare Schnittstellen zwischen dem Nutzer und dem System sind also erforderlich [Schmidt, 2002].

4. *Privacy und Security*
   Schutz von Privatsphäre und Datenschutzinteressen um Nutzerakzeptanz zu gewährleisten, Schutz vor unbefugtem Zugriff [Rieback et al., 2005, Fahrmair et al., 2005, Fahrmair et al., 2007]. Wirksame und effiziente Gesamtstrategien erforderlich.

5. *Standards*
   Standards zur Beschreibung von Informationen, Diensten und Funktionalität, Standards zum Propagieren, zur Installation, zur Suche und zur Inanspruchnahme von Diensten, aber auch Standards für das Management von Diensten sind erforderlich. Oft stehen wirtschaftliche Interessen und Konkurrenzdruck einer Standardbildung zur Ermöglichung von Interoperabilität entgegen.

Die aufgeführten Voraussetzungen werden bereits heute intensiv erforscht, so dass die Verwirklichung der Vision einer allgegenwärtigen Computernutzung, eventuell in abgeschwächter Form, in greifbare Nähe rückt. Allerdings bleibt über diese einzelnen Voraussetzungen hinaus die Frage der Entwicklungswerkzeuge und Methoden zu klären. Diese sind, wenn überhaupt, noch in einem sehr frühen Stadium [Davies et al., 2005], obwohl sie eine profitable Umsetzung der Ubiquitätsgedanke ermöglichen würden.

Neben Ubiquitous Computing, etwa der allgegenwärtige Computer, haben sich unter anderem auch die Begriffe Pervasive Computing, etwa der alles durchdringende Computer, Ambient Intelligence und der Disappearing Computer im Sprachgebrauch eingebürgert. Hinter all diesen Begriffen steckt jedoch eine Gemeinsamkeit: Sie basieren allesamt auf einem wesentlich flexibleren Systemverständnis als bisher. Dabei werden die Berücksichtigung der Bedürfnisse des Benutzers und die Erfüllung seiner Wünsche betont in Vordergrund gestellt. Ferner implizieren diese Begriffe mittelbar die Nutzung situationsbezogener Informationen, um die Anwendungen so zu optimieren, dass sie in den jeweils aktuellen Nutzungssituationen angebracht sind. Für eine explizite Hervorhebung dieser Implikation führte Schilit in seine Arbeiten zu seiner Zeit in dem von Marc Weiser geführten Computer Science Lab in Xerox PARC den *context-aware* Begriff ein [Schilit et al., 1993, Schilit et al., 1994, Schilit und Theimer, 1994] und [Schilit, 1995]. Context Awareness (Kontextsensitivität) in Zusammenhang mit einer Anwendung meint die Fähigkeit dieser Anwendung Änderungen in der Umgebung über Sensorik feststellen zu können und darauf reagieren zu können. Die Fragestellungen, welche Änderungen in der Umgebung festzustellen sind, und was mit „darauf reagieren“ gemeint ist, sind bis dato nicht ausführlich adressiert worden.

2.4.2 Kontext: Awareness, Sensitivität und Adaptivität

Die immer intensivere Kontextbezogenheit der Anwendungen führte zur Einführung des Context-Awareness [Schilit et al., 1993]. Etymologisch bezeichnet Context Awaren-

ess das Bewusstsein oder die Erkenntnis des Sinnzusammenhangs und ist somit näher an Kontextsensitivität als das, was eigentlich damit gemeint ist, nämlich die Kontextadaption. Die Sensitivität in Zusammenhang mit den betrachteten Anwendungen geht nicht ohne die Anpassung der Anwendung. Eine solche Anpassung ist explizit in dem Begriff Adaption ausgedrückt, und mit Kontextadaption wird der expliziten Anpassung an den Kontext Ausdruck verliehen.

Seit der Einführung des Context Awareness Begriffs ist eine Reihe von Beiträgen auf diesem Gebiet geleistet worden. Allerdings waren diese Beiträge in erster Linie von technischer Natur und legen den Fokus auf die grundlegende Architektur oder die technische Realisierung der Anwendungen unabhängig von welcher Perspektive aus sie betrachtet wird. Zu diesen Beiträgen zählen die Context Aware Systemarchitektur [Schilit, 1995], das Context-Toolkit [Dey et al., 2001] als Entwicklungsumgebung für Context-Aware Anwendungen, das Computing in Context [Schmidt, 2002] mit dem Fokus auf die Sensor-orientierte Kontexterfassung, das Konzept der kontextadaptiven Dienstnutzung [Samulowitz, 2002], das CAWAR Architektur-Framework [Fahrmair, 2005] zur Entwicklung ubiquitärer Anwendung und die Konzepte der Kontextbereitstellung [Krause, 2006] um hier nur einige Beispiele zu nennen. Obwohl Context Awareness ursprünglich als wichtige technologische Voraussetzung zur Verwirklichung der Ubiquitous Computing Vision von Mark Weiser eingeführt wurde, kommt dieses Konzept nicht nur in Zusammenhang mit Ubiquitous Computing zum Einsatz. Das Licht im Korridor wird automatisch eingeschaltet, sobald man diesen Bereich betritt; Das Licht eines Fahrzeugs wird automatisch eingeschaltet, sobald es draußen dunkler wird; die Lautstärke des Radios im Auto wird automatisch erhöht, sobald der Motor lauter wird. Hierzu werden benötigte Sensoren und Aktuatoren miteinander verknüpft. Es werden also Informationen mittels Sensoren erfasst, und daraufhin automatische Anpassungen durch Aktuatoren vorgenommen. In deutschem Sprachgebrauch werden die Begriffe Kontextadaptivität und Kontextsensitivität als Synonym zum englischen Begriff des Context Awareness verwendet [Samulowitz, 2002, Fahrmair, 2005, Krause, 2006].

Obwohl es offensichtlich ist, dass Kontextsensitivität und Kontextadaptivität nicht zwangsläufig gleichbedeutend sind, werden diese Begriffe verwendet, um das selbe zu bezeichnen: Es werden abhängig von Umgebungsgegebenheiten (also Kontext) Anpassungen an einer Anwendung vorgenommen (also Adaption). In dieser Arbeit konzentrieren wir uns genau auf diese Art von Anwendungen. Wir stellen den Kontext der Nutzung einer solchen Anwendung und die Anpassung der Anwendung bezüglich dieses Kontextes in den Mittelpunkt. Demnach werden im Abschnitt 2.4.3 die Grundlagen zur Kontextadaption aufgeführt. Dabei werden eine Reihe grundlegender Merkmale der Kontextadaption diskutiert. Aus konzeptueller Sicht lassen sich mittels Kontextadaption neuartige Systeme einführen, die wir kontextadaptiv oder kontextsensitiv nennen.

Definition 2.2 Kontextadaptives (auch kontextsensitives) System

☞ Ein kontextadaptives System ist ein Computer-basiertes System mit der Fähigkeit:

1. *Änderungen in der Domäne*, in der sie genutzt wird, zu erkennen, und
2. sein *beobachtbares Verhalten*, selbst ohne direkte *Nutzerinteraktion*, so zu ändern, dass dieses der veränderten Domäne *angepasst* ist.

Die in der Definition 2.2 geforderten Fähigkeiten eines kontextadaptives Systems werden ebenfalls in folgendem Abschnitt genauer erläutert. Anschließend wird eine allgemeine Architektur kontextadaptiver Systeme in Form eines von spezifischen Lösungen abstrahierten, funktionalen Mehrschichtenmodells vorgestellt.

2.4.3 Grundlagen zur Kontextadaption

Die wachsenden technologischen Fortschritte und die damit verbundene Integration von Software-Systemen in viele Bereiche unseres täglichen Lebens verlangen nach einem höheren Grad an Flexibilität bzw. an Multifunktionalität der Systeme. Erste Ansätze bestehen darin, die vielen Funktionalitäten direkt in die Systeme abzubilden und sie kontinuierlich bereitzustellen. Damit stößt man aber sehr schnell an die Grenze der verfügbaren Ressourcen. Als Verbesserung dieses Ansatzes bietet sich die Gruppierung zusammengehörender Funktionalitäten des Systems und daraus sinnvolle Konfigurationen zu bilden, so dass bei Bedarf die jeweils passende Konfiguration ausgewählt wird. Allerdings erfordert die Auswahl der passende Konfiguration weitere Interaktionen des Nutzers mit dem System, was man ebenfalls nicht erzielen möchte. Denn das eigentliche Ziel, das verfolgt wird, ist es, die Fülle von Funktionalitäten der Systeme situationsgerecht bereitzustellen, und dabei die direkte Interaktion des Nutzers mit dem System auf das minimal Notwendige zu reduzieren.

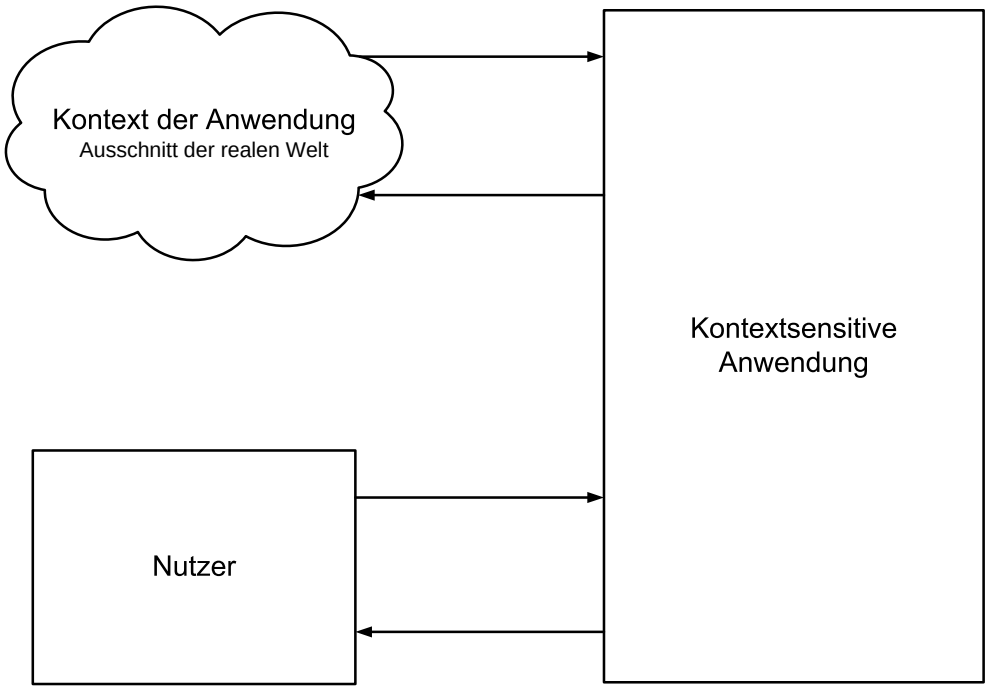

Abbildung 2.3: Grundkonzept kontextsensitiver Anwendungen

Weitere Überlegungen zur Erreichung dieses Ziels führen zur situationsgerechten Automatisierung der Auswahl der geeigneten Konfiguration. Dadurch entsteht eine neue Art interaktiver Systeme, die so genannten kontextadaptiven bzw. kontextsensitiven Systeme. Während die Kommunikation zwischen interaktiven Systemen und ihren Nutzern (Benutzern oder anderen technischen Systemen) im Normalfall von den Betei-

ligten gewollt ist, und die Beteiligten dabei bewusst ein bestimmtes Ziel verfolgen bzw. eine bestimmte Wirkung erzielen wollen, kommen bei kontextsensitiven Anwendungen zusätzliche Merkmale hinzu. Den Beteiligten einer Interaktion ist die Kommunikation nicht notwendigerweise bewusst. Auch der Weg, der zum verfolgten Ziel führt, ist in der Regel nicht bekannt. Kontextsensitive Anwendungen, wie sie in dieser Arbeit betrachtet werden, sind also interaktive Systeme mit der besonderen Eigenschaft, neben der direkten Kommunikation mit dem Nutzer, auch den Kontext der Kommunikation zu berücksichtigen, um ihr beobachtbares Verhalten anzupassen (siehe Abbildung 2.3).

Aus der Perspektive des Nutzers weisen diese Anwendungen ein nichtdeterministisches Verhalten auf, weil sie unter Umständen auf die gleiche Nutzereingabe (in identischen Situationen aus Sicht eines Nutzers), unterschiedlich verhalten können. Als Beispiel hierzu betrachten wir eine kontextsensitive Schranke eines Parkhauses. Eine solche Schranke kann sich aus Sicht des Nutzers (hier eines Autofahrers) nichtdeterministisch verhalten: Mal öffnet sie sich automatisch, wenn der Fahrer mit seinem Auto in der Nähe der Schranke kommt, und mal nicht, obwohl es aus Sicht des Fahrers die gleiche Situation herrscht. Dieses nichtdeterministisches Verhalten der Schranke kommt von daher, dass das Öffnen der Schranke von zusätzlichen Informationen abhängt (dem Kontext der Nutzung), die für den Fahrer nicht unbedingt ersichtlich sind. Im Abschnitt 3.2 kommen wir ausführlicher auf diese Besonderheit kontextsensitiver Anwendungen zurück.

Definition 2.3 Kontext

☞ Kontext ist die hinreichend genaue Charakterisierung der Situationen eines Systems anhand von für die automatische Anpassung des Verhaltens des Systems relevanten und von dem System wahrnehmbaren Informationen.

Kontext in diesem Zusammenhang ist wie in Definition 2.3 angegeben zu verstehen und ist angelehnt an [Dey et al., 2001]. Als Kontext gilt also jede Information, die dazu verwendet werden kann, die Situation von Entitäten zu charakterisieren, die für die Interaktion zwischen einem Nutzer und einer Anwendung relevant sind. Dazu gehören mittelbar auch der Nutzer und die Anwendung selbst, und dabei kann eine Entität eine Person, ein Ort oder ein Objekt sein. Diese Definition impliziert eine Kategorisierung der Eingaben in ein System nach direkten und indirekten Eingaben. Die indirekten Eingaben kommen aus der Systemumgebung und werden verwendet, zusätzlich zu den direkten Eingaben, um die Kommunikation zwischen dem Nutzer und dem System anzureichern. Behandelt man den Nutzer als Teil der Systemumgebung, so ist das Modell des Kontextes eines Systems ein Datenmodell, das die Beschreibung relevanter Aspekte der Systemumgebung inkl. dem Nutzer ermöglicht.

Mit der Anreicherung der direkten Interaktion zwischen dem Nutzer und dem System mit zusätzlichen Kontextinformationen, wird das Ziel verfolgt die Menge der direkten Interaktionen zu reduzieren, und somit dem Nutzer zu entlasten. Die Wirkung der direkten Interaktion im System wird durch die über Kontextinformationen charakterisierten Situation beeinflusst und gegebenenfalls angepasst. Diese Anpassung wird Adaption genannt; dabei ist der Anpassungsprozess gemeint. Im alltäglichen Sprachgebrauch versteht man unter Adaption ein Anpassungsvermögen an die Umwelt. Ein Anpassungsvermögen steht für die Fähigkeit, sich den gegebenen Verhältnissen an-

zupassen bzw. sich nach jeweiligen Umständen zu richten. Die allgemeine Fähigkeit eines Systems, sich in einer bestimmten Situation nach den durch die Umgebung vorgegebenen Verhältnissen zu richten, ist die Adaptivität dieses Systems.

Definition 2.4 Kontextadaption

☞ *Kontextadaption* bezeichnet die automatisierte Anpassung des beobachtbaren Verhaltens eines Systems an seinen Kontext. Das Verhalten (Ein-/Ausgabeverhalten) des Systems ist bestimmt durch die an der Schnittstelle beobachtbaren Ein- und Ausgaben. Die Ausgaben des Systems hängen dabei nicht nur von den Nutzereingaben ab, sondern auch von den über Sensorik gesammelten Kontextinformationen.

Der gesamte Prozess der Kontextadaption umfasst die Schritte der Informationserfassung, Situationserkennung, Adaptionsumsetzung und der Rückkopplung (Siehe Abbildung 2.4) [Fahrmair et al., 2006a].

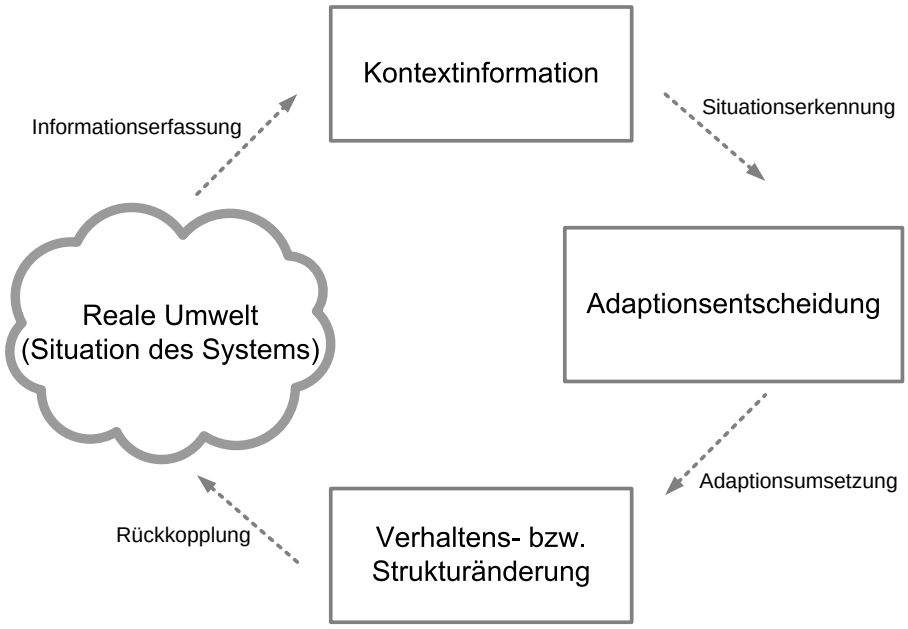

Abbildung 2.4: Der Prozess der Kontextadaption

Die Erfassung von Informationen erfolgt mittels Sensoren und die erfasste Information wird in einem Kontextmodell festgehalten. Ausgehend von den aktuellen Werten, die in das Kontextmodell abgebildet sind, wird eine Situation identifiziert. Eine Situation ist somit eine Instanz des Kontextmodells. Abhängig von der erkannten Situation wird entschieden, inwiefern das Verhalten der Anwendung angepasst werden soll. Eine solche Entscheidung nennt sich Adaptionsentscheidung und gilt als Voraussetzung für den nächsten Schritt, den Schritt der Adaptionsumsetzung. Dieser Schritt resultiert in einer Änderung des Systemverhaltens, etwa der Reduzierung der Qualität der bereitgestellten Dienste, in bestimmten Fällen sogar in einer Änderung des Systemstruktur, etwa der Änderung der Kommunikationskanäle. Wichtig ist zu merken, dass

jede Änderung des Systemverhaltens eine automatische Rückkopplung in der realen Welt bewirkt.

Die Automatisierung des Anpassungsprozesses realisiert durch Kontextadaption (siehe Definition 2.4) bringt insbesondere den Vorteil, dass dem Nutzer einen Teil der direkten Interaktion mit dem System abgenommen wird. Gleichzeitig stellt die Kontextadaption besondere Herausforderungen, die wir im Kapitel 3 näher erläutern werden. Darunter zählen das besagte nicht-deterministische Verhalten der Anwendung, die Vielfältigkeit des Kontextes und die Häufigkeit des Auftretens von unerwünschtem Systemverhalten. Zuvor werden wir uns im nächsten Abschnitt mit der logischen Architektur kontextsensitiver Anwendungen beschäftigen.

2.4.4 Konzeptueller Aufbau kontextsensitiver Systeme

Betrachtet man kontextsensitive Systeme, wie sie uns in Rahmen dieser Arbeit interessieren, so sind sie in erster Linie reine interaktive Systeme (siehe Abbildung 2.5 a-). Sie interagieren, genau wie alle anderen interaktive Systeme, mit ihrem Nutzer über eine Nutzerschnittstelle basierend auf einem festgelegten Protokoll. Nutzer in diesem Zusammenhang kann sowohl ein menschlicher Nutzer, auch Benutzer genannt, als auch ein weiteres System sein. Bei kontextsensitiven Systemen kommt ein erstes Unterscheidungsmerkmal bezüglich der Eingabe hinzu. Es wird unterschieden zwischen Nutzereingaben und Kontexteingaben. Nutzereingaben sind solche Eingaben, die direkt vom Nutzer gemacht werden, um das System unmittelbar zu nutzen. Kontexteingaben hingegen sind solche Eingaben in das System, die aus dem Kontext entnommen werden. Sie werden also nicht direkt von dem Nutzer gemacht. Die direkte Interaktion zwischen dem Nutzer und dem System wird explizit mit zusätzlichen Informationen über den Nutzungskontext angereichert (siehe Abbildung 2.5 b-). Das Verhalten des Systems hängt dann nicht mehr allein von den Nutzereingaben ab (explizite Interaktion), sondern auch von den zusätzlichen Kontexteingaben (implizite Interaktion). Die Systemreaktion auf eine Nutzereingabe hin hängt von den über Sensorik gesammelten Kontextinformationen ab. Die Unterscheidung zwischen impliziten und expliziten Interaktion ist lediglich eine erste Differenzierung kontextsensitiver Systeme gegenüber ganz normalen interaktiven Systemen.

Eine Differenzierung gegenüber ganz normalen eingebetteten Systemen erfordert allerdings weitere Unterscheidungsmerkmale. Diese Merkmale werden offensichtlich, wenn wir die logische Architektur kontextsensitiver Systeme betrachten. Kontextsensitive Systeme können logisch in zwei Schichten strukturiert werden: eine Schicht für die Adaption des Systems und eine Schicht für die Grundfunktionen des Systems. Die Systeme sind also konzeptuell dadurch charakterisiert, dass sie neben ihren Grundfunktionen und der Nutzerschnittstelle, auch ein Adaptionsteilsystem enthalten. Die Menge der Grundfunktionen bildet den *Systemkern*. Die Nutzerschnittstelle ist für die Schließung der Brücke zwischen dem Systemkern und dem Nutzer zuständig. Das Adaptionsteilsystem ist für die Anreicherung der Grundfunktionen mit weiteren situationsbezogenen Informationen und Entscheidungen zuständig [Trapp und Schürmann, 2003]. Abbildung 2.6 illustriert diese konzeptuellen Teilsysteme samt ihrer Beziehungen unter einander. Die Adaptionsschicht wirkt als Filter zwischen den Grundfunktionen und der Systemumgebung. Sie filtert die Kommunikation zwischen dem System und seiner Umgebung, inklusive dem Nutzer, mittels der Adap-

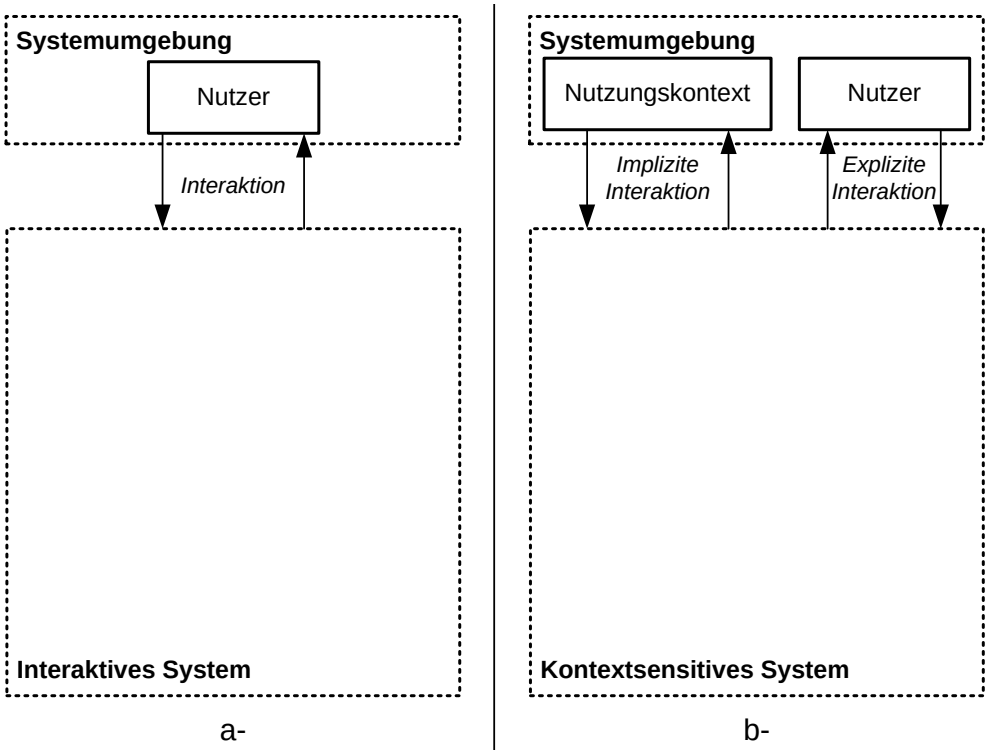

Abbildung 2.5: Kontextsensitive Anwendung: eine erste Differenzierung

tionslogik auf Basis des Kontextmodells. Das Kontextmodell wird also dazu verwendet, die Ein- und Ausgaben des Systems zu manipulieren bzw. die direkte Interaktion zwischen dem Nutzer und dem System anzureichern. Die dem System über Sensorik bereitgestellten Informationen dienen explizit dazu, das System entsprechend zu adaptieren. Eine solche Adaption besteht darin, aus den verfügbaren Grundfunktionen die gerade passenden auszuwählen und sie eventuell anzupassen.

In speziellen Fällen enthalten kontextadaptive Systeme zusätzlich zu den hier aufgeführten Teilsystemen ein *Kalibrierungsteilsystem*, das für eine nachträgliche Anpassung der Logik des Adaptionsteilsystems [Fahrmair, 2005, Fahrmair et al., 2006a] verantwortlich ist. Diese optional Teilsystem kann technisch durch einen zusätzlichen Kalibrierungskanal zwischen der Nutzerschnittstelle und dem Adaptionsteilsystem realisiert werden.

Definition 2.5 Kalibrierung

☞ Die Kalibrierung eines kontextadaptiven Systems ist die nachträgliche manuelle Anpassung der zugrunde liegenden Logik der Adaption des Systems. Sie ist also eine Adaption der Adaption, und erfordert eine entsprechende Expertise über das Adaptionsverhalten des Systems.

Die Kalibrierung, wie sie in Definition 2.5 definiert ist, bewirkt eine Anpassung des Kontextmodells und/oder der Adaptionslogik. Sie erfolgt manuell und zeitversetzt zur Nutzungszeit des Systems und erfordert eine gewisse Expertise. Sie ist ähnlich einer neuen Programmierung des Systems und kann nicht durch das System selbst sti-

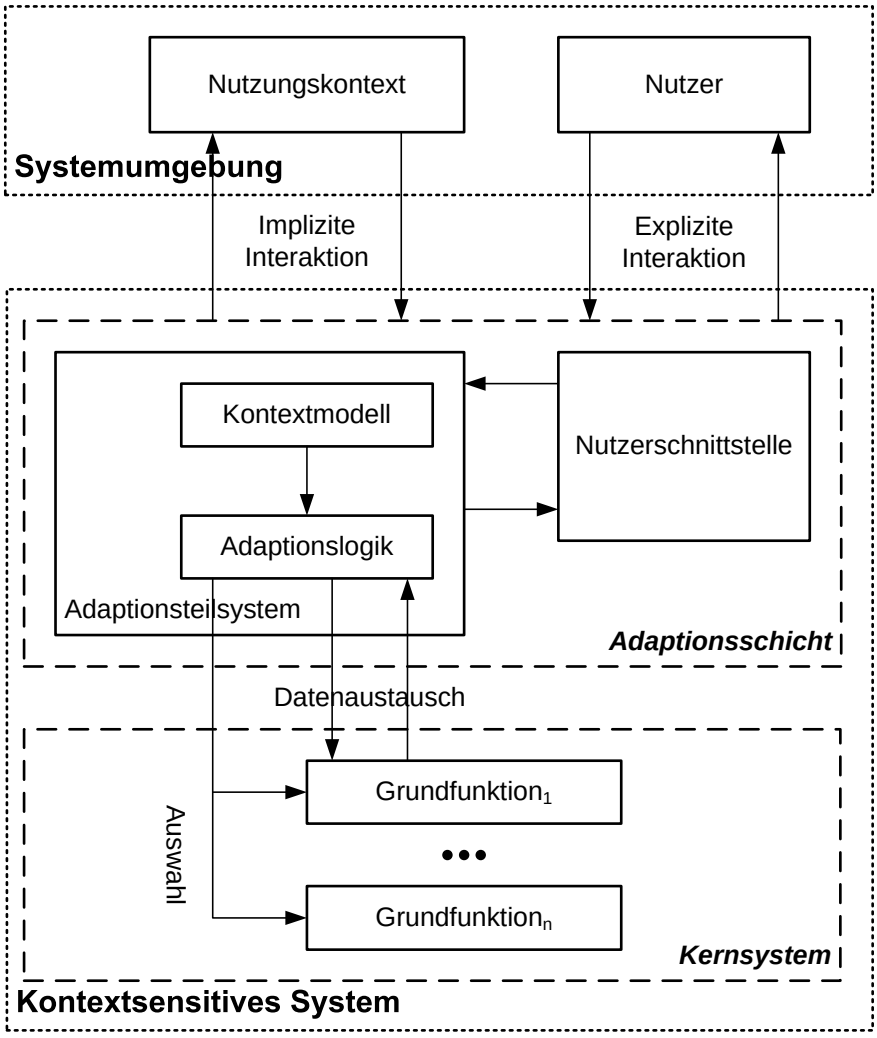

Abbildung 2.6: Logische Architektur kontextsensitiver Systeme

muliert werden. Aufgrund bestimmter Änderungen des Kontextes kann die Berücksichtigung von zuvor nicht weggelassenen Kontextinformationen bei den Adaptionsentscheidungen durch eine Kalibrierung erzielt werden. Dies führt zur Hinzunahme der neuen Größe im Kontextmodell und zur entsprechenden Anpassung der Adaptionslogik, etwa weil man zukünftig den Verkehrsfluss (Verkehrstau, stockender Verkehr, freie Fahrt ...) mit in der Routenberechnung berücksichtigen möchte. Die Kalibrierung kann jedoch auch eingesetzt werden, um allein die Adaptionslogik anzupassen, etwa weil eine effizientere Rechenregel einer Navigationsroute erfunden wurde.

Wie bereits erwähnt, wird bezüglich der Kommunikation zwischen dem System und seiner Umgebung zwischen Eingaben an das System, die direkt durch den Nutzer getätigt werden, und solchen, die aus dem Nutzungskontext erfolgen unterschieden. Im letzteren Fall kommuniziert die Systemumgebung nicht gezielt mit dem System und verfolgt damit auch kein bestimmtes Ziel. Demnach hat das System gewisse Annahmen über die Bedeutung und die Relevanz der Kontexteingaben zu machen, und eine situationsgerechte Anpassung seines eigenen Verhaltens vorzunehmen. Dies erfolgt mittels dem Kontextmodell und der Adaptionslogik. Im RE kontextadaptiver Anwen-

dungen muss daher geklärt werden:

1. Welche Informationen, charakteristisch für die jeweiligen Einsatzsituationen, sind als Kontext zu betrachten?
2. Welche Informationen sind als direkter Eingaben vom Nutzer in das System erforderlich[2]?
3. Inwiefern können beide Art von Informationen das Verhalten des Systems bezogen auf eine Situation beeinflussen?

Geeignete Ansätze zur Erlangung derartigen Informationen, die die Interaktion zwischen dem Nutzer und dem System sowie das Systemverhalten beeinflussen, müssen erarbeitet werden. Der Umstand, dass im System bestimmte Aktionen ohne direkten Anstoß von dem Nutzer, bzw. ohne für den Nutzer nicht direkt nachvollziehbare Ereignisse in der Einsatzumgebung, ausgeführt werden, ist bereits im RE in Rechnung zu tragen. Im Bereich des Context-Aware Computing wird die Fragestellung der Festlegung der relevanten Kontextinformation als charakteristische Herausforderung eingestuft [Shen und Shen, 2001], [Krause, 2006]. Auf diese und weitere Herausforderungen kontextsensitiver Anwendungen gehen wir im Kapitel 3 ein.

2.5 Begleitende Fallstudien

Die im Laufe der Arbeit entwickelten Konzepte und Methoden des Requirements Engineering kontextsensitiver Anwendungen, inklusive der Kontextmodellierung, werden anhand von Fallstudien erprobt. Die Konzepte werden anhand geeigneter Beispiele aus diesen Fallstudien illustriert. Eine einführende Beschreibung der Fallstudien erfolgt in diesem Abschnitt.

2.5.1 Ein kontextsensitiver Terminplaner

Mit dem Comtext Aware Task Scheduler (CATS) wird das Ziel verfolgt, Benutzer mit einem kontextbezogenen Terminplaner auszustatten, welcher den persönlichen Tagesablauf möglichst raffiniert organisiert.

Der tägliche Terminplan des Benutzers soll so nützlich, aktuell und konsistent wie möglich gestaltet werden. Das System besteht aus einem Server (CATSS), welcher vom Betreiber bereitgestellt wird und mehreren Clients (CATSC), welche auf diversen mobilen Endgeräten wie Notebooks, Mobiltelefone und Pocket PC's laufen. Der CATS-Server fungiert als eine Art Kontext-Server, welcher relevante Informationen bezüglich Events wie etwa eine Pressekonferenz oder Fußballspiel speichert und verwaltet. Sobald sich Daten bezüglich der verwalteten Events ändern, informiert der Server alle registrierten Clients über die durchgeführten Änderungen, so dass die Benutzer rechtzeitig auf die Änderungen reagieren können. Ferner verfügt jeder CATSC über die Möglichkeit, audio-visuelle Ausgabegeräte in seiner Umgebung zu nutzen, um mit dem Benutzer jeweils über einen der Situation angemessenen Weg zu kommunizieren.

[2]Diese Fragestellung ist primär als Pendant zur ersten Fragestellung zu sehen, allerdings ist ihre explizite Behandlung wichtig für die Feststellung der Interaktionsfreiheit des Nutzers.

2.5.2 Eine kontextsensitive Scheibentönung

Die Sichtbehinderung durch die blendende Sonne stellt eine Gefährdung im Straßenverkehr dar und führt häufig zu Unfällen. In den Wintermonaten wird der Fahrer durch die tief stehende Sonne geblendet, im Sommer ist die Sicht des Fahrers aufgrund der hohen Lichtintensität stark eingeschränkt.

Sonnenbrillen sind ein weit verbreitetes Mittel, um der Sichtbehinderung durch die Sonne entgegenzuwirken. Auch die Sonnenblenden innerhalb des Fahrzeugs verbessern die Lichtsituation. Dennoch sind beide Möglichkeiten lediglich suboptimal, da ihr Einsatz die Konzentration des Fahrers (zumindest kurzfristig) beeinflusst: der Fahrer muss die Sonnenbrille während der Fahrt aufsetzen (im schlimmsten Fall ist bei einem Fahrer mit Sehschwäche damit auch ein Brillenwechsel verbunden, d.h. seine Sehkraft ist kurzzeitig nicht ausreichend), die Sonnenblende muss ebenfalls während der Fahrt heruntergeklappt werden. Besonders deutlich wird dieser Aufwand an kognitiver Aufmerksamkeit bei häufig wechselnden Lichtverhältnissen (Wechsel zwischen hell und dunkel in zeitlich kurzen Abständen). Der Fahrer kann sich dann nicht mehr ausreichend auf die Verkehrssituation und das eigentliche Autofahren konzentrieren. Dementsprechend steigt das Unfallrisiko.

Das kontextsensitive Scheibentönungssystem *(Context-Aware Shading System – CASHAS)* soll den Fahrer vor einer Blendung durch die Sonne schützen, indem es die Scheiben des Fahrzeugs entsprechend der Lichtverhältnisse (Lichtintensität, Lichteinfallswinkel), der Präferenzen des Fahrers und weiterer Einflussgröße tönt. Außerdem kann die Scheibentönung bei geparktem Fahrzeug als Diebstahlschutz fungieren, indem die Scheiben bis zu einer Lichtdurchlässigkeit von 0% getönt werden und der Einblick von außen ins Fahrzeuginnere dadurch verhindert wird.

Kapitel 3

Herausforderungen im RE kontextsensitiver Anwendungen

If one is master of one thing and understands one thing well,
one has at the same time, insight into and understanding of many things.

Vincent Van Gogh

Inhaltsangabe

3.1 Einleitung

Kontextsensitive Anwendungen werden konzipiert, um in dynamischen Umgebungen operieren zu können und dabei nur die minimal notwendige Aufmerksamkeit und Interaktionen ihrer Nutzer zu verlangen. Zudem sollen die betrachteten Anwendungen meistens eine Vielfalt an Funktionalitäten anbieten, und müssen dabei mit teilweise sehr eingeschränkten Ressourcen (z.B. Netzwerk-Bandbreite, Laufzeit) zur Erfüllung der Nutzerbedürfnisse auskommen. Um diesen Anforderungen zu genügen,

wird die Technik der Kontextadaption eingesetzt, welche abhängig von dem aktuellen Nutzungskontext ein adäquates Systemverhalten aus einer Menge von möglichen Systemverhalten auswählt.

Das beobachtbare Verhalten einer kontextsensitiven Anwendung hängt dadurch nicht nur von den Eingaben des Nutzers ab, sondern auch von ihrem Nutzungskontext. Aus Sicht des Nutzers, welcher diese zusätzlichen Kontexteingaben des Systems u.U. nicht kennt bzw. nicht beobachten kann, reagiert die Anwendung auf die gleiche Nutzereingabe *scheinbar* unterschiedlich. Das Verhalten der Anwendung wird vom Nutzer als nicht-deterministisch empfunden. Der Kontext ist allerdings sehr vielfältig, und seine Modellierung, ein notwendiger Schritt in der Entwicklung kontextsensitiver Anwendungen, erfordert eine sorgfältige Betrachtung von allen für die Adaption des Anwendung relevanten Informationen. Angesichts eines gewissen Grades an Autonomie (das System entscheidet autonom über die Auswahl der geeigneten Systemreaktion), welcher mit der Kontextadaption einhergeht, kommt es häufiger als bei konventionellen Anwendungen zu unerwünschtem Verhalten.

Im Abschnitt 3.2 werden zunächst die Eigenheiten kontextsensitiver Anwendungen aufgeführt, mit besonderem Augenmerk auf den aus Nutzersicht wahrgenommenen Nichtdeterminismus. Dieser resultiert aus der Tatsache, dass die Systemreaktion neben den Nutzereingaben auch von über Sensorik gewonnenen Kontextinformationen abhängt, welcher dem Nutzer nicht bekannt sind. Wenn der Nutzer im Umkehrschluss wüsste, wie sich bestimmte Kontextinformationen auf die Systemreaktion auswirken, würde sicherlich keine Nichtdeterminismus wahrgenommen.

Bisherige Forschungsarbeiten, die sich mit Context-Awareness befasst haben, haben unterschiedliche Taxonomien des Kontextbegriffs aufgestellt. Diese Taxonomien werden zusammen mit den für diese Arbeit relevanten Besonderheiten von Kontext im Abschnitt 3.3 aufgeführt.

Aufgrund des Automatismus, der hinter der Nutzung von Kontextinformationen zur Anpassung des Verhaltens der Anwendungen steckt, kann das Phänomen des unerwünschten Verhaltens bei kontextsensitiver Anwendungen stärker zutreffen. Diese Problematik wird im Abschnitt 3.4 näher untersucht.

3.2 Besonderheiten kontextsensitiver Anwendungen

Kontextsensitive Anwendungen besitzen die Fähigkeit, ihr beobachtbares Verhalten mit dem Ziel zu ändern, sich an veränderliche Gegebenheiten der Einsatzumgebung automatisch anzupassen. Betrachtet man diese Art von Anwendungen etwas genauer, so stellt man fest, dass sie eine Reihe von charakteristischen Eigenschaften aufweisen, auf welche im Nachfolgenden näher eingegangen wird.

3.2.1 Nicht-Determinismus aus Sicht des Nutzers

Eine Eigenschaft kontextsensitiver Anwendungen betrifft ihr aus Nutzersicht *nichtdeterministisches Verhalten*. Eine kontextsensitive Anwendung weist in bestimmten Situationen aus der Perspektive des Nutzers unterschiedliche Verhalten auf. Zum Beispiel kann die Anwendung auf dieselben Eingabe des Nutzers unterschiedlich rea-

gieren. Der Nutzer ist zwar in der Lage, die geänderte Systemreaktion auf identische Nutzereingaben zu beobachten, die Gründe der Änderungen (veränderter Kontext) bleiben für ihn jedoch verborgen.

Ein illustrierendes Beispiel für das nicht-deterministische Verhalten kontextsensitiver Anwendungen aus der Perspektive des Nutzers betrifft ein kontextsensitives Navigationssystem im Fahrzeug. Das System wird beispielsweise jeden Freitag verwendet, um den Fahrer von seiner Arbeitsstelle zu einem wöchentlichen Treffen bei seinem Partner zu navigieren. Dabei führt ihn das System mal über die Autobahn, mal über die Landstraße. Dies ist beispielsweise der Fall bei Navigationssystemen der *TomTom Go*™ Familie mit *IQ Routes*™ -Technologies. Jedes Mal, wenn der Fahrer in das Auto steigt und sein Ziel eingibt, weiß er a priori nicht, welche Route das System dieses Mal vorschlagen wird. Er folgt jedoch stets der vom System vorgeschlagenen Route, welche auch ans Ziel führt.

Das rätselhafte Verhalten des Navigationssystems kann viele Gründe haben, die jedoch für den Nutzer nicht sofort nachvollziehbar sind. Ein möglicher Grund für dieses nicht-deterministische Verhalten ist die Berücksichtigung zusätzlicher Informationen zur Berechnung der Route – abgesehen von der eigentlichen Zielangabe. Die zusätzlichen Informationen können Stau-Meldungen, Informationen zu Baustellen, Bergungsarbeiten oder ähnliches sein. Das Verhalten des Systems hängt also von zusätzlichen Informationen ab, welche aber aus der Perspektive des Fahrers nicht ersichtlich sind.

Der Nichtdeterminismus aus der Perspektive des Nutzers kann jedoch behoben werden, indem idealerweise dem Nutzer erklärt wird, welche Faktoren bei der Anpassung des Verhaltens eine Rolle spielen. Demnach wäre der Nutzer nicht nur in der Lage, Änderungen der Reaktion des Systems auf seine Eingaben zu beobachten, sondern auch Änderungen der weiteren Einflussgrößen zu beobachten. Der Nutzer wäre somit über die Abhängigkeiten zwischen Änderungen der Einflussgrößen und Änderungen des beobachtbaren Verhaltens des Systems informiert und wäre in der Lage, die vorgenommenen Adaptionen nachzuvollziehen. Wir reden in diesem Fall von einer *transparenten* Adaption. Abbildung 3.1 illustriert diese Sichtweise.

Im vorherigen Beispiel des kontextsensitiven Navigationssystems, würde eine *transparente* Adaption wie folgt aussehen. Würde sich der Nutzer – in diesem Fall der Fahrer – beispielsweise anhand des Benutzerhandbuchs darüber informieren, dass das System auch Informationen über den Verkehrsfluss für die Berechnung der Route berücksichtigt, kann er die Systemreaktionen besser nachvollziehen. Dementsprechend könnte der Fahrer über das Autoradio informiert worden sein, dass eine Stausituation auf der Autobahn eingetreten ist. Den resultierenden Vorschlag des Systems, diesen Stau durch die Benutzung der Landstraße zu umfahren, wäre dadurch sofort nachvollziehbar. Die Entscheidung des Systems würde aufgrund des erkannten Zusammenhangs nicht mehr als nicht-deterministisches Verhalten empfunden. Der Fahrer würde wahrscheinlich eher von einem *situationsbedingten* Verhalten des Navigationssystems sprechen.

Aufgrund der Kenntnis des Nutzers über die Faktoren, die das Verhalten des Systems beeinflussen, ist es jedoch nicht auszuschließen, dass der Nutzer diese Faktoren unter Umständen mittelbar beeinflussen könnte, um ein bestimmtes Verhalten des Systems zu erzielen. Der Nutzer würde also das System über diese Kanäle (Kontextkanäle) nutzen, die nicht hierfür vorgesehen wurden. Die Nutzung solcher Kanäle könnte unter

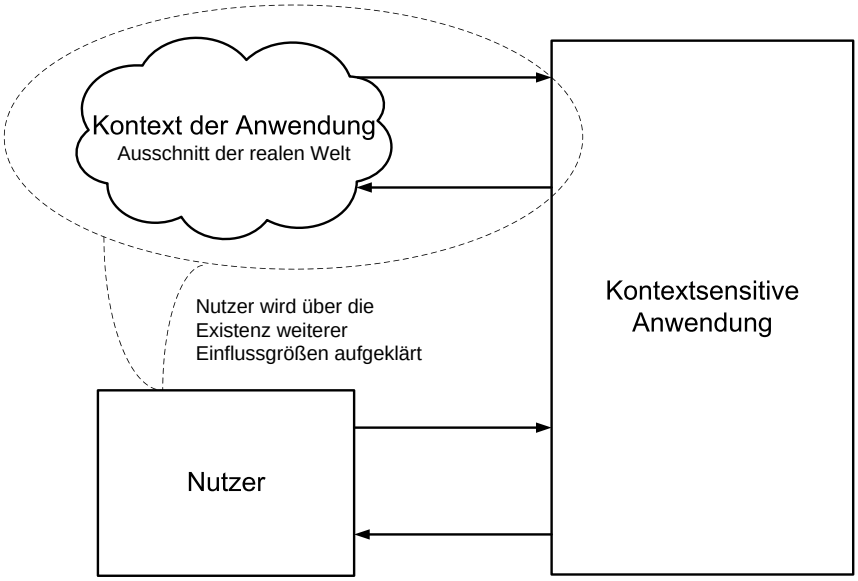

Abbildung 3.1: Erläuterung des Nichtdeterminismus

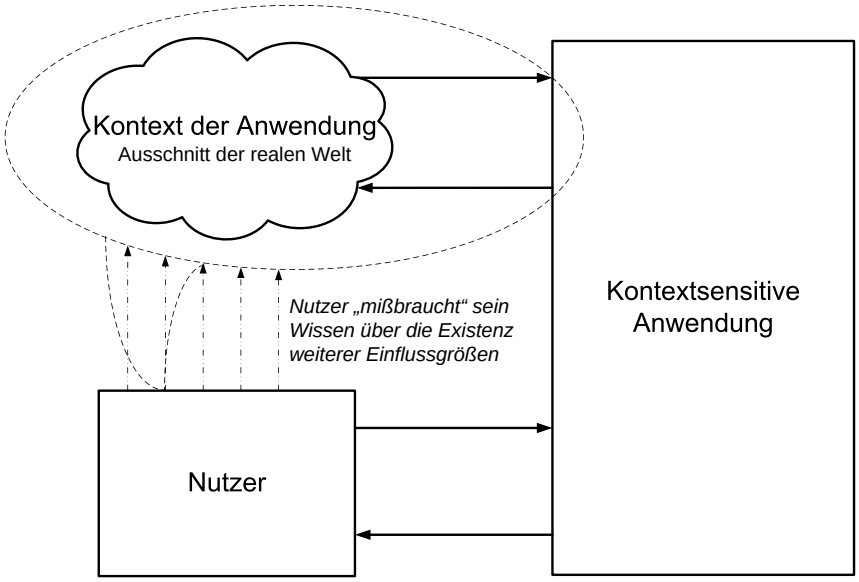

Abbildung 3.2: Mittelbare Beeinflussung der Zusatzfaktoren

Umständen unangenehme Effekte haben. Wir reden in diesem Fall von einer *kontrollierbaren* Adaption. Eine Illustration hiervon ist in Abbildung 3.2 gegeben.

Als Beispiel hierfür betrachten wir ein kontextsensitives Radio-System im Fahrzeug, das seine Lautstärke unter anderem abhängig von der Motordrehzahl automatisch regelt. Der Nutzer weiß von dieser Abhängigkeit und könnte dieses Wissen nutzen, um das System zu bedienen, z.B. in dem er bei Bedarf einfach schneller fährt (d.h. die Motordrehzahl erhöht), um das Radio lauter zu stellen.

Entsprechend der in diesem Abschnitt beschriebenen Klassifizierung von Adaptionen

werden die Nutzer der Anwendung klassifiziert. Eine solche Klassifizierung ist für die Modellierung des Nutzers von Bedeutung, zumal die Kontextinformationen, die verwendet werden, um die Interaktion zwischen dem Nutzer und dem System zu adaptieren, auch Informationen über den Nutzer und sein Benutzungsverhalten beinhalten. Dabei ist zwischen *gewöhnlichen Nutzern* (das Systemverhalten muss nicht nachvollziehbar sein), *vorsichtigen Nutzern* (das Systemverhalten ist nachvollziehbar aber diese Nutzer wollen es nicht manipulieren) und *erfahrenen Nutzern* (sie wollen das Systemverhalten durch bewusste Manipulation des Kontextes beeinflussen) zu unterscheiden.

Eine Formalisierung der in diesem Abschnitt aufgeführten Klassifizierung kontextsensitiver Anwendungen auf Basis der mathematisch fundierten Ansätze von FOCUS und JANUS (siehe [Broy und Stølen, 2001, Broy, 2005]) ist in [Broy et al., 2009] gegeben.

3.2.2 Weitere Besonderheiten

Verglichen mit anderen Anwendungen (d.h. Anwendungen ohne Abhängigkeit zwischen Änderungen im Nutzungskontext und beobachtbarem Verhalten) haben kontextsensitive Anwendungen eine Reihe weiterer Besonderheiten. Diese Besonderheiten hängen vor allem mit der Kontextbezogenheit der Anwendungen zusammen. In [Rodden et al., 1998, Pascoe et al., 1999] und [Dey und Abowd, 2000] wurden sie zum Teil ausführlich diskutiert. Sie wurden ebenfalls in zahlreichen Forschungsprojekten im Kompetenzzentrum *Mobility & Context Awareness* am Informatik-Lehrstuhl für Software & Systems Engineering der TU-München [C2MCA, 2008] bestätigt. Im Folgenden führen wir einige weitere Besonderheiten kontextsensitiver Anwendungen kurz auf.

Hoher Bedarf an Ressourcen

Jedes kontextsensitive (Software-)System beobachtet seine Umgebung. Für gewöhnlich erfolgt dies unter Zuhilfenahme von Sensoren, welche die aktuell gemessenen Werte bestimmter Umgebungsausprägungen liefern. Die Abfrage kann periodisch oder kontinuierlich erfolgen. Hinzu kommen u.a. mögliche Weiterverarbeitung der Sensordaten und die Herleitung einer jeweiligen Adaptionsentscheidung.

Dynamik und begrenzte Nutzerinteraktion

Das zeitliche Verhalten der Anwendungen kennzeichnet ihre Dynamik. Dieser Aspekt hängt stark damit zusammen, dass der Kontext der Anwendungen hochgradig veränderlich ist (siehe Abschnitt 3.3). Der Kontext kann unter Umständen sehr komplex werden, was sich negativ auf die Aufmerksamkeit und Interaktionsfreiheit des Nutzers auswirken kann [Chen und Kotz, 2000, Shen und Shen, 2001, Kolos-Mazuryk et al., 2006].

Wie bereits erwähnt, hängen die aufgeführten Eigenschaften kontextsensitiver Anwendungen sehr stark von ihrem Kontext und seinen Besonderheiten ab. Die Besonderheiten des Kontextes erschweren eine klare Definition des Begriffes, sowie die Erstellung eines umfassenden und gleichzeitig praktikablen Modells davon. Im folgenden Abschnitt besprechen wir bisherige Definitionen von Kontext und führen die für die diese Arbeit relevanten Eigenheiten des Konzeptes auf.

3.3 Kontext und seine Eigenheiten

Das zentrale Konzept in Zusammenhang mit kontextsensitiven Anwendungen ist der Kontextbegriff. Eine präzise Definition dieses Begriffs und eine sorgfältige Behandlung seiner Dynamik sollten einen hohen Stellenwert in der Entwicklung kontextsensitiver Anwendungen einnehmen. Bisherige Arbeiten auf dem Gebiet des Context-Aware Computing, die sich mit Kontext beschäftigt haben, haben den Begriff immer wieder neu definiert. Die jeweiligen Definitionen waren entweder eine Erweiterung bestehender Definitionen um neue Aspekte oder eine Beschränkung um bestimmte Aspekte. Dies liegt vor allem an der Komplexität und der Vielfalt des Begriffs. In diesem Abschnitt fassen wir diese Definitionen zusammen und heben die besonderen Eigenheiten des Begriffs hervor.

3.3.1 Der Kontextbegriff im Rückblick

Inspiriert von Mark Weiser's Vision vom *Ubiquitous Computing*, führen Schilit und Theimer im Rahmen des ParcTab-Projekts [Schilit et al., 1993] die Idee von Anwendungen ein, die sich an Änderungen in Systemkonfigurationen und ihrer Umgebung anpassen. Bereits zwischen 1989 und 1992 wurde im Rahmen eines anderen Forschungsprojekts ein erstes Prototyp eines kontextsensitiven Anwendung entwickelt (*the active badge system* [Want et al., 1992]). Das System war sensitiv bzgl. des Aufenthaltsorts der Mitarbeiter in dem Firmengebäude. Als Beispielanwendung wird ein automatisiertes Büro (*automated office*) aufgeführt, in dem ankommende Anrufe sofort auf dasjenige Telefon weitergeleitet werden, das am nächsten zu dem aktuellen Aufenthaltsort des Angerufenen steht. Innerhalb dieser Projekte entstand dann das Forschungsgebiet des Context-Aware Computing, und erste Definitionen des Kontextbegriffs wurden eingeführt. Diese Definitionen wurden in darauf folgenden Arbeiten revidiert und durch Ergänzungen erweitert. In diesem Abschnitt lassen wir die gebräuchlichen Definitionen Revue passieren.

Zunächst betrachten wir den Kontextbegriff aus einer linguistischen Perspektive. In Duden Band 5 – Das Fremdwörterbuch ist die folgende Definition für Kontext angegeben [Duden, 2001]: *(1) (Spachw.) ... c) der umgebende inhaltliche (Gedanken-, Sinn)zusammenhang, in dem eine Äußerung steht, und der Sach- und Situationszusammenhang, aus dem heraus sie verstanden werden muss. (2) umgebender Zusammenhang, z.B. den Menschen aus dem sozialen - heraus verstehen*. Im Englischen wird folgende Definition für Kontext (engl. *context*) verwendet [Merriam-Webster Inc., 2007]: *(1) the parts of a discourse that surround a word or passage and can throw light on its meaning. (2) the interrelated conditions in which something exists or occurs: environment, setting.* Der erste Teil der englischen Definition kann so interpretiert werden, dass der Kontext etwas Implizites ist, was als Zusatzinformation geliefert wird. Der zweite Teil ist etwas allgemeiner Natur und gibt Synonyme für Kontext an. Ähnliches gilt auch für die Definition des Duden. Eine weitere Definition des Kontextbegriffs geht ebenfalls in eine ähnlichen Richtung [FOLDOC – Compting Dictionary, 2007]. Dort wird Kontext definiert als *„That which surrounds, and gives meaning to, something else"*. Diese Definitionen bieten allerdings eine Vielfalt von Interpretationen und können deshalb nicht ohne weiteres in der Softwareentwicklung verwendet werden – etwa um festzustellen, ob eine gegebene Information Kontext ist oder nicht bzw. welche Informationen als Kontext betrachtet werden

können und welche nicht.

Eine der ersten Definitionen des Kontextbegriffs auf dem Gebiet des Context-Aware Computing geht auf Schilit und Theimer zurück. In [Schilit und Theimer, 1994] bezeichnen sie den Nutzungsort einer Anwendung, die Sammlung benachbarter Personen und Objekte sowie deren Historie als Kontext einer Anwendung. In [Schilit et al., 1994] wurde diese Definition um erreichbare Hosts und andere computerbasierte Geräte erweitert. Die Fragestellungen (*„Where you are, who you are with and what resources are nearby“*) charakterisieren demzufolge den Kontext einer Anwendung.

Im Projekt *stick-e document* bezeichnet Kontext eine Kombination von Elementen aus der Umgebung, die der betrachteten Anwendung bekannt sind [Brown, 1996]. Aufgeführte Beispiele sind Ort, Nähe zu weiteren Objekten und Zeitpunkt der Nutzung der Anwendung. In einer späteren Arbeit der gleichen Gruppe wurden Identitäten von Personen aus dem Umfeld des Nutzers, die Tages- und Jahreszeit sowie die Temperatur als weitere Beispiele für Kontext genannt [Brown et al., 1997]. Auf welche Weise das System Informationen über seine Umgebung erlangt, wird in diesen Überlegungen offen gelassen.

[Ryan et al., 1997] betrachtet als Kontext einfach jede beliebige Sammlung von Informationen über die Umgebung eines Systems und nennt die Beispiele Ort, Zeit, Temperatur und Benutzeridentität. In einer späteren Arbeit der gleichen Gruppe steht Kontext für die unterschiedlichen Zustände einer Anwendung und ihrer Umgebung [Pascoe, 1998]. Wofür die Umgebung steht, bleibt allerdings offen. In [Pascoe et al., 1999] wird der Kontextbegriff noch gründlicher durchleuchtet und festgestellt, dass die Definition von Kontext eine sehr komplexe Angelegenheit ist, und eine einfache Auflistung möglicher Kontextinformationen unzureichend für eine Definition des Begriffes ist. Eine erste umschreibende Definition lautet: *„The subset of physical and conceptual states of interest to a particular entity“*. Alle Informationen, die automatisch vom System über Sensorik gewonnen werden, sind hier Teil des Kontextes. Sie betonen außerdem, dass Kontext ein Attribut ist, das zu einer bestimmten Entität zugeordnet werden kann. In [Rodden et al., 1998] bezeichnet Kontext einfach als die *Situation* einer Anwendung. Da hierunter viel zu verstehen ist, werden eine Reihe von Kategorien festgelegt: Infrastrukturkontext, Anwendungskontext, Systemkontext, Ortskontext und physikalischer Kontext, jeweils mit Angaben von Beispielen.

Weitere bemerkenswerte Beiträge auf dem Gebiet des Context-Aware Computing wurden von Dey und Abowd geleistet, beginnend mit dem CyberDesk-Projekt [Dey et al., 1998]. Sie definieren Kontext als *„Any information about the user and the environment that can be used to enhance the user's experiences“*. Genannte Beispiele im Rahmen des CyberDesk-Projekts sind der emotionale Zustand des Benutzers, der Aufmerksamkeitsfaktor, der Ort und die Orientierung, der Tag und die Uhrzeit sowie die Objekte und die Personen im Umfeld des Benutzers [Dey, 1998]. In seiner Dissertation liefert Anind Dey eine etwas umfassendere Definition: *„Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves“* [Dey, 2000]. In vielen Forschungsarbeiten wird diese Definition als wesentlicher Grundstein gesehen, weil sie wichtige Aspekte des Kontextbegriffs mit einbezieht und gleichzeitig eine gewisse Interpretationsfreiheit mit dem Begriff „relevant“ gibt. In der Definition wird also hervorgehoben, dass Kontext die Situation einer Entität charakterisiert. Mögliche Entitäten werden genannt, und die Idee der

Relevanz bzgl. der betrachteten Anwendung wird ebenfalls darin hervorgehoben.

Auch [Chen und Kotz, 2000] liefert eine bedeutende Definition für Kontext: *„The set of environmental states and settings that either determines an application's behavior or in which an application event occurs and is interesting to the user"*. In dieser Definition wird hervorgehoben, dass Kontext einen Satz von Umgebungszuständen und Einstellungen bezeichnet. Auch die Beeinflussung des Verhaltens einer Anwendung durch den Kontext wird hier hervorgehoben. Ein besonderer Aspekt, der ebenfalls in dieser Definition enthalten ist, betrifft die Bedeutung des Kontextes für den Nutzer.

Im Rahmen des TEA-Projekts wird Kontext als Information über den Zustand des Nutzers und des Systems, einschließlich der Umgebung, der Situation und in geringerem Maßen des Ortes definiert [Schmidt et al., 1999a]. In [Schmidt et al., 1999b] wird zum ersten Mal ein ausführliches Modell zum Herausarbeiten von Kontext eingeführt. Die dem Modell zugrunde liegende Definition von Kontext lautet: *„Context describes a situation and the environment, a device or user is in"*. Diese Definition hat jedoch ein Defizit: die Definition des Kontextbegriffs einer Anwendung wird lediglich auf die Definition von Situation und Umgebung verlagert.

In der Literatur folgten noch weitere Definitionen, die allesamt auf die hier aufgeführten Definitionen zurückgeführt werden können. Demnach beschränken wir uns auf die oben aufgelisteten Definitionen und führen darauf eine Analyse durch.

3.3.2 Analyse bisheriger Definitionen

Eine Analyse der aufgeführten Definitionen ergibt, dass Kontext in Zusammenhang mit kontextsensitiven Anwendungen entweder durch eine Auflistung von Beispielen oder durch Angabe von Synonymen oder von Umschreibungen definiert wird.

Die Ansätze, die eine Reihe von Beispielen auflisten, sind statisch und schränken die Menge der möglichen Informationen ein, die als Kontext betrachtet werden können. Die Frage, ob eine bislang nicht betrachtete Art von Information Kontext ist oder nicht, kann hiermit nicht behandelt werden. Es gibt auch ähnliche Ansätze, die Kategorien von Kontext auflisten. Sie erleichtern in einem ersten Schritt eine derartige Entscheidungsfindung, in dem sie feststellen, ob eine Kontextinformation zu einer Kategorie gehört oder nicht. Die Herausforderung dabei liegt aber in der Charakterisierung der aufgelisteten Kategorien. Eine Kontextdefinition basierend auf der Angabe von Synonymen ist sehr vage, und bietet einen breiten Raum an Interpretationsmöglichkeiten. Darüber hinaus wird die Aufgabe lediglich auf die Definition der genannten Synonyme verschoben, was ebenfalls nicht konsequent gemacht wird. Auch die bisherige Ansätze, die den Kontext durch die Angabe von Umschreibungen definieren, sind zu generisch, um die Menge von Informationen einzuschränken, die als Kontext betrachtet werden können.

Ein wichtiger Aspekt bei der Definition von Kontext, unabhängig von dessen Art, ist die Feststellung, ob Kontext *absolut* oder *relativ* ist. Einige der vorgeschlagenen Definitionen, wie z.B. die oft zitierte Definition von Anind Dey [Dey, 2000] betonen den relativen Aspekt von Kontext. Dieser Aspekt ist besonders hervorzuheben, zumal Kontext relativ zu einem Subjekt, einer Anwendung oder einem System zu betrachten ist. Es sollte aber nicht erst zur Laufzeit einer Anwendung bestimmt werden, ob eine gegebene Information als Kontext relevant ist oder nicht. Diese Entscheidung sollte bereits

während der Konzeption der Anwendungen getroffen werden. Dafür ist ein Kontextbegriff erforderlich, welcher einen konstruktiven Ansatz unterstützt.

Ein weiterer Aspekt, der sich aus den vorangegangenen Definitionen ergibt, betrifft die Frage, ob Kontext eine *implizite* oder *explizite* Eingabe an das System ist. Hierzu herrscht keine Einigkeit in der Literatur. Es gibt Definitionen, wie etwa die in [Pascoe, 1998] Gegebene, welche den Kontext einer Anwendung als implizite Information behandeln. Es gibt aber auch Definitionen, wie etwa [Dey, 2000], die zulassen, dass Kontext sowohl implizite als auch explizite Systemeingaben umfasst. Betrachten wir zur Illustration die Identität eines Benutzers: sie kann implizit aus einem personalisierten Gerät entnommen werden oder explizit durch den Benutzer über einen Login-Dialog geliefert werden. In beiden Fällen ist die Identität des Benutzers die gleiche Information und kann als Kontext betrachtet werden, sobald sie für die Anwendung relevant ist.

In jüngsten Arbeiten auf dem Gebiet des Context-Aware Computing [Samulowitz, 2002, Henricksen, 2003, Fahrmair, 2005, Trapp, 2005, Krause, 2006] wurde der Kontextbegriff immer weiterentwickelt und verfeinert. Dies liegt daran, dass die bislang eingeführten Definitionen unzureichend und oft unpräzise für die jeweiligen Zwecke sind. Daran wird sich allerdings aufgrund des relativen Aspekts des Begriffs, selbst mit großartig erweiterten Definitionen, nichts ändern. Vielmehr ist ein konstruktiver Ansatz notwendig, mit dem sich der Kontext einer kontextsensitiven Anwendung systematisch aufbauen lässt. Ein solcher Ansatz soll vor allem die Aspekte von Kontext berücksichtigen, welche immer wieder (kombiniert oder einzeln) vorkommen. So sollen die immer wieder auftauchenden Aspekte von Zeit, Nutzer, Anwendung, Entitäten und Bedingungen in der Nutzungsumgebung sowie Aktivitäten des Nutzers erachtet werden. Darüber hinaus ist bei der Aufstellung eines solchen Ansatzes, den Besonderheiten von Kontext Rechnung zu tragen.

3.3.3 Eigenschaften von Kontext

Kontext als Information hat eine Reihe von Eigenschaften, die in der jüngsten Literatur zu Context Awareness immer wieder genannt werden (siehe hierzu beispielsweise [Henricksen et al., 2002, Strang und Linnhoff-Popien, 2004, Krause, 2006]). In diesem Abschnitt führen wir die für diese Arbeit relevanten Eigenschaften von Kontext auf.

Kontextinformation kann sowohl statisch als auch dynamisch sein

Kontextinformationen können sowohl *statischer* als auch *dynamischer* Natur sein. Statische Kontextinformationen beschreiben *invariante* Aspekte wie etwa den Geburtstag einer Person, die Matrikelnummer eines Studenten oder auch den Ort einer Autobahnbrücke. Die Reaktionen kontextsensitiver Anwendungen hängen vom gemessenen Kontext ab, wobei sich die betrachteten Kontextwerte überwiegend dynamisch über die Zeit hinweg verändern können. Die Persistenz des Kontextes der Nutzung einer Anwendung im Sinne der Verfügbarkeit dynamischer Kontextinformationen kann hoch variabel sein. Beispielsweise kann eine Freundschaft zwischen zwei Personen über Monate oder Jahren hinweg andauern, genau wie der Besitz eines Fahrzeugs, während sich der Lichteinfallwinkel auf einem in Bewegung befindlichen Fahrzeug

von einem Augenblick zu einem anderen ändern kann oder sich der Aufenthaltsort eines Studenten von einer Stunde zur nächsten wechseln kann. Die Persistenz einer Kontextinformation beeinflusst die Art und Weise, wie diese Information gewonnen und verwendet werden soll. Statische Kontextinformationen (also sich selten ändernde Informationen) können zum Beispiel direkt in Form von Nutzereingaben erhalten werden, während dynamische Kontextinformationen (also sich häufig ändernde Informationen) gewöhnlich über indirekte Wege als Eingabe in das System fließen sollten – beispielsweise über entsprechende Sensorik. Dabei spielt die Zeit eine wichtige Rolle: der Kontext enthält eine Komponente Zeit, mit der ihre Dynamik festgestellt wird. Kontextsensitive Anwendungen berücksichtigen häufig mehr als lediglich den aktuell gemessenen Kontext; auch die Kontext-*Historie* kann für die Entscheidungsfindung der Anwendung von Bedeutung sein. Für das kontextsensitive Scheibentönungssystem (siehe 2.5.2) ist beispielsweise für das Treffen einer Adaptionsentscheidung (Scheiben tönen) nicht nur die aktuelle Lichtintensität wichtig, sondern auch einige Sekunden zurückliegende Werte. Darüber hinaus kann in diesem Zusammenhang auch von Bedeutung sein, was der Fahrer in den nächsten 10 Sekunden vorhat (Parken, demnächst abbiegen etc.). Demnach betrifft die Zeitbezogenheit von Kontext, sowohl ihre aktuelle Belegung, als auch ihre vergangenen und zukünftigen Belegungen (*past, present and future of context informations*).

Kontextinformationen können in Wechselbeziehung stehen

Per Definition stehen Kontextinformationen in hohen Wechselbeziehungen miteinander (*interrelated conditions*). Viele Beziehungen zwischen Nutzern, ihren Geräten bzw. Gerätetypen, ihren Kommunikationskanälen oder auch ihren Aktivitäten sind offensichtlich. So sind beispielsweise Zusammenhänge wie der Besitz eines modernen PDA durch einen Geschäftsmann, der Vorlesungsbesuch eines Studenten oder die Präferenz eines Autofahrers mit hoher Sehschwäche, die Scheiben seines Fahrzeugs nur leicht zu tönen, offensichtlich. Es existieren aber auch weniger offensichtliche Beziehungen: so ist zum Beispiel die Beziehung zwischen der Präsenz einer Person in seinem Büro und der aktuellen Aktivität seines Fernsehens oder die Beziehung zwischen Auto fahren und telefonieren nicht notwendigerweise offensichtlich. In einigen Fällen genügt es, nur wenige Kontextinformationen zu erfassen, um ein angemessene Systemreaktion zu erzielen; in anderen Fällen muss ein sehr viel komplexerer Kontext erfasst werden, um aus Nutzersicht angemessen reagieren zu können. Außerdem lässt sich aus bestimmten Kontextinformationen weiterer Kontext ableiten. Aus dem aktuellen Aufenthaltsort eines Mitarbeiters (etwa *„... ist im Raum MI 01.11.054“*) kann beispielsweise abgeleitet werden, was er gerade macht (etwa *„... bespricht etwas“*) oder bei welcher Person er sich gerade befindet (etwa *„... ist bei seinem Vorgesetzten“*), vorausgesetzt, es ist bekannt, dass der Raum MI 01.11.054 nicht der Büroraum des Mitarbeiters ist, sondern der seines Vorgesetzten.

Kontextinformationen können widersprüchlich und unsicher sein

Ein wohl bekanntes Problem für kontextsensitive Anwendungen betrifft die Tatsache, dass Kontextinformationen widersprüchlich sein können. Semantisch gleichwertige Sensoren können unterschiedliche Informationen liefern. Es kann u.U. nicht eindeu-

tig festgestellt werden, welche Sensorwerte korrekt sind und welche nicht. Die Zwiespältigkeit von Kontext ist nur eine der offensichtlichen Qualitätseigenschaften von Kontextinformation. Auch unvollständige Kontextinformationen sind im Zusammenhang mit kontextsensitiven Anwendungen eine große Herausforderung. Inkonsistenz und Unvollständigkeit von Kontextinformationen sind in einem kontrollierbaren Umfang zu halten, und sie sind für ein korrektes Funktionieren der Anwendungen bereits während der Konzeption zu handhaben.

Ein Beispielszenario hierfür ist das Folgende:

- Über den GPS-Sensor des PDA eines Nutzers lässt sich ermitteln, wo sich das Gerät gerade befindet;
- Darüber hinaus hat der Nutzer in seinen Präferenzen für den kontextsensitiven Diebstahlschutz angegeben, dass er sein PDA stets bei sich hat;
- Außerdem, wenn sich der Nutzer an seinem Arbeitsplatzrechner einloggt, lässt sich daraus schließen, dass er sich gerade im Büro befindet;
- Ein Widerspruch entsteht, wenn die ermittelte Position des PDA am Flughafen ist, und sich am Arbeitsplatzrechner (etwa 15 km entfernt) gerade eingeloggt wurde.

Solche Widersprüche können öfter zwischen Kontextinformationen vorkommen, und bedürfen präzise Regeln der Anwendungsdomäne. Für einen kontextsensitiven Diebstahlschutz im obigen Szenario ist festzustellen, welche der Kontextinformationen richtig ist? Soll die Anwendung den Sicherheitsalarm am PDA auflösen? Soll der Arbeitsplatzrechner automatisch gelockt werden? Also, wo befindet sich der Benutzer gerade: am Arbeitsplatz oder am Flughafen? Es kann nämlich sein, dass er sein Gerät verloren hat, oder dass ein dritter sein Passwort kennt, oder etwas Ähnliches. Die einzelnen Kontextinformationen könnten beispielsweise mit Prioritäten versehen werden, so dass im Falle von Widersprüchen, Inkonsistenzen oder Unvollständigkeit, immer noch adäquate Entscheidungen über das Verhalten der Anwendungen getroffen werden.

Bei der Aufstellung eines konstruktiven Ansatzes zur Bestimmung von relevanten Kontextinformationen für die Adaption einer Anwendung ist es wichtig, die hier aufgeführten Eigenschaften von Kontext zu berücksichtigen.

3.4 Unerwünschtes Systemverhalten

Eine Herausforderung bei der Entwicklung (kontextsensitiver) Anwendungen besteht darin, eine möglichst hohe Nutzerzufriedenheit zu gewährleisten. Prinzipiell kommt es immer dann zur Unzufriedenheit des Benutzers eines Systems, wenn sich das System anders verhält, als vom Benutzer erwünscht. Es werden Abweichung zwischen dem Verhalten des Systems und den Erwartungen des Benutzers beobachtet. Wir bezeichnen dieses Phänomen als *unerwünschtes Verhalten* (engl. *Unwanted Behaviour*, kurz *UB*) und verweisen für eine genauere Analyse dieses Problems auf [Fahrmair et al., 2006b]. UB ist ursprünglich als SUB *(Spontaneous Unexpected Behavior)* Phänomen in [Fahrmair, 2005] eingeführt worden. In diesem Abschnitt führen wir eine kurze Beschreibung und Charakterisierung des UB-Phänomens auf, und legen dabei

ein besonderes Augenmerk auf die für das RE relevanten Aspekte des Phänomens.

3.4.1 Erläuternde Beschreibung und Beispiele

Unerwünschtes Verhalten, wie es in diesem Abschnitt aufgeführt wird, ist in erster Linie keine Exklusivität für kontextsensitive Anwendungen. Die Häufigkeit seines Auftretens erhöht sich jedoch besonders mit der Dynamik und Komplexität von Kontext. Mithilfe des Kontextes wird angestrebt, dem Nutzer Interaktionen abzunehmen, und die Anwendungen eine Autonomie durch Kontextadaption zu geben.

Erläuternde Beschreibung

Während klassische Systeme als Hammer und Schraubenzieher von dem Nutzer verwendet werden, und den Nutzer entscheiden lassen, wann reagiert wird, sind kontextsensitive Anwendungen in dieser Hinsicht eigenständige Diener. Sie ergreifen selber die Initiative und bestimmen mithilfe des Kontextes, wann welche Reaktion bzw. welches Verhalten angebracht ist. Dieser Paradigmenwechsel verstärkt allerdings Probleme bzgl. der Benutzerakzeptanz.

In den letzten Jahren haben wir in unserer Forschungsgruppe [C2MCA, 2008] eine Reihe von Prototypen entwickelt, basierend auf den Ideen von Context-Awareness, d.h. ausgehend von den Wünschen und Bedürfnissen des Nutzers das beobachtbare Verhalten der Systemen an die Nutzungsgegebenheiten anzupassen. Unter diesen Prototypen zählen ein In-House Navigations- und Informations-Assistenzsystem [Amann et al., 2004], eine Smart-Home Anwendung [Fahrmair, 2005] und ein kontextsensitiver Terminplaner [Leuxner, 2006, Fahrmair et al., 2008]. Dabei haben wir beobachtet, dass eine Klasse von Systemfehlern existiert, die sich nicht auf Fehler im Design oder der Implementierung der Anwendung zurückführen lassen. Die Systeme verhalten sich zwar wie von den Entwicklern spezifiziert, allerdings kann das Verhalten des Systems im Falle bestimmter Kombinationen von Nutzereingaben und Nutzungskontext aus Sicht dieses Nutzers als Fehlverhalten oder unerwünschtes Verhalten gesehen werden. Das eigentliche Auftreten unerwünschten Verhaltens kann allerdings nicht systematisch mittels bewährten Techniken der Verifikation, wie etwa *Model Checking* oder *Theorem-Beweis*, verhindert werden. Mit anderen Worten: ein unerwünschtes Verhalten kann auch dann auftreten, wenn das betrachtete System konform zu seiner Spezifikation entwickelt wurde.

Ausgehend von dieser einführenden Beschreibung, kann man nicht ausschließen, dass unerwünschtes Verhalten auch bei nicht-kontextsensitiven Anwendungen auftreten kann. Das Phänomen kann sicherlich auch in ganz klassische Anwendungen auftreten. Allerdings weisen kontextsensitive Anwendungen besondere Eigenschaften auf, die die Wahrscheinlichkeit des Auftretens von UB erhöhen:

E1- Kontextsensitive Anwendungen, wie wir sie betrachten, sind *multifunktional* und operieren oft in *heterogenen Umgebungen*.

E2- Kontextsensitive Anwendungen agieren in gewisser Hinsicht selbständig (ohne direkte Eingabe des Nutzers) und agieren in Eigeninitiative (*proaktiv*).

E3- Kontextsensitive Anwendungen basieren auf dem technischen Konzept der *Kon-*

textadaptation zur automatischen Anpassung der Anwendung an ihren Nutzungskontext.

Aufgrund von E1 sind kontextsensitive Anwendungen aus Nutzersicht weniger transparent, und aufgrund von E2 werden Bedienungsfehler im Zusammenhang mit Context-Awareness vernachlässigt bzw. ihrer Existenz keine Wichtigkeit zugeordnet. Mit der Grundidee von Context-Awareness wird grundsätzlich bei dem Nutzer der Eindruck erweckt, dass die Anwendungen immer ein für den Nutzer „optimales“ Verhalten zeigt. Wegen dem technischen Automatismus aus E3, auf dem kontextsensitive Anwendungen basieren, werden die Wirkungen von E1 und E2 noch verstärkt. Der Automatismus ist allerdings zur Erreichung des Ziels erforderlich, die notwendigen Interaktion zu reduzieren und dadurch die Systemnutzung auf Situationen zu erweitern, in denen der Nutzer keine Möglichkeit hat, mit dem System direkt zu interagieren, d.h. Eingaben zu tätigen. Der UB-Problematik muss demnach bei der Entwicklung kontextsensitiver Anwendungen Rechnung getragen werden.

Illustrierende Beispiele

In diesem Abschnitt illustrieren wir das Auftreten unerwünschtes Verhaltens bei Software-Systemen, besonders wenn die Systeme einen gewissen Automatismus besitzen, anhand von zwei kleinen Beispielen.

Microsoft's automatische Rechtschreibprüfung

Ein bekanntes und auch einfaches Beispiel für UB lässt sich bei der automatischen Rechtschreibprüfung von MS Office Word™ registrieren. Eine Funktion in der automatischen Rechtschreibung wandelt immer nach einem Punkt den ersten Buchstabe des nachfolgenden Worts in einem großen Buchstaben um. Diese Funktion kann nicht zwischen einem Punkt am Satzende und einem Punkt nach einer Abkürzung unterscheiden. So wird der erste Buchstabe nach einer Abkürzung, etwa nach *usw.* oder *bzw.*, stets automatisch in einen großen Buchstabe umgewandelt. Diese Funktion kann für den Nutzer unangenehm werden. Wenn diese Funktionsweise nur gelegentlich auftreten würde, könnte man noch damit leben. Allerdings funktioniert die automatische Rechtschreibprüfung stets auf diese Weise, so dass die Nutzer den unerwünscht groß geschriebenen Buchstaben manuell korrigieren müssen.

Navigationssystem

Betrachten wir als weiteres Beispiel für UB ein herkömmliches GPS-basiertes Navigationssystem. Ein solches System hat die primäre Aufgabe, den Nutzer von einem Ort A zu einem Ort B zu führen. Zu Beginn der Routenführung wird von dem System eine Berechnung der Route durchgeführt. Dabei zieht das System neben Start und Ziel auch die Präferenzen des Nutzers in Betracht und führt ihn auf der geplanten Route zum gewünschten Ziel. Wenn sich der Nutzer während der Routenführung verfährt, berechnet das System die Route erneut, und zwar von der aktuellen Position zum Ziel. Stellen wir uns ein Szenario vor, in dem sich eine neulich begonnene Baustelle auf der gewählten Route befindet, welche allerdings noch nicht in das Straßenverzeichnis aufgenommen ist. Das System hat demnach keine Information über die Baustelle und führt den Nutzer zu Selbiger. Der Fahrer dreht daraufhin um, fährt etwa 500m zurück und nimmt einen anderen Weg. Da er sich jedoch in dieser Stadt nicht auskennt, lässt er sich nach einer Weile wieder von dem System führen. Das Navigationssystem be-

rechnet eine neue Route, die zwar anders als die ursprüngliche, jedoch wieder zur obigen Baustelle führt. Um diese Situation zu vermeiden, müsste der Nutzer weit genug zurück fahren, um endlich ans Ziel geführt zu werden. Selbstverständlich existieren heute Navigationssysteme, die Kontextinformationen verwenden, um ihr Verhalten zu optimieren und die sogar dem Nutzer die Möglichkeit bieten, bestimmte Straßen aus der Berechnung der Route auszuschließen. Dies erfordert allerdings zusätzliche Nutzerinteraktionen, was mit dem Ziel von Context-Awareness, Nutzerinteraktionen auf das notwendige Minimum zu reduzieren, im Widerspruch steht.

3.4.2 Charakterisierung: Kriterien und Ursachen

Wir verwenden den Begriff *Unerwünschtes Verhalten (Unwanted Behavior – UB)*, um das Phänomen zu bezeichnen, bei dem das Verhalten eines System, selbst wenn es fehlerfrei konstruiert ist, immer noch von den Erwartungen seines aktuellen Nutzers abweicht. Was sind die eigentlichen Kriterien und Ursachen für das Auftreten von UB? Auf diese Frage gehen wir in diesem Abschnitt ein.

Kriterien

Eine ausführliche Analyse des Phänomens ergibt, dass UB genau dann registriert wird, wenn die folgenden Kriterien allesamt erfüllt sind:

K1- Es existiert eine beobachtbare Divergenz zwischen den Erwartungen des Nutzers und dem Verhalten des Systems

K2- Das System verhält sich korrekt bezüglich seiner Spezifikation

K3- Die Spezifikation entspricht den zum Entwicklungszeitpunkt gesammelten Anforderungen

Die Existenz der beobachtbare Divergenz zwischen Nutzererwartungen und Systemverhalten ist mit der bisherigen Erläuterungen einleuchtend. Sie hängt mit der Unterscheidung zwischen dem mentalen Modell des Benutzers und dem Modell des Systems zusammen, und hat bestimmte Ähnlichkeiten mit den kanonischen Erläuterungen zur Rolle eines mentalen Modells im Prozess der Softwareentwicklung [Norman, 2004]. Dort wird argumentiert, dass es das Ziel des Entwicklers ist, das System so zu entwickeln, dass das mentale Modell des Benutzers über den Einsatz des Systems deckungsgleich mit seinen eigenen Vorstellungen ist. Wenn Diskrepanzen zwischen dem beobachtbaren Verhalten des Systems und den Erwartungen des Benutzers existieren, heißt es nicht zwangsläufig, dass Konstruktionsfehler vorliegen, solange das System sich genauso verhält, wie es spezifiziert wurde. Der Hauptgrund für das Auftreten von UB scheint also ein Defizit in der Gewinnung und Analyse der Anforderungen bzw. der Wünsche und Bedürfnisse der Nutzer zu sein.

Aus den Kriterien K1, K2 und K3 leiten wir ab, dass ein unerwünschtes Verhalten des Systems auftritt, wenn entweder der Benutzer die Fähigkeiten des Systems nicht kennt und Erwartungen entwickelt, die von dem System nicht realisiert werden können, oder wenn sich die zur Entwicklungszeit betrachteten Bedürfnisse des Benutzers geändert haben, etwa aufgrund externer Einflüsse, die aus Systemsicht nicht erkennbar sind. Darüber hinaus kann UB auftreten, wenn das System in Situationen eingesetzt wird,

die zur Entwicklungszeit nicht vorhersehbar waren bzw. nicht berücksichtigt wurden. Das System geht in solchen Fällen von Annahmen (bzgl. Nutzer & Umgebung) aus, die mittlerweile nicht mehr gültig sind, und verhält sich aus Sicht des Benutzers fehlerhaft.

Ursachen

Ereignisse, die unsere Vorstellungen von UB entsprechen, können auf drei Gründe zurück geführt werden (siehe Abbildung 3.3). Diese Gründe sind teilweise allein verantwortlich für das Auftreten von UB, sie können jedoch auch kombiniert vorkommen.

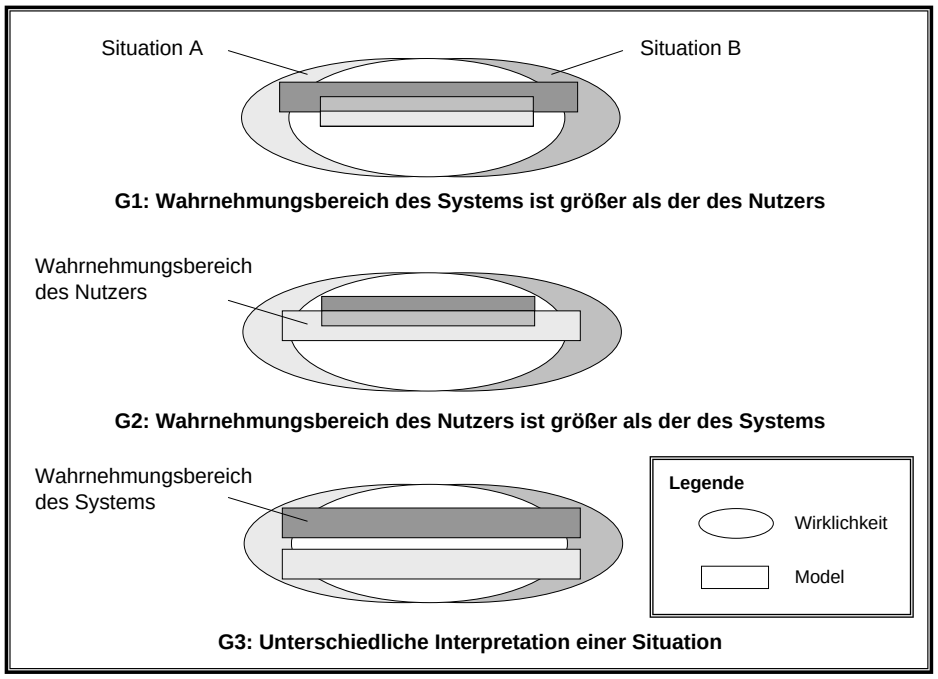

Abbildung 3.3: Unterschied in der Wahrnehmung von Situationen

G1- Der Nutzer kann Situationen voneinander unterscheiden, das System jedoch nicht.

G2- Das System kann Situationen voneinander unterscheiden, der Nutzer jedoch nicht.

G3- Sowohl System als auch Nutzer können Situationen voneinander unterscheiden, allerdings haben sie unterschiedliche Interpretationen der Situationen, d.h. der damit verbundenen Systemreaktionen.

G1 ist typisch für kontextsensitive Anwendungen, die nicht über eine ausreichende Sensorik verfügen. Dieser Fall tritt ein, wenn bestimmte Situationen im RE nicht betrachtet bzw. benötigte Sensoren nicht integriert wurden. Das Modell des Nutzungskontextes (auch Kontextmodell genannt), das dem System zugrunde liegt, ist dann nicht ausreichend präzise, um alle relevanten Situationen zu charakterisieren. Da das Kontextmodell eine Abstraktion der möglichen Nutzungssituationen ist, bei der eben irrelevante Details weggelassen werden, kann es auch sein, dass das Modell zu abstrakt ist und für den Nutzer relevante Details nicht berücksichtigt.

G2 ist das exakte Gegenteil von G1: Wenn ein System Sensoren verwendet, die die Umgebung mit Mitteln beobachten, die jedoch für den Nutzer nicht wahrnehmbar sind, kann es dazu kommen, dass das System neue Situationen identifiziert, die von dem Nutzer nicht identifizierbar sind. Es ist ebenfalls möglich, dass das Kontextmodell zu detailliert ist. Manchmal sind kleine Änderungen in der Umgebung grundsätzlich von dem Nutzer wahrnehmbar, allerdings werden diese Details im Alltagsleben ignoriert. Wenn das System solche Details beobachtet, ist es dann eine Selbstverständlichkeit, dass es Situationen identifiziert, die der Nutzer nicht identifizieren kann.

G3 betrifft die unterschiedlichen Möglichkeiten der Interpretation einer Situation. Der Nutzer und das System können nämlich eine offensichtlich identische Situation unterschiedlich interpretieren. Im Gegensatz zu G1 und G2, wo der Fokus darin lag, zwischen Situationen zu unterscheiden, betont G3 die unterschiedliche Interpretation von Situationen. Dieser Fall kann auftreten, wenn beispielsweise die kulturellen und/oder sozialen Hintergründe der Entwicklers anders als die des Nutzers sind.

Obwohl die Gründe für das Auftreten von UB unterschiedlich sein können, lassen sie sich auf einen gemeinsamen Nenner bringen: das Modell der Wirklichkeit des Nutzers, aus dem sich seine Wünsche und Bedürfnisse ableiten lassen und anhand dessen er seine Erwartungen an das System entwickelt, sind anders als das Modell des Systems. Im nächsten Abschnitt gehen wir auf diese Diskrepanz zwischen beiden Modellen näher ein.

3.4.3 Diskrepanz zwischen Modellen

Diskrepanzen zwischen den Modellen – Modell der Wirklichkeit oder der Erwartungen des Nutzers und eigentlichem System (bzw. Modell des Systems) – sind der eigentliche Grund für das Auftreten unerwünschten Verhaltens in einem technisch für sich fehlerfreien System. Im Zusammenhang mit dieser Feststellung ergeben sich folgende Fragen:

1. Wann entstehen derartige Modelldiskrepanzen?
2. Können sie erkannt und beseitigt oder zumindest vermieden oder kompensiert werden?

Die Fragen zielen nicht auf den Vergleich zweier technischer Modelle, wie in der Software-Verifikation üblich. Im Fokus steht der Vergleich zwischen Modell und Wirklichkeit, also die Validierung eines Modells. Es gilt zu prüfen, ob die zum Entwicklungszeitpunkt erhobenen Annahmen bzgl. dieses Modells korrekt sind und während des gesamten Lebenszyklus des Systems ihre Gültigkeit behalten.

Während das Modell des Systems letztlich in Form einer Implementation vorliegt, lebt das mentale Modell des Nutzers nur in dessen Gedankenwelt; insbesondere ist letzteres meist nicht explizit dokumentiert. Das Modell des Systems steht spätestens nach dessen Implementierung fest und bleibt für gewöhnlich unverändert. Das mentale Modell des Benutzers unterliegt hingegen ständigem Wandel. Folglich müsste dessen Repräsentation beizeiten aktualisiert werden, um Änderungen in der realen Welt zu reflektieren. Ein Beispiel einer solchen Änderung ist die Erfahrung bzgl. der Nutzung eines Systems, welche über die Lebenszeit des Systems hinweg zunimmt. Das mentale Modell des Nutzers selbst spiegelt abhängig von der eingenommenen Perspek-

tive wiederum nur einen Teil der Wirklichkeit wieder. Letztere muss natürlich nicht notwendigerweise mit der von dem Entwickler eingenommenen Perspektive übereinstimmen.

Aus dieser Überlegung folgt, dass Modelldiskrepanzen entweder bereits zum Zeitpunkt der Entwicklung existieren *(versteckte Modelldiskrepanzen)* oder aber während der Laufzeit entstehen können *(spontane bzw. evolutionäre Modelldiskrepanzen)*. Abbildung 3.4 illustriert die beschriebenen Kategorien von Modelldiskrepanzen. Für diese Arbeit sind lediglich Modelldiskrepanzen der ersten Kategorie (Entwicklungszeitpunkt) von Interesse. Diese Modelldiskrepanzen können frühzeitig erkannt und beseitigt werden, adäquate Entwicklungsmethoden und Werkzeuge vorausgesetzt. Die Entwicklung kontextsensitiver Anwendungen erfordert, dass mögliche Nutzungssituationen identifiziert und anschließend ein angepasstes Systemverhalten festgelegt und transparent gehalten wird. Unsere Analyse haben ergeben, dass sich versteckte Modelldiskrepanzen auf zwei möglichen Ursprünge zurückführen lassen.

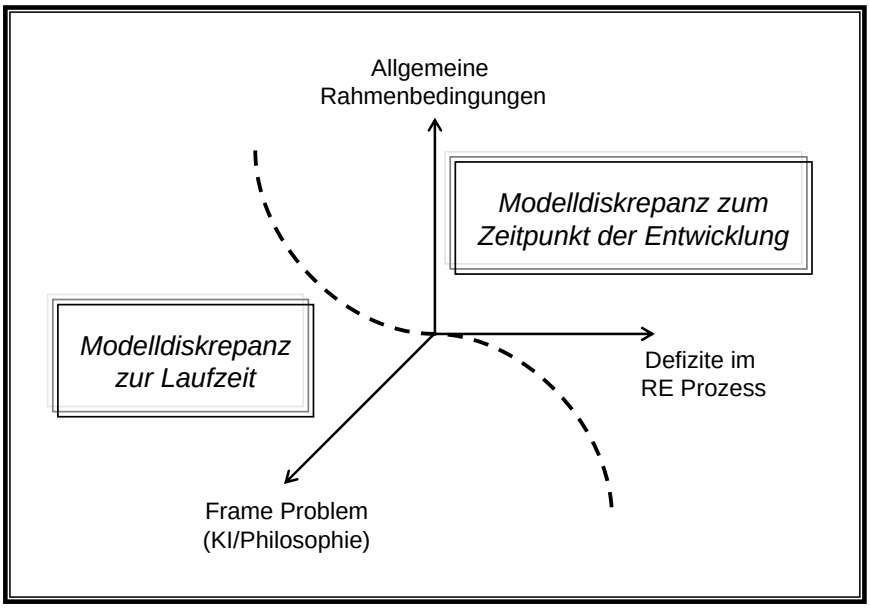

Abbildung 3.4: Mögliche Gründe für Modelldiskrepanzen

Einerseits basieren manche Modelldiskrepanzen zum Entwicklungszeitpunkt auf bewusst getroffenen Entscheidungen (strategischen Überlegungen, Design Constraints usw.). Zum Beispiel könnte während der Entwicklung festgestellt werden, dass unter tausend möglichen Nutzungssituationen zwei Situationen existieren, deren zugehörige Systemanforderungen wirtschaftlich nicht rentabel realisierbar sind und infolge das Endprodukt nur unnötig verteuern würden. Aus Kostengründen wird folglich bewusst auf die Realisierung dieser zwei Sonderfälle verzichtet. Man nimmt damit ganz bewusst in Kauf, dass das Systemverhalten in zwei von tausend Fällen vermutlich nicht den Erwartungen des Nutzers entspricht. 0,2% ist zwar gering, aber wenn man das hochskaliert, kann eine solche Fehlerrate plötzlich ärgerlich sein, besonders in Kombination mit anderen Diensten, wie es öfter der fall ist bei Context-Aware Anwendungen. Bei bewusster Inkaufnahme von unerwartetem Systemverhalten sollte der Nutzer zumindest über das mögliche Auftreten in Kenntnis gesetzt werden, z.B.

in Form von Handbüchern.

Andererseits lassen sich manche Diskrepanzen zum Entwicklungszeitpunkt auf ein defizitäres Requirements Engineering zurückführen. An dieser Stelle reden wir bewusst von Defiziten anstatt von Fehlern, um sie von methodischen Fehlern der Art „*Sammlung und Erfassung von Anforderungen in einem Anforderungsdokument, die ohne Weiteres später mit einer neueren Version überschrieben wird*“ (fehlendes/unzureichendes Rational Management im RE) oder „*Weglassen bestimmter Interviews während der Gewinnung oder Abstimmung von Anforderungen*“ abzugrenzen. Im Fokus unserer Betrachtungen stehen vielmehr Defizite der Art „*Entwickler wusste es zum damaligen Zeitpunkt nicht besser*“. Die Motivation zur Untersuchung solcher Defizite besteht darin, dass derzeitige RE-Ansätze diese Defizite nicht betrachten und deshalb zur Entwicklung komplexer kontextsensitiver Anwendungen ungeeignet scheinen. Sie behandeln weder das für kontextsensitive Anwendungen zentrale Element des *Nutzungskontextes*, noch einen Situationsbegriff im Zusammenhang mit den Anforderungen – im Weiteren auch *situationsbezogene* Anforderungen genannt.

Wie bereits erwähnt, existiert noch eine weitere Ursache für Modelldiskrepanzen, welche im Rahmen dieser Arbeit allerdings nicht weiter betrachtet wird. Der Vollständigkeit halber nennen wir diese Diskrepanzen kurz und verweisen für weitere Details auf [Fahrmair et al., 2006a] und [Fahrmair, 2005]. Die betrachteten Modelldiskrepanzen unterscheiden sich hinsichtlich des Zeitpunktes ihrer *Entstehung*, welcher während der Laufzeit des Systems anzusiedeln ist. Diese Problematik wird seit mehr als zwanzig Jahren in der Künstlichen Intelligenz und der Philosophie diskutiert und ist unter dem Begriff des *Frame Problems*[Dennett, 1984] bekannt. Ob solche zur Laufzeit entstandenen Modelldiskrepanzen (automatisch) erkannt und beseitigt werden können, bleibt eine offene Frage solange das Frame Problem nicht gelöst ist. Mittels *Kalibrierung* können derlei Diskrepanzen allerdings im Nachhinein ohne vollständige Reprogrammierung des Systems manuell kompensiert werden, siehe [Fahrmair, 2005].

3.4.4 Verwandten Begriffe

In der Software- und Systementwicklung existieren eine Reihe von Problemen und Phänomen, die zwar mit dem Phänomen unerwünschten Verhaltens verwandt sind, jedoch eine andere Bedeutung haben. Zu dieser Kategorie zählen *Spezifikations- und Implementierungsfehler von Software*, *Feature Interactions* und *Automation Surprises*. Die genannten Begriffe werden im Folgenden kurz erläutert.

Spezifikations- und Implementierungsfehler von Software

Eine oft diskutierte Frage in Zusammenhang mit UB ist, ob es sich hierbei um Implementierungsfehler der erstellten Software handelt. Diese Frage ist berechtigt, zumal Implementierungsfehler eines Software-Systems genau wie UB zu Systemreaktionen führen können, welche vom Nutzer unerwünscht sind. Das gilt auch für Spezifikationsfehler, die infolge eines unzureichendes Requirements Engineering vorkommen. UB und Spezifikations- bzw. Implementierungsfehler unterscheiden sich allerdings darin, dass Letztere für jeden Nutzer ungeachtet der jeweiligen Nutzungssituation auftreten können. Außerdem lässt sich ein fehlerhaft implementiertes System anhand der

zugrunde liegenden Spezifikation als solches identifizieren. Mittels gängigen Verifikationstechniken lassen sich Implementierungsfehler in der Software zumindest theoretisch[1] noch *während* der Entwicklung aufdecken. Unerwünschtes Verhalten, wie es in dieser Arbeit verwendet wird, kann dagegen *prinzipiell* nicht mittels Verifikation entdeckt werden. Diese Feststellung basiert auf der einfachen Erkenntnis, dass Systeme zwar ohne Fehler implementiert werden können und trotzdem in bestimmten Situationen im Sinne einer ubiquitären Nutzung *unbrauchbar* sind. Im Gegensatz zu Implementierungsfehlern treten im Zusammenhang mit UB-anfälligen Systemen Modelldivergenzen aufgrund diverser *impliziter* Annahmen in der Spezifikation und/oder Implementierung auf.

Feature Interaction und Automation Surprises

Die Begriffe *Feature Interaction* und *Automation Surprise* bezeichnen ähnliche Phänomene, wie sie auch im Zusammenhang mit UB beobachtet werden können.

Das Phänomen der Feature Interaction beinhaltet, dass ein oder mehrere Features weitere Feature(s) bei der Definition des Verhaltens des Gesamtsystems modifizieren oder beeinflussen [Zave, 1993, Zave und Jackson, 2000, Pulvermueller et al., 2002]. Feature in diesem Zusammenhang bezeichnen vom Nutzer erlebbare Funktionen. Bei einem Fahrzeug sind die Funktionen der Kindersicherung und der Fensterheber Beispiele von Feature. In der Literatur wird zwischen positiver und negativer Feature Interaction unterschieden. Negative Feature Interaction sind nicht erwünscht und sollen bei der Modellierung vermieden werden. Negative Feature Interactions sind nicht erwünscht. Sie führen zu inkorrektem Systemverhalten und sollten bei der Entwicklung eines Systems entdeckt und beseitigt werden. Sie werden auch *Unwanted Feature Interactions* genannt und sind nicht mit Spezifikations- bzw. Implementierungsfehler zu verwechseln. Unwanted Feature Interaction ist eng mit dem durch Modelldiskrepanzen verursachten Phänomens des unerwünschten Systemverhaltens verwandt. Allerdings sind die bei UB auftretenden Modelldiskrepanzen nicht nur auf Kombinationen von und Interaktionen zwischen Features beschränkt. Demnach machen Feature Interactions, welche nicht auf Implementierungsfehler zurückzuführen sind, eine Teilmenge von UB aus.

Automation Surprises [Hourizi und Johnson, 2001], auch *Mode Errors* genannt, bezeichnet ebenfalls ein ähnliches Phänomen wie unerwünschtes Verhalten. Genau wie im Fall von UB, werden Automation Surprises registriert, wenn sich das System anders verhält als von dem Nutzer erwartet [Palmer, 1995]. Dabei geht es um Aspekte der Nutzerschnittstelle eines Systems, insbesondere im Bereich der Avionik, und die Transparenz des Systemverhaltens sowie die Interaktion mit dem Benutzer. Darüber hinaus liegt der Fokus bei Automation Surprises lediglich auf dem Entwurf der Nutzerschnittstelle und Fragestellungen bzgl. der Egonomie [Sarter et al., 1997].

[1] Praktisch ist eine Verifikation im Sinne eines vollständigen Tests von komplexen Systemen nicht möglich.

3.5 Schlussfolgerung

Kontextsensitive Anwendungen haben das zentrale Merkmal, dass ihr Verhalten sehr stark von dem Kontext ihrer Nutzung (Nutzungskontext) abhängt. Der Nutzungskontext eines Systems, ist häufig der am wenigsten verstandene und am wenigsten untersuchte Aspekt eines Systems [Cheng und Atlee, 2007].

Gleichzeitig ist der Kontext eine häufige Ursache für das Versagen eines Systems im Betrieb [Jackson, 1995, Hammond et al., 2001]. In vielen Fällen ist ein Versagen des Systems, dessen Ursache vermeintlich in fehlerhaften oder unvollständigen Anforderungen liegt, tatsächlich auf eine unzureichende Analyse des Kontextes zurückzuführen (siehe beispielsweise [Loucopoulos und Karakostas, 1995]). Ein System kann sich kaum erwartungsgemäß verhalten, wenn sein umgebender Kontext im Zuge des Requirements Engineering fehlerhaft bzw. unzureichend berücksichtigt wurde.

Ein weiteres zentrales Merkmal kontextsensitiver Anwendungen betrifft die Kontextadaptivität, wobei die Perspektive des Nutzers eine besondere Rolle spielt. In einer übergreifenden Taxonomie des Kontextes ist der Nutzer der Anwendung explizit zu betrachten. Ob als expliziter Teil des Kontextmodells oder nicht, das Modell des Nutzers ist für die Zwecke des Context-Awareness, insbesondere für die Adaption der Anwendung, unabdingbar.

Ausgehend von den in diesem Kapitel aufgeführten Besonderheiten kontextsensitiver Anwendungen (Kontextbezogenheit und Nichtdeterminismus aus Nutzersicht), den Eigenheiten des Nutzungskontextes (unterschiedliches Verständnis, Zeitbezogenheit, Zusammenhalt, eventuelle Widersprüchlichkeit) und dem Phänomen des unerwünschten Verhaltens (verursacht durch Diskrepanzen zwischen dem eigentlichen Modell des Systems und dem mentalen Modell des Nutzers), stellen sich im Requirements Engineering kontextsensitiver Anwendungen folgende Herausforderungen:

Präventiv gegen Unerwünschtes Systemverhalten im RE vorgehen

Mit der Kalibrierung besteht grundsätzlich eine Möglichkeit, gegen unerwünschtes Systemverhalten vorzugehen. Allerdings kann das System mittels der Kalibrierung erst nach einem ersten Vorkommen von UB nachjustiert werden. Auch, wird vorausgesetzt, dass die notwendige Expertise zum Nachjustieren bzw. Kalibrieren der Adaptionslogik vorhanden ist. Die Kalibrierung hilft also nicht, um UB zu vermeiden bevor es überhaupt auftritt. Sie ist kein echt präventives Mittel gegen UB. Ein guter Maßstab sollte es sein, UB noch vor seinem ersten Auftreten zu vermeiden. Wir haben festgestellt, dass es Ursachen von UB gibt, die bereits während der Entwicklung des Systems, insbesondere im Requirements Engineering, vermeidet werden können. Diese Ursachen betreffen:

- *Unzureichendes Kontextmodell*
  Mit einem unzureichenden Kontextmodell werden die Nutzungssituationen durch das System nicht ausreichend abgedeckt. Dadurch werden die Nutzer gewisse Situationsänderungen feststellen und entsprechende Adaptionen erwarten, welche jedoch aufgrund unzureichender Kontextinformationen im Kontextmodell nicht stattfinden werden.

- *Unterschiedliche Nutzererwartungen*
  Unterschiedliche Nutzer haben unterschiedliche Erwartungen und unterschiedliche Bedürfnisse und Fähigkeiten. Mit der Erstellung von Nutzermodellen im Rahmen der Kontextmodellierung im RE werden solche relevanten Unterschiede festgestellt und nutzergerechte Adaptionsentscheidungen herbeigeführt.
- *Falsche Regeln für die Adaptionsentscheidung*
  Selbst mit einem ausreichenden Kontextmodell und einer ausführlichen Berücksichtigung der unterschiedlichen Nutzererwartungen, kann es zu unerwünschtem Systemverhalten kommen, welche jedoch durch geeignete Maßnahmen im RE vermieden werden können. Die den Adaptionsentscheidungen zugrunde liegenden Regeln sind sorgfältig zu definieren.

Zusätzlich muss im Phase des Systementwurfs vermieden werden, dass *falsche Kontextmodellinhalte* produziert werden. Es müssen zuverlässige Kontextinformationen erfasst werden. Die Zuverlässigkeit der Kontextinformationen ist mit adäquaten Maßnahmen im Systemdesign (z.B. durch Ausstattung des System mit einer zuverlässigen Sensorik) zu garantieren. Heutzutage werden Requirements Engineering und Systementwurf nicht mehr scharf getrennt von einander gehalten (siehe z.B. [Nuseibeh, 2001, Broy, 2006b, Broy, 2006a, Geisberger und Schätz, 2007] und [Pohl und Sikora, 2007]). Dementsprechend muss ermöglicht werden, dass Feedbacks aus dem Systementwurf, beispielsweise bezogen auf die Zuverlässigkeit der Kontextinformationen, iterativ im RE eingearbeitet werden.

Nutzer wissen nicht, was sie wollen; auch nicht, was das System kann

Diese im RE wohl bekannte Tatsache trifft auf kontextsensitive Anwendungen ganz besonders zu, zumal Nutzer meist nur vage Vorstellungen einer kontextsensitiven Anwendung haben, mit denen sie noch keine Erfahrung sammeln konnten. Context-Awareness soll die Interaktion für den Nutzer komfortabler gestalten, wobei Nutzer ihre Wünsche bzgl. der Nutzung meist nur schwer einschätzen und artikulieren können.

Es existiert keine feste Form von Interaktion

Um Systemanforderungen zu ermitteln bieten sich typische Anwendungsfälle zur leichteren Identifikation an. Im Requirements Engineering kontextsensitiver Anwendungen scheint diese Aufgabe noch komplizierter aufgrund ihrer Kontextbezogenheit. Diese Gesichtpunkte sollten im RE kontextsensitiver Anwendungen explizit behandelt werden.

Der Kontext ist integriert mit den Anforderungen zu entwickeln

Kontextsensitive Anwendungen verhalten sich abhängig von dem Kontext, in dem sie genutzt werden. Wechselnde Nutzer, Aufgaben, Umgebungsbedingungen und die gegenseitigen Beziehungen zwischen diesen Faktoren sind nicht getrennt von den Anforderungen zu ermitteln, zu analysieren und zu dokumentieren. Es existiert noch keine Methodik zur integrierten Entwicklung des Kontextes mit den Anforderungen. Die

einzelnen Faktoren des Kontextes sind bei der Entscheidung, welche Anwendungsfunktionen gerade angebracht sind, zu berücksichtigen. Eine Nicht-Berücksichtigung dieser Faktoren im Requirements Engineering würde dazu führen, dass Adaptionsentscheidungen getroffen werden, welche die Nützlichkeit der Anwendung negativ beeinflussen. Eine integrierte Behandlung (insbesondere die Ermittlung, die Analyse und die Dokumentation) des Kontextes und der Anforderungen ist vorzunehmen.

Nicht jede Adaption ist angebracht

Wenn sich der Nutzungskontext relativ schnell ändert, könnte sich die Anwendung theoretisch auch relativ schnell adaptieren, je nachdem wie fein granular die Adaptionslogik formuliert ist *(Context Triggered Adaptation)*. Die Folge wäre ein ständiges Hin- und Herwechseln zwischen den Funktionalitäten des Systems. Solche Systemreaktionen sind für den Nutzer unter Umständen schwer nachzuvollziehen. Um die Systemreaktion nachvollziehbarer zu gestalten, sollte aus Sicht des Nutzers eine gewisse Transparenz über alle relevanten Teile der für die Adaption in Betracht gezogenen Kontexte herrschen. Abgesehen von der Nutzerakzeptanz können sich häufige Adaptionen negativ auf die Performanz des Systems auswirken, da ggf. viel Rechenzeit für die Anpassungen und Rekonfigurationen des Systems verbraucht wird.

Die Berücksichtigung der in diesem Kapitel aufgeführten Herausforderungen ist äußerst wichtig für die Entwicklung kontextsensitiver Anwendungen für realitätsnahe Anwendungsszenarien. Bisherige Arbeiten im Bereich des Context-Aware Computing haben diese Herausforderungen allzu oft außer Acht gelassen und ihren Fokus lediglich auf die technische Realisierbarkeit der Anwendungen gelegt.

In der Literatur zum Thema Requirements Engineering werden kontextsensitive Anwendungen bislang nicht explizit behandelt. Um diese Lücke zu füllen, müssen geeignete Entwicklungsmethoden erarbeitet werden, welche sowohl Nutzer als auch Nutzungsgegebenheiten in den Mittelpunkt stellen. Die Untersuchung der Nutzerwünsche und der beabsichtigten Nutzungsumgebung sind dabei expliziter Bestandteil der Methodik. Derartige Untersuchungen werden integriert mit der Ermittlung, der Analyse und der Dokumentation der Anforderungen durchgeführt.

Kapitel 4

Framework zur Modellierung des Nutzungskontextes

Inhaltsangabe

The notion of context is very broad and incorporates lots of information, not just about the current location, but also about the current activity ...

[van Laerhoven und Aidoo, 2001]

4.1 Einleitung

Die Realisierung der Adaptivität in Zusammenhang mit kontextsensitiven Anwendungen erfordert eine systematische Berücksichtigung des Nutzungskontextes, anhand dessen die Anwendung ihr beobachtbares Verhalten automatisch zur Laufzeit anpasst. Für die automatische Anpassung benötigt die Logik der Adaption ein Modell des Kontextes. Basierend auf diesem Modell werden die unterschiedlichen Situationen der Nutzung der Anwendung mittels relevanten und über Sensorik wahrnehmbaren Informationen festgestellt. Abbildung 4.1 illustriert diesen Sachverhalt und hebt die Bedeutung eines Kontextmodells in einer kontextsensitiven Anwendung hervor[1].

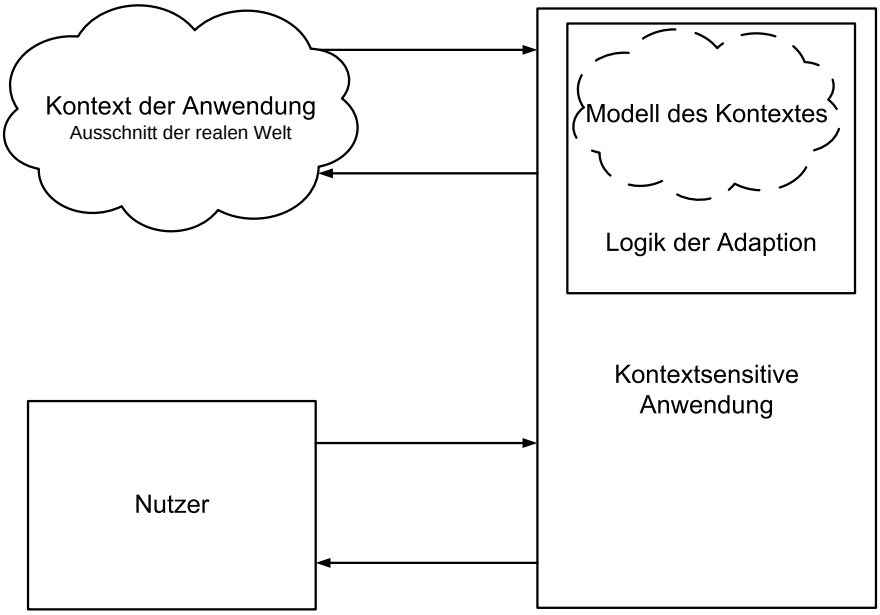

Abbildung 4.1: Kontext als Grundlage der Adaptionen

Um die im Kapitel 3 aufgeführten Herausforderungen kontextsensitiver Anwendungen zu meistern, ist aus Sicht dieser Arbeit zunächst der aspektreiche Begriff von Kontext sorgfältig – am besten systematisch – auf einer konzeptuellen Ebene optimal zu erfassen (erheben, analysieren und dokumentieren). Damit wird ermöglicht, alle für die Adaption der Anwendung notwendigen Kontextinformationen in einer für alle Beteiligte verständlichen Form festzuhalten. Eine Voraussetzung für die systematische Erfassung des Kontextes ist das Erstellen geeigneter Modelle, welche die unterschiedlichen Aspekte des Konzeptes abbilden. Dabei stellt sich die Frage, welche Information

[1] Das in der Abbildung 4.1 dargestellte Modell des Nutzungskontextes einer kontextsensitiven Anwendung suggeriert, dass der Nutzer nicht Teil des Kontextes ist bzw. dass ein Nutzermodell keine Rolle in kontextsensitiven Anwendungen spielt. Dies trifft allerdings nicht zu. Denn wie es später in diesem Kapitel aufgeführt wird, behandeln wir das Nutzermodell sogar als zentralen Bestandteil eines integrierten Kontextmodells.

die Situation der Nutzung einer Anwendung kennzeichnet, bzw. welche Informationen Teil des Kontextes sind und welche nicht. Für die Zwecke der systematischen, und von der Nutzung der Anwendung getriebenen, Modellierung des Kontextes auf einer konzeptuellen Ebene wie im Requirements Engineering ist es notwendig, ein grundlegendes Verständnis dieses Konzeptes in Form einer Taxonomie zu schaffen. Eine solche Taxonomie soll die unterschiedlichen Aspekte des Begriffes widerspiegeln.

Für die Entwicklung eines umfassenden Kontextmodells sind Modellierungstechniken erforderlich, welche die gesamten Kontextinformationen beschreiben, inklusive der Abhängigkeiten zwischen ihren einzelnen Bestandteilen. Angesichts der Vielfältigkeit des Nutzungskontextes erweist sich die Eignung einer einzigen Modellierungstechnik in Hinblick auf die Ermittlung und Analyse aller für die Adaption einer Anwendung relevanten Informationen als nicht effektiv [Strang und Linnhoff-Popien, 2004, Henricksen et al., 2004, Baldauf et al., 2007]. Eine Zusammenstellung verschiedener Modellierungstechniken ist vielmehr angebrachter.

In diesem Kapitel wird auf die aufgeführten Fragestellungen eingegangen:

1. Im Abschnitt 4.2 wird ein grundlegendes Modell des Kontextes der Nutzung einer Anwendung aufgestellt, wie er im Requirements Engineering kontextsensitiver Anwendungen zu verstehen ist. Dabei sind gängige Definitionen und Modelle aus dem Bereich der Kontextmodellierung in Ubiquitous und Context-Aware Computing, wie sie im Abschnitt 3.3 aufgeführt wurden, in Betracht gezogen. Die zentralen Aspekte bzw. Dimensionen des Kontextes der Nutzung einer Anwendung – Nutzer, Aktivität, Einsatzumgebung, ihre Wechselwirkungen unter einander und ihre Veränderungen über die Zeit – werden beschrieben.
2. Im Abschnitt 4.3 wird ein konzeptioneller Rahmen *(conceptual framework)* zur Modellierung von Kontextinformationen im RE kontextsensitiver Anwendungen aufgeführt. Eine Überführung der einzelnen Aspekte des Kontextes in entsprechende Modelle (Teilmodelle eines integrierten Kontextmodells) erfolgt in diesem Framework. Ein Überblick des integrierten Modells wird in diesem Abschnitt gegebenen. Außerdem werden Kriterien vorgestellt, anhand dessen die Eignung der einzelnen Modellierungstechniken festgestellt wird.
3. In den Abschnitten 4.4, 4.5, 4.6, 4.7 und 4.8 erfolgen ausführliche Beschreibungen der einzelnen Modelle. Dabei werden die jeweiligen Informationsgehalte der Modelle beschrieben und geeigneten Modellierungstechniken diskutiert. Die Bewertung der Eignung der Techniken basiert auf die zuvor definierten Kriterien.

4.2 Zentrale Aspekte des Kontextes

„There is more to context than location“ [Schilit et al., 1994].

Im Umfeld von Software Anwendungen, die ihren Nutzungskontext berücksichtigen, wird des Öfteren der Kontextbegriff ausschließlich mit dem Einsatzort *(location)* gleich gesetzt. Die meisten ersten kontextabhängige Anwendungen sind ausschließlich ortsbezogene Anwendung – *location-based services, auch location-aware computing* – (siehe beispielsweise [Abowd, 1999], [Pham, 2001] und [Timpf, 2008]). Verbreitete Beispiele solcher Anwendungen sind Empfehlungsdienste, die ihre Emp-

fehlungen davon abhängen, wo sich der Nutzer gerade aufhält oder Thermometer-Anwendungen, die die Angaben der Temperaturwerte abhängig von dem Einsatzort (z.B. USA, Europa) automatisch in Grad Fahrenheit (°F) oder Grad Celsius (°C) machen. Seit Beginn der Forschungsarbeiten auf dem Gebiet des Context-Aware Computing wurde jedoch immer wieder bekräftigt, dass der Kontextbegriff mehr als den Einsatzort einer Anwendung bezeichnet [Schilit et al., 1994], [Schmidt et al., 1999b] und [van Laerhoven und Aidoo, 2001].

4.2.1 Grundlegende Taxonomie des Kontextes

Der Kontext einer Anwendung hat eine Reihe unterschiedlicher Aspekte, deren Relevanz für die Adaption der Anwendung nicht vernachlässigt werden soll. Für die Zwecke der Modellierung des Kontextes ist eine Taxonomie dieses Begriffs erforderlich, auf deren Basis das Modell erstellt werden kann. In diesem Abschnitt führen wir die Taxonomie auf, welche eine essentielle Grundlage der Modellierung von Kontext im Rahmen dieser Arbeit bildet.

Im Kapitel 3.3.1 haben wir unterschiedliche Definitionen des Kontextes diskutiert, die sehr schwer kongruent sind. Bei manchen Definitionen werden Aspekte des Kontextes weggelassen, welche jedoch in anderen Definitionen von wichtiger Bedeutung sind. Darüber hinaus sind viele dieser Definitionen nicht mit der Zielsetzung entstanden, eine systematische Erstellung übergreifender Modelle des Kontextes zu ermöglichen. Vielmehr sind sie aus der Überlegung entstanden, ausführbare Modelle zu erstellen, welche direkt in prototypischen Realisierung kontextsensitiver Anwendungen fest eingebunden sind und lediglich dort zum Einsatz kommen. Je mehr Kontextinformationen die Anwendungen verwenden sollen, desto umfangreicher werden die zugrunde liegenden Kontextmodelle. Eine Systematik zum Herausarbeiten von in sich abgeschlossenen Teilen eines Modells, welche dann als Bestandteil des Kontextes einer Anwendung fungieren, war kein Motiv bei der Aufstellung der Definitionen, und dementsprechend auch bei der Konzeption der Kontextmodelle. Eine konstruktive Anleitung zum Feststellen, welche Information Teil des Kontextes ist und welche nicht, ist ebenfalls nicht vorhanden.

Ein übergreifendes, und verständliches, Modell des Kontextes soll nicht nur eine prototypische Entwicklung bezwecken, sondern und vor allem auch die Entwicklung kontextsensitiver Anwendungen für realitätsnahe Nutzungsszenarien unterstützen. Ein solches Modell soll grundlegende Aspekte des Kontextes enthalten. Es soll so deskriptiv gestaltet werden, dass ein Zuschneiden auf anwendungsspezifische Forderungen weitgehend intuitiv erfolgt. Ferner ist eine Systematik notwendig, welche es dem Entwickler ermöglicht, die Erstellung der Modelle konstruktiv zu gestalten, beispielsweise bei der Entscheidung, welche Elemente in dem Modell aufzunehmen sind und welche nicht. Allerdings ist eine solche Entscheidung nicht durch den Entwickler allein zu treffen, sondern in Zusammenarbeit mit relevanten Stakeholder wie Nutzern und Domänenexperten. Auf den konstruktiven Ansatz gehen wir in Kapitel 5 ein, und führen dort eine integrierte Kontext- und Szenario-Analyse zur Erarbeitung der einzelnen Bestandteile eines Kontextmodells ein. In diesem Kapitel beschäftigen wir uns zunächst lediglich mit dem deskriptiven Ansatz.

[Tarasewich, 2003] macht einen ersten Schritt in diese Richtung und stellt eine Ta-

xonomie auf, welche die Erstellung eines umfassenden Kontextmodells ermöglichen soll. Die in dieser Arbeit verwendete Taxonomie als Grundlage der Kontextmodellierung im RE kontextsensitiver Anwendungen lehnt sich an die in [Tarasewich, 2003] aufgeführte Taxonomie. Dort wird argumentiert, dass die Komplexität des Kontextes der Nutzung einer Anwendung dadurch reduziert werden kann, indem er aus unterschiedlichen Perspektiven analysiert wird. In der aufgestellten Taxonomie sind die in dieser Arbeit betrachteten zentralen Aspekte des Kontextes genannt, nämlich die beteiligten Nutzer, die durchgeführten Aktivitäten und die Umgebung der Nutzung der Anwendung (*participants, activities and environments*). In dieser Taxonomie lassen sich die im Abschnitt 3.3 aufgeführten Definitionen des Kontextes einordnen, insbesondere die Definitionen von [Schilit et al., 1994], [Schmidt et al., 1999b], [Schmidt et al., 1999a], [Abowd und Mynatt, 2000], [Dey, 2000], [Jameson, 2001] und [Henricksen et al., 2002]. Auch der Zeitbegriff im Sinne von Zeitverlauf (Vergangenheit, Gegenwart und Zukunft), wie er beispielsweise in [Chen und Kotz, 2000] und [Dey und Abowd, 2000] erwähnt wurde, spielt in dieser Taxonomie eine wichtige Rolle. Eine ausführliche Beschreibung der Aspekte ist allerdings nicht gegeben.

Ein integriertes Modell des Kontextes in kontextsensitiven Anwendungen sollte die aufgeführten unterschiedlichen Aspekte berücksichtigen. Ein solches integriertes Modell soll also neben der Einsatzumgebung, wie in [Hong et al., 2005] beschrieben, auch den Aktivitäten, die in der Einsatzumgebung durchgeführt werden, sowie den Nutzern, die diese Aktivitäten durchführen, umfassen. Der Nutzer-Aspekt sollte sogar in Vordergrund gestellt werden. Abbildung 4.2 illustriert diese Taxonomie mit den benannten Aspekten.

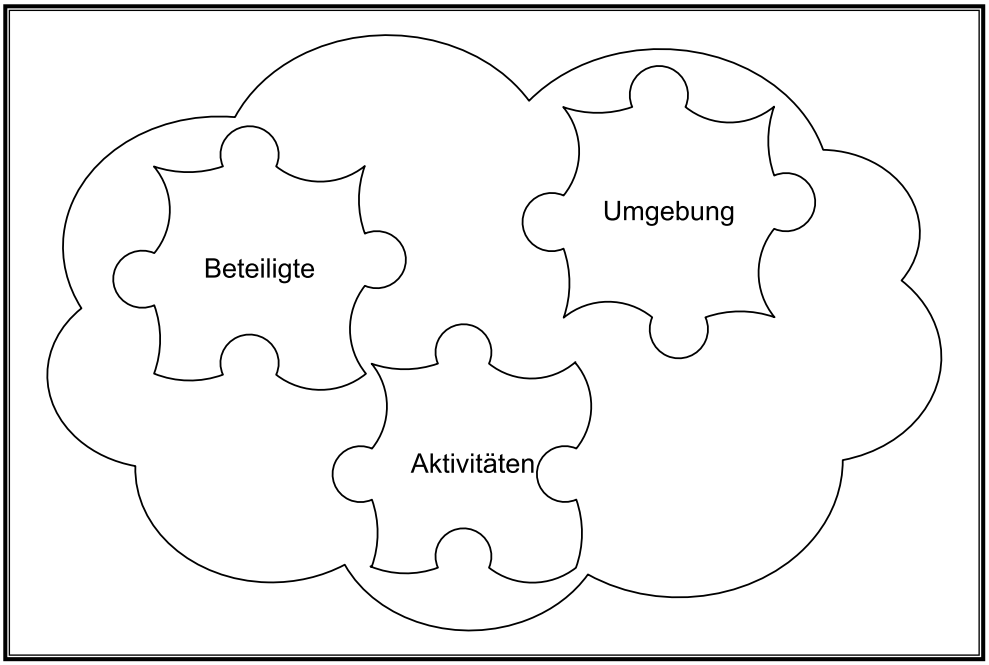

Abbildung 4.2: Zentrale Aspekte des Kontextes

Die Erstellung einer solchen Taxonomie des Kontextes lässt sich aus der Nutzungssituation (kurz Situation) motivieren. Damit angemessene Adaptionen vorgenommen werden können, werden charakteristische Informationen über die einzelnen Situatio-

nen der Nutzung einer Anwendung benötigt. Die Nutzungssituation, wie sie beispielsweise in der Kontextdefinition von [Rodden et al., 1998, Schmidt et al., 1999a] oder auch [Dey, 2000] (siehe Abschnitt 3.3) erwähnt wurde, ist also näher zu beschreiben. Dabei betonen wir die drei zentralen Aspekte von Kontext und ihre Wechselwirkungen unter einander. In Zusammenhang mit kontextsensitiven Anwendungen kennzeichnet sich eine Situation zusätzlich dadurch, dass sie sich auf eine bestimmte Zeit t bezieht und dabei die unterschiedlichen Aspekte des Kontextes in ihrer Belegung zu der Zeit t miteinander in Beziehung setzt. Beispiel einer Situation zum Zeitpunkt $t = 04.11.2007um08 : 24Uhr$ ist:

*User **A302568TG** at Date **2007/11/4/0824** is travelling by Transportation Mean **ICE 518** from **Munich** to **Stuttgart**.*

In einer Situation stehen also die unterschiedlichen Kontextinformationen miteinander in Wechselwirkungen. Sie haben jeweils einen bestimmten Zustand bzw. Wert. Ihre Veränderungen über die Zeit wird durch eine Auswertung des Kontextmodells festgestellt. Eine Beschreibung der einzelnen Aspekte, inklusive ihren Wechselwirkungen unter einander für die Charakterisierung der Situation einer Anwendung, erfolgt im folgenden Abschnitt. Abbildung 4.3 illustriert diesen Situationsbegriff.

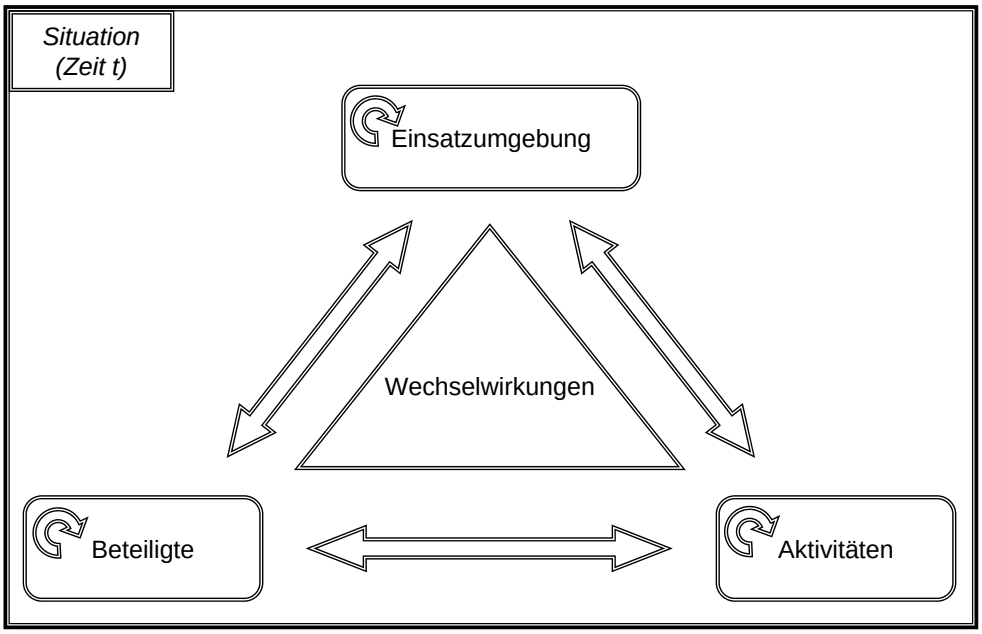

Abbildung 4.3: Situation der Nutzung einer Anwendung

4.2.2 Die einzelnen Bestandteile und die Wechselwirkungen

In diesem Abschnitt beschreiben wir die einzelnen Aspekte des Kontextes (und letztendlich der Situation der Nutzung) einer Anwendung – *Beteiligte, Aktivität und Einsatzumgebung* – und gehen anschließend auf die Wechselbeziehungen zwischen ihnen ein.

Beteiligte

Der Aspekt „Beteiligte“ steht für die Nutzer der Anwendung, und wird daher auch Nutzer-Aspekt genannt. Bei der Modellierung dieses Aspekts wird ermittelt, was die Anwendung über die Beteiligten wissen soll. Ziel der Modellierung ist die Erfassung bzw. Beschreibung ihrer Rolle und Festlegung, was für sie charakteristisch ist. Beispielsweise sind die Beteiligten bei unserem kontextsensitiven Terminplaner *Studenten, Mitarbeiter, Professoren und auch Geschäftsleute*.

Es ist schwer vorstellbar, dass der Nutzer-Aspekt in einem Kontextmodell vernachlässigt werden darf. Eine Entwicklung kontextsensitiver Anwendungen für realitätsnahe Szenarien ist ohne die Berücksichtigung dieses Aspekts kaum denkbar, da das System andernfalls die Interessen des Nutzers nicht kennt und nicht unterstützen kann. Die Bedürfnisse, Wünsche und Fähigkeiten der Nutzer werden stärker in den Vordergrund gestellt. Der Kontext-Aspekt „Beteiligte“ hat somit eine Schlüsselrolle bei der Modellierung von Kontext inne.

Die Beteiligten der Nutzung einer Anwendung beschränken sich nicht auf den eigentlichen Nutzer. Weitere Personen in der Nutzungsumgebung können ebenfalls von Interesse sein, so bald sie mit dem Nutzer in einer Beziehung stehen, die für die Adaption der Anwendung relevant ist. Für die automatische Ermittlung des Status eines Nutzers des kontextsensitiven Terminplaners ist es beispielsweise wichtig zu wissen, welche Beziehungen zwischen diesem Nutzer und weiteren Personen in einer Nutzungssituation des Systems besteht. Zum Beispiel ist der Nutzer der Vorgesetzte der anderen Personen in der Nutzungssituation, oder ist er ein Mitarbeiter, der sich gerade in einer Besprechung mit seinem Vorgesetzten befindet. Diese Informationen sind beispielsweise für die Entscheidung wichtig, welche Art der Benachrichtigung gerade angebracht ist und welche nicht. Bei der Modellierung dieses Aspekts sind auch psychologische, soziale und kulturelle Hintergründe der Beteiligte mit einzubeziehen.

Aktivitäten

Neben dem Nutzer-Aspekt ist in der Kontextmodellierung der Aspekt „Aktivität“ von Interesse. Mit diesem Aspekt wird beschrieben, wofür der Nutzer die Anwendung nutzen möchte und welche Aktivitäten er hierzu durchführt. Die Modellierung des Kontextaspekts „Aktivitäten“ besteht also darin, ein Modell von den Zielen zu erstellen, welche die benannten Beteiligte mit der Anwendung erreichen wollen und welche Aufgaben sie beim Umgang mit der Anwendung durchführen.

Ziele und Aufgaben sind sehr eng miteinander verwandt [Horvitz et al., 1998, Chen et al., 2002]. Je nachdem welche Ziele ein Nutzer gerade erreichen möchte, führt er bestimmte Aufgaben durch [Schmidt-Belz, 2005, Baldes et al., 2005]. Wenn ich zum Beispiel das Ziel verfolge ein Tisch zu decken, denke ich mir aus, welche Geschirr ich dafür benötige. Mit Context Awareness wird angestrebt, den Nutzer bei der Durchführung seiner Aufgaben zu unterstützen, indem abhängig von Kontextinformationen beispielsweise Vorschläge gemacht werden oder Aufgaben übernommen werden. Daher müssen kontextsensitive Anwendungen ein Modell aktueller Ziele und Aufgaben der Nutzer enthalten. Ziel- und Aufgabenmodellierung bilden einen unabdingbaren Teil der Kontextmodellierung.

Einsatzumgebung

Ein weiterer Aspekt des Kontextes betrifft die Umgebung, in der die Anwendung eingesetzt wird. Mit dem Aspekt „Einsatzumgebung“ werden die Umgebungsbedingungen, unter denen der Nutzer die Anwendung verwendet, um seine Ziele zu erreichen bzw. seine Aufgaben durchzuführen, beschrieben.

Bei der Modellierung dieses Aspektes befassen wir uns mit der Fragestellung, in welcher Umgebung und unter welchen Umweltbedingungen, die Beteiligten ihre Aktivitäten durchführen. Wir beschreiben dann alle Gegebenheiten der Einsatzumgebung, die für die Nutzung und die Adaption der Anwendung von Bedeutung sind.

Die Einsatzumgebung ist der Aspekt des Kontextes, der am selbst verständlichsten scheint. So wird sie in fast allen Definitionen des Kontextes (siehe Abschnitt 3.3) betrachtet, zumindest in Form von Ortsinformationen. Allerdings durch die Tatsache, dass dieser Aspekt unterschiedliche Arten von Informationen enthält, kann seine Erarbeitung komplex werden. Eine sorgfältige Betrachtung der Einsatzumgebung ist daher unabdingbar.

Die Einsatzumgebung kann Informationen über den Einsatzort (*location of use*) sowie weitere relevante Zusatzinformationen enthalten. Dazu zählen sowohl augenfällige Objekte in der Einsatzumgebung wie technische Geräte, Straßen und Gebäude, als auch physikalische Größen, wie Wärme, Lärm/Lautstärke und Helligkeit.

Wechselwirkungen

In einer Nutzungssituation besteht eine gewisse Beziehungen zwischen den aufgeführten zentralen Aspekten (siehe Abbildung 4.3). Die Belegungen (Wert bzw. Zustand) eines Aspektes können die Belegungen innerhalb eines anderen Aspektes beeinflussen, und vice versa (**Inter-Aspekt Wechselwirkung**). Eine Konstellation wie etwa

*Der Vorgesetzte des Mitarbeiters **A302568TG** befindet sich im Raum MI 01.11.018 von 10:20 Uhr bis 10:40 Uhr, und der Mitarbeiter **A302568TG** führt private Telefonate im Raum MI 01.11.018 von 10:20 Uhr bis 10:40 Uhr durch*

oder

*Nutzer **A302568TG** ist taub, und Nutzer **A302568TG** verfolgt Nachrichten im **Radio***

ist unwahrscheinlich bzw. nicht sinnvoll innerhalb einer und derselben Situation. Sie soll daher vermieden werden, damit das System für den Nutzer nützlich wird. An diesen relativ kleinen Beispielen sehen wir, dass die aktuellen Aktivitäten eines Nutzers stark von der Umgebung, in der sie durchgeführt werden, und auch von seinen Charakteristiken, Präferenzen oder von den sozialen Bedingungen beeinflusst werden. Diese Art von Wechselwirkungen werden insbesondere ersichtlich, wenn Kontextinformationen der drei unterschiedlichen Aspekte in ihren Wechselwirkungen betrachtet werden, bzw. miteinander integriert werden.

Neben den Wechselwirkungen zwischen den Aspekten, können auch Wechselwirkungen innerhalb der einzelnen Aspekte existieren (**Intra-Aspekt Wechselwirkung**). Beispielsweise hängen die Informationen „Der Straßenabschnitt $[A, B]$ ist eine Allee“ und „Auf dem Straßenabschnitt $[A, B]$ herrschen wechselhafte Lichtverhältnisse“ von ein-

ander ab. Oder auch der aktuelle Wert der Dunkelheit (physikalische Größe) in der Einsatzumgebung hängt von der Kontextinformation Lokalität ab. Der Grad der Dunkelheit ist nämlich von dem betrachteten Ort abhängig. Beispielsweise ist um 23 Uhr auf einer Hauptstraße in der Innenstadt grundsätzlich heller als auf der Autobahn. Derartige Abhängigkeiten werden bereits während der Ermittlung und Analyse der einzelnen Aspekte, noch vor ihrer Integration, identifiziert.

Beziehung zwischen Situation und Kontext

Mit der Erweiterung des Situationsbegriffs aus Abbildung 4.3 um die Betrachtung der Veränderungen über die Zeit, erhalten wir ein umfassendes grundlegendes Modell des Kontextes der Nutzung einer Anwendung. Abbildung 4.4 illustriert dieses grundlegende Modell.

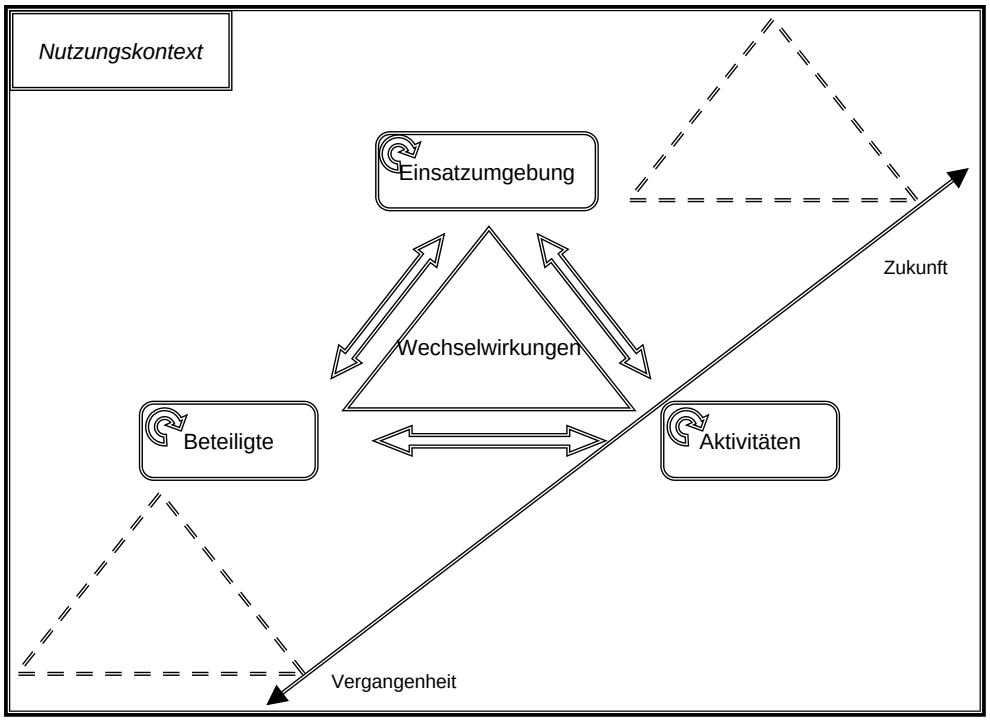

Abbildung 4.4: Kontext als Modell einer Situation mit ihren Änderungen über die Zeit

Bei diesem Modell gehen wir davon aus, dass jeder Aspekt des Kontextes aktiv bzw. inaktiv in einer bestimmten Situation sein kann. Obwohl wir mit dem aufgeführten Modell versuchen, alle mögliche Kontextinformationen zu berücksichtigen, die für die Adaption einer kontextsensitiven Anwendung von Bedeutung sein können, ist uns bewusst, dass nicht alle diese Informationen für die Modellierung der Situationen einer Anwendung betrachtet werden müssen. Es kann nämlich vorkommen, dass lediglich Informationen aus Teilen der drei Aspekte zur Charakterisierung einer Situation benötigt werden. Für den kontextsensitiven Terminplaner ist es z.B. ausreichend zur Charakterisierung der Situation für die Zwecke der Adaption bezüglich der Art der Benachrichtigung zu wissen, in welcher Umgebung sich der Nutzer gerade befindet. „Der Nutzer ist in einer Besprechung mit seinem Vorgesetzten“ ist eine eindeutige

Charakterisierung einer Situation für die Nutzung der Funktion „automatisches Einschalten der lautlosen Benachrichtigung“.

Mit der Einführung der Zeitachse in diesem Modell ermöglichen wir die Rückverfolgung von Kontextinformationen. Dies kann bei der Feststellung der aktuellen Situation bzw. bei der Vorhersage zukünftiger Situationen, abhängig von vorherigen Situationen, hilfreich sein. Im folgenden Abschnitt führen wir die Rolle der Zeit in einem Kontextmodell ausführlich auf.

4.2.3 Änderung des Kontextes und die Rolle der Zeit

In Abbildung 4.4 sind zwar die Aspekte Beteiligte, Aktivität und Einsatzumgebung zusammen mit ihren Wechselwirkungen unter einander zentral dargestellt, aber eine weitere wichtige Information in diesen Abbildungen betrifft die Zeit.

Bei der Charakterisierung der Situation einer Anwendung haben wir aufgeführt, dass sie zeitbezogen ist. Sie ist zu einer Zeit t durch die momentane Belegung eines Kontextmodells eindeutig bestimmt. Bei dem grundlegenden Modell des Kontextes in Abbildung 4.4 spielt die Zeit eine fundamentale Rolle[2]:

1. Die Zeit ändert sich ständig, und die drei zentralen Aspekte des Kontextes einer Anwendung können sich ebenfalls im Laufe der Zeit ändern;
2. Zeit ist ein wichtiger Schlüssel zur Feststellung einer Situation, und Situationsänderungen lassen sich durch das Konzept von Zeit feststellen;
3. Aus Sicht eines Nutzers – und diese Sicht ist schließlich die wichtigste Sicht bezüglich der Nutzerakzeptanz der Anwendungen – immer die aktuelle Nutzungssituation interessant ist, und „Aktuell“ grundsätzlich bezogen auf einen Zeitbegriff ist.

In dem grundlegenden Modell des Kontextes betrachten wir Zeit hinsichtlich Situationen in *der Vergangenheit, der Gegenwart und der Zukunft* (**Stempel einer Kontextinformation**, siehe Abbildung 4.5 Teil a-). Die Einführung einer Zeitachse bietet die Möglichkeit Werte aller Faktoren einer Situation (vorwärts und rückwärts) zu verfolgen, und aktuelle (zum Zeitpunkt t_i), vergangene (zum Zeitpunkt t_j mit $j < i$) sowie zukünftige (zum Zeitpunkt t_k mit $k > i$) Situationen zu bestimmen. Mit der Gegenwart meinen wir die aktuelle Situation einer Anwendung, ohne explizite Behandlung der Historie. Die Betrachtung der Vergangenheit wird wichtig, wenn uns die **Historie einer Kontextinformation** interessiert, d.h. wenn sie relevant für die Adaption der Anwendung ist. Abhängig von dem zu untersuchenden Gesichtspunkt kann eine explizite Behandlung der Vergangenheit erforderlich sein. Sie wird beispielsweise erforderlich, wenn die Adaption eine gewisse Antizipation bedarf [Geihs, 2008]. Dadurch wird nämlich ermöglicht, Vorhersage mit relativer Wahrscheinlichkeit bzgl. zukünftigen Situationsänderungen auf Basis vergangener Belegungen eines Kontextmodells (also vergangener Situationen) zu liefern.

[2] Die Zeit spielt in unserem grundlegenden Modell eine fundamentale Rolle: die drei zentralen Aspekte ändern sich über die Zeit. Man kann jedoch auch argumentieren, dass sich beispielsweise Aktivitäten und Einsatzumgebung abhängig von den Beteiligten ändern oder dass Aktivitäten abhängig von dem Einsatzort ändern. Diese Art von Abhängigkeit fallen unter die aufgeführten Wechselwirkungen zwischen der einzelnen Aspekten von Kontextinformationen, die so genannten Inter-Aspekt Wechselwir-

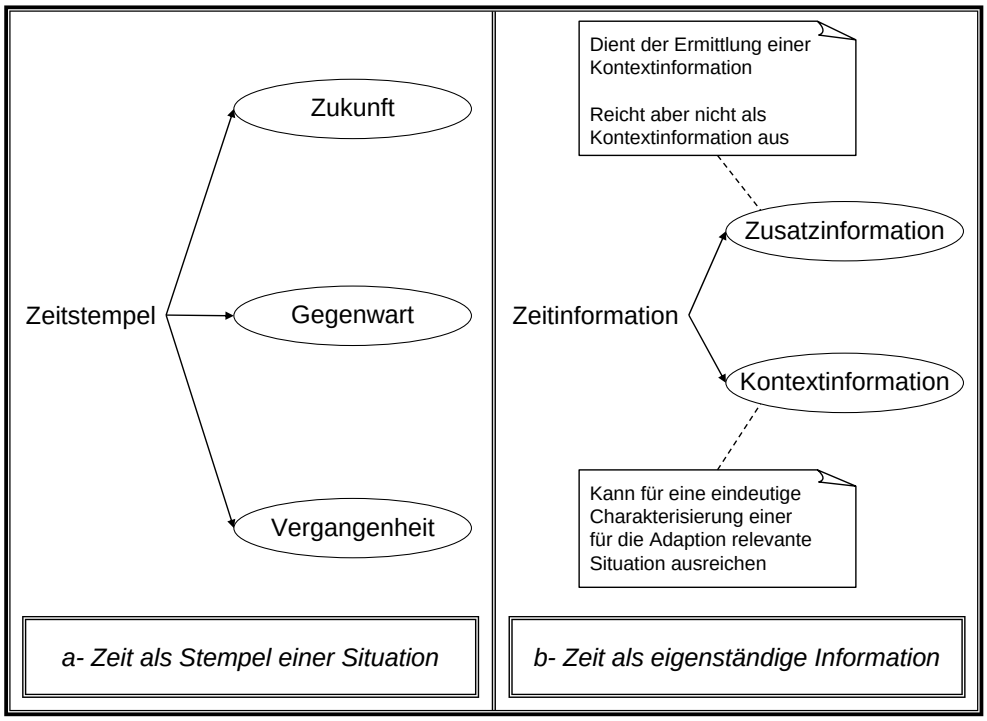

Abbildung 4.5: Rolle von Zeit in einem Kontextmodell

Während Kontextbelegungen nicht immer eindeutig für die Zukunft im Voraus bestimmt werden können, können Vorhersage-Modelle und Methoden verwendet werden, um diverse Änderungen von Kontextinformationen vorauszusagen. Beispielsweise, dass ein Nutzer in dieser Woche auch Termine bis über 19 Uhr wahr nimmt, lässt nicht darauf schließen, wie er mit Terminen in zwei Wochen umgehen wird, die über 18 Uhr hinaus gehen. Es kann nämlich sein, dass in zwei Wochen das Wetter schön sein wird und der Nutzer bereits ab 18 Uhr Feierabend machen wird. Wetterbedingungen z.B. werden routinemäßig für gewisse Orte und Zeiträume vorhergesagt. Ein kontextsensitiver Terminplaner würde beispielsweise in Betracht ziehen, dass der Nutzer in der Vergangenheit bei schönem Wetter etwas früher als sonst Feierabend gemacht hat. Alle Termine in zwei Wochen, die erst nach 19 Uhr beginnen, würde das System bereits bei der Anfrage ablehnen oder sinnvollerweise nicht automatisch zusagen, wenn vorhergesagt wurde, dass in zwei Wochen das Wetter schön sein wird. Die bei der Behandlung von Terminen verwendeten Entscheidungslogik beruht also auf die Historie gewisser Kontextinformationen. Ein weiteres Beispiel, in der die Historie der Daten eine wichtige Rolle spielt, betrifft einen kontextsensitiven Bekleidungsassistent. Das System würde für seine Empfehlungen Wettervorhersage in Betracht ziehen, welche auf die Historie der Wetterbedingungen basieren.

Abgesehen von der Bedeutung der Zeit als Stempel von Kontextinformationen, also Zeitachse für die Feststellung von Situationsänderung (Vergangenheit, Gegenwart und Zukunft), kann die Zeit als **eigenständige Information** betrachtet werden. Zu solchen eigenständigen Informationen zählen Konzepte wie etwa Tageszeit (z.B. am Morgen, am Mittag, am Abend), Werktage oder auch Jahreszeit. Allerdings ist hier Vorsicht ge-

kungen.

boten: eine weiterführende Differenzierung der Informationen, wie in Abbildung 4.5 Teil b-, ist notwendig, um etwas Klarheit zu verschaffen. In der Literatur existieren eine Reihe von Arbeiten, die die Wichtigkeit der Zeit in der Betrachtung von Kontext betonen, allerdings nicht aufführen, inwiefern Zeit als Kontextinformation behandelt werden kann. In [Henricksen et al., 2002] z.B. wird die Bedeutung von Zeit hinsichtlich der Vergangenheit und der Gegenwart hervorgehoben. Zeit wird in Zusammenhang mit Aktivitäten über einen Zeitraum verwendet; z.B. die Zeit einer Verabredung. In [Tarasewich, 2003] wurde Zeit hauptsächlich als Stempel einer Kontextinformation aufgeführt. Jede Kontextinformation trägt einen Zeitstempel, anhand dessen festgestellt wird, ob sie in der Vergangenheit, Gegenwart oder Zukunft liegt. Auch die Behandlung der Zeit als Zusatzinformation zur Bestimmung anderer Kontextinformation wird bei Tarasewich erwähnt. Allerdings wird dort nicht aufgeführt, ob Zeit als explizite Kontextinformation fungieren kann oder nicht.

Zeit wird nur dann als explizite Kontextinformation behandelt, wenn sie eine unmittelbare Adaptionsentscheidung bewirken kann (also als Charakterisierung einer Situation). Zum Beispiel *„morgens ist es kalt [und wenn es kalt ist, empfehlt der Bekleidungsassistent dem Nutzer einen Pullover anzuziehen]“*. Die Zeitangabe „morgens“ ist an dieser Stelle lediglich eine Zusatzinformation zur Feststellung, ob es kalt ist oder nicht. Sie reicht nicht aus, um eine Adaptionsentscheidung zu treffen. Denn in tropischen Ländern z.B. ist es morgens nicht unbedingt kalt. Würde das System seine Entscheidung lediglich von der Zeitinformation „morgens“ abhängen, so werden seine Empfehlungen für einen Nutzer in einem tropischen Land nicht angebracht sein. Das deutet auf ein Defizit im Requirements Engineering hin: es wurde nämlich nicht die notwendige Kontextinformation für den Bekleidungsassistent ermittelt. Am Beispiel des kontextsensitiven Scheibentönungssystems erläutern wir den selben Sachverhalt wie folgt: *In der Nacht wird es relativ dunkel draußen*. „In der Nacht“ ist hier allerdings keine Kontextinformation, sondern eher eine Zusatzinformation zur Feststellung der Kontextinformation „es ist dunkel“; vorausgesetzt es ist festgelegt, was das genau bedeutet (z.B. es ist dunkel bedeutet die Lichtstärke ist kleiner als 5 Lux). Die Information „es ist dunkel“ wird demnach als Kontext verwendet, um eine Tönung der Scheiben zu bewirken. Es ist diese Information, die dem System über Sensorik bereitgestellt wird, und nicht etwa die Information „es ist Nacht“. Die Dunkelheit kann nämlich von anderen Faktoren abhängen, etwa Wetterbedingungen wie Trübheit oder Art der Straße wie Allee-, Brücken- oder Tunnelbereich.

In dem folgenden Beispiel aus dem kontextsensitiven Terminplaner hingegen ist Zeit eine eigenständige Kontextinformation. Anfragen dienstlicher Termine, welche an einem Sonntag stattfinden, werden von dem Terminplaner nicht ohne explizite Zustimmung des Nutzers zugesagt, selbst wenn er an dem genannten Sonntag genug Zeit hätte. Die Information Wochentag kann in diesem Fall ohne Weiteres eine Adaption bewirken. Sie kann dadurch als Kontextinformation behandelt werden.

4.3 Überblick des Frameworks

Mit der Erstellung eines grundlegenden Modells des Kontextes der Nutzung einer Anwendung (und somit der Situation als Instanz des Kontextmodells), verfolgen wir das Ziel, die Erstellung eines Kontextmodells konstruktiv zu gestalten und dabei keinen

Aspekt unbewusst zu vernachlässigen. Bei der Kontextmodellierung möchten wir die relevanten Informationen aus den einzelnen Aspekten nicht aus den Augen verlieren. Darüber hinaus trägt ein solches grundlegendes Modell dazu bei, wie wir später im Kapitel 5 sehen werden, die Analyse der Wirkung von Situationsänderungen auf die Funktionalitäten einer kontextsensitiver Anwendung systematisch durchzuführen.

Um die Erfassung und die Analyse von Kontextinformationen zu erleichtern, führen wir einen konzeptuellen Rahmen ein, innerhalb dessen die einzelnen Aspekte des Kontextes in geeignete Modelle überführt werden. Die Modelle sind zunächst einzeln iterativ zu entwickeln und anschließend miteinander integriert. Durch die Entwicklung der einzelnen Modelle für sich wird sichergestellt, dass kein Aspekt unbewusst außen vor bleibt. Die Integration der Modelle ist allerdings notwendig, zumal eine Situation das Zusammenspiel zwischen den einzelnen Aspekten zu einem gegebenen Zeitpunkt beschreibt. Außerdem bilden die einzelnen Modelle nicht notwendigerweise die für die Adaption der Anwendung nötigen Informationen bzgl. der Situationsänderungen eindeutig ab. Zur Erstellung der einzelnen Modelle, und natürlich auch des integrierten Modells, werden Modellierungstechniken benötigt, welche gewissen Kriterien genügen müssen.

In diesem Abschnitt führen wir eine Übersicht des Frameworks in Form eines integrierten Modells des Kontextes zusammen mit den Kriterien zur Auswahl der Modellierungstechniken auf.

4.3.1 Integriertes Modell des Nutzungskontextes

Ein integriertes Modell des Nutzungskontext hat die unterschiedlichen Aspekte des Kontextes zu berücksichtigen. Abbildung 4.6 stellt eine Übersicht des integrierten Modells dar.

Das Modell besteht aus einem *Nutzermodell*, einem *Aufgabenmodell* und einem *Umgebungsmodell*, welche die drei zentralen Aspekte von Kontext unmittelbar abbilden. Über diese drei Modelle hinaus können weitere Modelle in einem integrierten Modell von Kontext vorkommen. Diese weiteren Modellen (auch *unterstützende Modelle* genannt) werden, im Gegensatz zu den ersten drei, die grundsätzlich zwingend sind, als optionale Teilmodelle eingestuft. Dazu gehören ein *Plattformmodell*, ein *Interaktionsmodell* und ein *Darstellungsmodell*. Eine Beschreibung der einzelnen Modelle erfolgt weiter unten in diesem Kapitel.

Das Vorhandensein eines optionalen Teilmodells hängt insbesondere davon ab, welches Anwendungsgebiet betrachtet wird – z.B. Telekommunikation (Mobiltelefon), Automobil (Navigation) oder Smart Environment (Küchen-Assistent) – und welche Ziele mit der Entwicklung der Anwendung verfolgt werden. Wird beispielsweise mit der Entwicklung einer Anwendung eine Unterstützung einer Reihe von Darstellungsarten der Informationen bezüglich der Interaktion zwischen dem Benutzer und dem System in unterschiedlichen Situationen angestrebt, so ist es wichtig, ein besonderes Augenmerk auf diesbezügliche Informationen zu richten. Demnach werden Informationen über die aktuelle Darstellungsmöglichkeit für die Adaption des Systems entsprechend relevant behandelt, und demnach auch explizit als Kontext modelliert. In diesem Fall werden sie in einem Darstellungsmodell festgehalten. In einem Smart Environment sind die physikalische Infrastruktur und das Zusammenspiel der Geräte

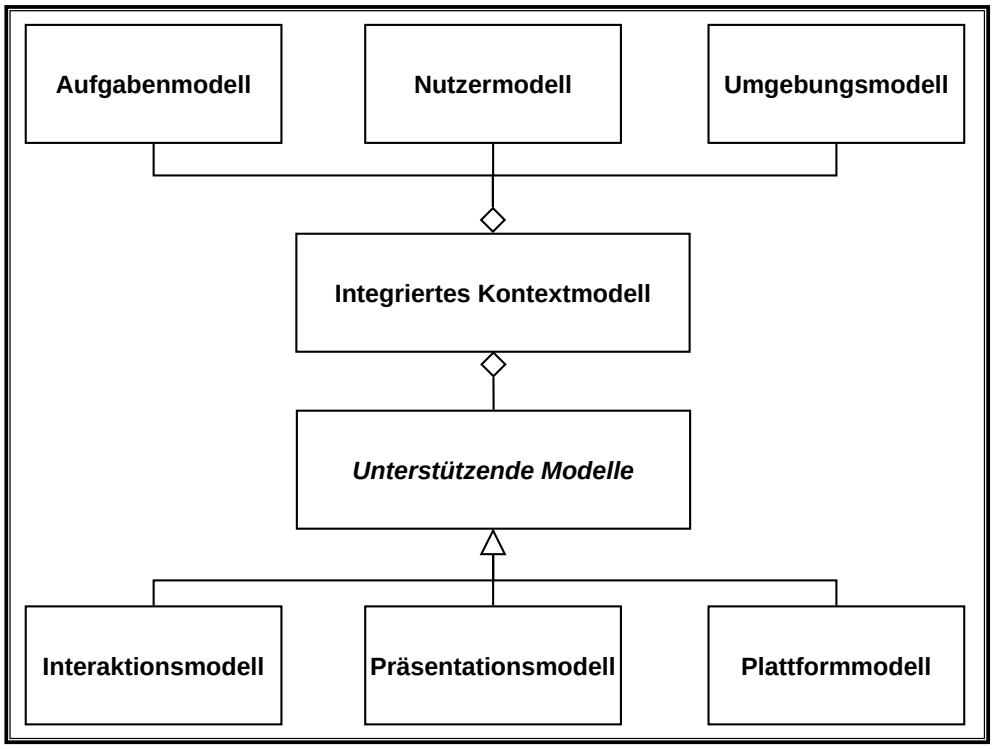

Abbildung 4.6: Integriertes Modell des Nutzungskontextes

grundsätzlich variabel; Hinreichende Informationen über diese Infrastruktur können besonders in diesem Bereich für Adaptionsentscheidungen relevant sein; Es bietet sich daher an, in diesem Fall ein Plattformmodell zu erstellen, welches die Infrastruktur explizit abbildet.

4.3.2 Auswahlkriterien der Modellierungstechniken

Für die Modellierung und Integration der einzelnen Teilaspekte/Teilmodelle eines Kontextmodells auf einer konzeptuellen Ebene werden Techniken benötigt. Es existiert zwar eine Reihe von Techniken, die zur Modellierung von Kontextinformationen eingesetzt werden können, allerdings muss die Auswahl der Techniken gezielt erfolgen. Die eingesetzten Techniken müssen gewisse Kriterien erfüllen. Im Folgenden stellen wir eine Reihe von Kriterien vor, welche die Modellierungstechniken genügen müssen.

Das Hauptkriterium besteht in der Eignung der Techniken zur *Modellierung von Informationen auf einer konzeptuellen Ebene*, wie im Requirements Engineering. Auf einer solchen Ebene befassen wir uns mit der Modellierung von Konzepten aus wirklichen Dingen der Problemdomäne. Die extrahierten Konzepte werden auf Modellelemente abgebildet, welche einzelne Aspekte der Nutzungssituationen der Anwendung repräsentieren. Ein Ziel, das im RE kontextsensitiver Anwendungen verfolgt wird, betrifft die Modellierung des Nutzungskontextes. Dabei wird ein Informationsmodell der unterschiedlichen Situationen der Nutzung der Anwendung als Teil der Problemstellung beschrieben. Wir sind im RE allerdings nicht daran interessiert, wie Kontextmodelle

technisch realisiert werden. Auch sind wir nicht daran interessiert, welche Sensoren benötigt werden, um Kontextinformationen zu sammeln. Diese Fragestellungen wurden beispielsweise bereits in [Dey, 2000] und [Schmidt, 2002] thematisiert. Vielmehr zielen wir darauf ab, festzustellen, welche Informationen überhaupt zu erfassen bzw. abzubilden sind. Wir extrahieren Konzepte aus der Wirklichkeit, mit welchen sich die unterschiedlichen Situationen charakterisieren lassen. Die Entscheidung, welche Sensorik eigentlich eingesetzt wird, ist eher eine Entwurfsentscheidung. Sie kann erst dann getroffen werden, wenn es bekannt ist, welche Informationen zu erfassen sind. Das Gleiche gilt für die konkrete Realisierung bzw. Implementierung von Kontextmodellen. Dies kann ebenfalls erst dann erfolgen, wenn die Informationen bzw. die Art der Informationen, die in dem Modell abgebildet werden bekannt sind. Demzufolge ist die Feststellung der Informationen, die eine Situation charakterisieren, ein unvermeidbarer Schritt im Prozess der Entwicklung kontextsensitiver Anwendungen.

Bei der Modellierung von Kontextinformationen sind wir daran interessiert, bestimmte Objekte (konkrete Gegenstände oder abstrakte Dinge) und Beziehungen zu beschreiben. Hinter den Objekten und Beziehungen stecken Konzepte, welche von den wirklichen Nutzungssituationen extrahiert werden. Die Extraktion der Konzepte ist allerdings abgestimmt mit den unterschiedlichen Stakeholder (insbesondere Nutzern, Experten der Anwendungsdomäne und Entwicklern) durchzuführen und zu dokumentieren. Darüber hinaus können die zu extrahierenden Konzepte je nach Detaillierungsgrad komplexer Natur sein. Dies erfordert folgende Eigenschaften der eingesetzten Techniken:

- *Einfachheit und Übersichtlichkeit*
  Die verwendeten Ausdrücke und Beziehungen sollten einfach wie möglich gehalten werden, damit die erstellten Modelle verständlich, nachvollziehbar und leicht kommuniziert werden können. Darüber hinaus sollten die erstellten Modelle übersichtlich gehalten werden. Dieses Kriterium ist besonders wichtig für die Zwecke der Abstimmung mit den relevanten Stakeholder.
- *Ausdrucksfähigkeit*
  Mit der gewählten Technik soll es möglich sein, die extrahierten Konzepte auf unterschiedliche Detaillierungsgrade zu beschreiben. Bei der Modellierung von Kontext variiert die Detaillierung der Konzepte, je nach dem wie fein granulär die Informationen sein sollen, welche für die Adaption relevant sind.
- *Erweiterbarkeit*
  Techniken zur Modellierung des Kontextes auf einer konzeptuellen Ebene dürfen nicht restriktiv sein. Es soll möglich sein neue, bislang nicht angedachte, Konzepte (Objekte und Beziehungen) in den Modellen aufzunehmen. Dieses Kriterium wird insbesondere im Falle von Iterationen sehr wichtig. Es ist empfehlenswert die Modelle in mehreren Iterationen zu erstellen, damit sich Erkenntnisse, die bei der Erstellung der Teilmodelle des Kontextmodells in einer Iteration gewonnen werden, in einer nächsten Iteration in den restlichen Teilmodellen integrieren werden können. Oft werden pro Iteration neue Erkenntnisse gewonnen, die in den restlichen Modellen integriert werden müssen. Bestehende Modelle sollen daher mit den Konstrukten der verwendeten Modellierungstechnik erweitert werden können.

Grundlage des in diesem Kapitel vorgeschlagenen Framework zur Modellierung des

Kontextes ist ein Ansatz, bei dem der Kontext zunächst aus unterschiedlichen Perspektiven modelliert wird, und anschließend in ein Modell integriert wird. Durch die Modellierung des Kontextes aus unterschiedlichen Perspektiven wird angestrebt, die Modellierung des komplexen Geflechts des Kontextes systematisch zu gestalten. Allerdings ist die Eignung einer einzigen Technik zur Modellierung des Kontextes aus aller Perspektiven in der Literatur umstritten und praktisch auch nicht zu empfehlen [Henricksen et al., 2004, Strang und Linnhoff-Popien, 2004, Baldauf et al., 2007]. Informelle Beschreibungsmittel, in Form von freiem oder strukturiertem Text, können grundsätzlich eingesetzt werden, um den Kontext aus jeglichen Perspektiven zu beschreiben. Allerdings werden die Beschreibungen schnell unübersichtlich und fehlerträchtig. Für die Zwecke der Identifikation der Nutzungssituationen ist es wichtig, den Überblick während der Modellierung des Kontextes (aus den einzelnen Perspektiven) zu behalten.

Ein letztes aber nicht unwichtiges Kriterium ist die Eignung der Techniken zur *Modellierung der Konzepten des betrachteten Aspekts von Kontext*. Wie bereits erwähnt, modellieren wir den Kontext jeweils mit dem Fokus auf den unterschiedlichen Aspekten. Je nach dem welcher Aspekt gerade betrachtet wird, werden verschiedene Konzepte betont. So sind beispielsweise bei der Modellierung der Interaktion zwischen dem Nutzer und dem System nicht notwendigerweise Objekte und Beziehungen die wichtigsten und alleinigen Konzepte.

Die Kriterien sind zwecks der Übersicht in Tabelle 4.1 zusammengefasst.

Nach der Aufführung des integrierten Modells sowie der Kriterien zur Auswahl geeigneter Modellierungstechniken in diesem Abschnitt, gehen wir in den Abschnitten 4.4, 4.5, 4.6 und 4.7 auf die einzelnen Teilmodelle ein. Dabei beschreiben wir jeweils ihre potenziellen Informationsgehalte und führen Techniken auf, mit denen sie modelliert werden können. Wir betonen noch einmal, dass die Modelle eher für die Zwecke des Requirements Engineering erstellt werden, als für die technische Realisierung bzw. unmittelbare Implementierung von Kontextmodellen.

4.4 Modellierung der involvierten Nutzer

In dem vorliegenden Framework bilden wir alle relevanten Informationen bezüglich den Beteiligten in ein Nutzermodell ab. In diesem Abschnitt führen wir die in einem Nutzermodell zu erwarteten Inhalte auf und stellen Beschreibungstechniken vor, die sich zur Modellierung der Nutzer eignen.

4.4.1 Informationsgehalt eines Nutzermodells

Konzepte der Benutzermodellierung werden insbesondere in der Künstlichen Intelligenz eingesetzt, um es zu ermöglichen, dass sich Software-Anwendungen an den jeweiligen Benutzer anpassen. Eine Reihe von Forschungsarbeiten erfolgen auf dem Spezialgebiet des *User Modeling* [Kobsa, 1994, Kay, 1994, Kules, 2000, Fischer, 2001, Chen et al., 2002, Jameson, 2005] mit dem Ziel, die Anwendungen an den Benutzer anzupassen und ihn bei der Lösung seiner Aufgaben zu unterstützen. Dafür werden Nutzermodelle (auch Benutzermodelle) erstellt, welche relevante Informationen über

Kriterium	*Kurzbeschreibung*
Kr. 1) Modellierung auf einer konzeptuellen Ebene	*Im RE kontextsensitiver Anwendungen sind wir daran interessiert charakteristische Informationen der Nutzungssituationen der Anwendung zu modellieren (zu ermitteln, und integriert mit den Anforderungen zu analysieren und dokumentieren). Die verwendeten Techniken sollen in diesem interdisziplinären Prozess eingesetzt werden können.*
Kr. 2) Einfachheit und Übersichtlichkeit	*Die Einfachheit gilt für die verwendeten Konstrukte, damit sie leicht erlernt werden können. Die Übersichtlichkeit gilt für die erstellten Modelle, damit sie verständlich und nachvollziehbar bleiben und leicht kommuniziert werden können. Dies ist notwendig, damit die unterschiedlichen Stakeholder (u.a. Domänenexperte und Nutzer) mitwirken können.*
Kr. 3) Ausdrucksfähigkeit	*Die Ausdrucksfähigkeit der Beschreibungstechnik ist besonders wichtig, damit die Konzepte in unterschiedlichen Granularitäten modelliert werden können.*
Kr. 4) Erweiterbarkeit	*Die Erweiterung der erstellten Modellen, beispielsweise in einer späteren Iteration, soll mit den Konstrukten der verwendeten Beschreibungstechnik ohne Weiteres möglich sein.*
Kr. 5) Spezifika der modellierten Konzepte	*Die Art der Informationen, die modelliert werden, spielt bei der Wahl der Beschreibungstechnik eine entscheidende Rolle. Es soll sichergestellt werden, dass sich die Besonderheiten der zu betrachteten Konzepte mittels der gewählten Technik beschreiben lassen.*

Tabelle 4.1: Kriterien zur Auswahl der Beschreibungstechniken

die Nutzer speichert, auswertet und der Anwendung zur Verfügung stellt.

Die Notwendigkeit eines Benutzermodells in einem Kontextmodell liegt insbesondere darin, dass kontextsensitive Anwendungen, wie sie in dieser Arbeit betrachtet werden, in der Regel nicht notwendigerweise von einer homogenen Nutzergruppe genutzt werden. Es existiert daher nicht „der Nutzer“ einer Anwendung.

Ein Zweck der Benutzermodellierung als Teil der Kontextmodellierung ist die Perso-

nifizierung der Anwendung. Mit der Personifizierung wird die Anwendung an Nutzerpräferenzen und -fähigkeiten angepasst. Beispielsweise bei der Anpassung einer Anwendung für eine taube Person, wird keine Sprachausgabe gemacht. Benutzermodelle sollen unter anderem die individuellen Fähigkeiten, Vorkenntnisse und Präferenzen des Nutzers abbilden. Eine verbreitete Definition für Benutzermodell stammt von [Wahlster und Kobsa, 1989] und besagt:

„A user model is a knowledge source in a natural-language dialog system which contains explicit assumptions on all aspects of the user that may be relevant to the dialog behavior of the system. These assumptions must be separable by the system from the rest of the system's knowledge."

Diese Definition enthält interessante Aspekte wie „explizite Annahmen über den Nutzer" sowie die „die Relevanz der Annahmen für das Dialogverhalten des Systems". Wir bauen darauf auf und führen für die Zwecke der Benutzermodellierung im Zusammenhang mit der Kontextmodellierung die Definition 4.1 ein.

Definition 4.1 Benutzermodell für kontextsensitive Anwendungen

☞ Ein Benutzermodell (auch Nutzermodell) in einer kontextsensitiven Anwendung ist ein Modell, das Annahmen über alle Benutzeraspekte enthält, die für die Adaption der Anwendung relevant sind. Dazu gehören Annahmen über Nutzerdaten (demographische Daten und Präferenzen des Nutzers) und Nutzungsdaten (Benutzungsverhalten). Es ist ein potentieller Bestandteil des Kontextmodells.

Bei der Erstellung eines Benutzermodells als Teil des Kontextmodells einer kontextsensitiven Anwendung muss identifiziert werden, welche Informationen über den Nutzer für die Anwendung notwendig sind, damit sie sich situationsgerecht anpassen kann. Zwar interessiert uns im Hinblick auf der Implementierung der Anwendung ebenfalls, wie diese Informationen gewonnen werden (Sensorik, direkte Nutzereingabe zu Beginn bzw. während der Nutzung der Anwendung), allerdings ist dieser Gesichtspunkt weniger relevant für diese Arbeit[3].

Ein Nutzermodell, als Teil eines Kontextmodells, soll den Aspekt der Beteiligte widerspiegeln. Hinsichtlich der Inhalte eines Benutzermodells wird unterschieden zwischen *Benutzerdaten und Benutzungsdaten*.

Mit **Benutzerdaten** meinen wir, angelehnt an den Klassifizierungen von [Kobsa, 1993] und [Essing, 2001]:

- *Demografische Daten des Nutzers:* das sind personenbezogene Eigenschaften wie Alter, Körpergröße, Geschlecht, Bildungsabschluss und Beruf. Welche Eigenschaften relevant sind, hängt von der betrachtete Anwendung ab. Bei der Fallstudie des Scheibentönungssystems kann die Körpergröße des Fahrers beispielsweise für die automatische Tönung der Scheiben relevant sein, denn die Berechnung des Lichteinfallswinkels z.B., und damit einhergehend der Intensität der

[3] Die niedrigere Relevanz der Art und Weise, wie die Informationen bzw. Daten eines Benutzermodells gewonnen werden, liegt daran, dass wir uns im Rahmen dieser Arbeit mit den RE-bezogenen Fragestellungen der Benutzermodellierung als Teil der Kontextmodellierung beschäftigen. Auch die Frage, wie das Modell im Laufe der Zeit anzupassen ist, ist nicht Thema dieser Arbeit. Vielmehr geht es darum, zu identifizieren, welche Informationen überhaupt notwendig bzw. für die Adaption relevant sein können.

Blendung, ist unter anderem auch von der Körpergröße[4] des Fahrers abhängig. Bei dem kontextsensitiven Terminplaner hingegen hat die Körpergröße keine Relevanz.

- *Präferenzen des Nutzers:* das sind bevorzugte Einstellungen beispielsweise hinsichtlich Sprache, Ton bzw. Lautstärke oder Video/Bild. Bei dem Scheibentönungssystem kann der Fahrer seine bevorzugten minimalen und maximalen Werte für die Tönung der Scheiben vorgeben (minimaler und maximaler Tönungsgrad). Präferenzen des Nutzers werden prinzipiell explizit vor der Nutzung der Anwendung gesammelt, sie können jedoch auch von dem Nutzungsverhalten zur Laufzeit abgeleitet werden. Welche Präferenzen für eine Adaption der Anwendung relevant sind, sind allerdings in dem Nutzermodell festzuhalten.
- *Weitere Daten des Nutzers:* über die Präferenzen und die demografischen Daten hinaus gibt es noch eine Reihe von Daten eines Nutzers, die ebenfalls in einem Benutzermodell festgehalten werden können. Dazu zählen *kulturelle und soziale Hintergründe des Nutzers*. Das sind prinzipiell gesellschaftliche Faktoren, die für die Adaption einer Anwendung von Bedeutung sein können. Sie können die anderen Nutzerdaten maßgeblich beeinflussen. Beziehungen zwischen dem Nutzer und weiteren Personen in der Nutzungsumgebung sind erste Kandidaten für die gesellschaftlichen Faktoren. Auch *physische und psychische Zustände des Nutzers* sind mögliche Inhalte eines Benutzermodells. Beispiele hierfür sind Daten bezüglich Stress, Ärger, Stimmung und Blutdruck.

Die **Benutzungsdaten** betreffen das Nutzungsverhalten des Benutzers über die Zeit (z.B. seine Selektionen in Navigationsmenü oder Hilfetexte mit bestimmten Inhalten), die Bewertungen des Benutzers über Brauchbarkeit der Anwendung (User Feedback), Regelmäßigkeiten im Benutzungsverhalten wie z.B. häufig vorkommende Aktionssequenzen [Kobsa, 1994]. Bei die Modellierung des Nutzungsverhaltens unterscheiden wir zwischen *gewöhnlichen Nutzern, vorsichtigen Nutzern und erfahrenen Nutzern*. Diese Klassifikation ist analog zu unserer Klassifikation kontextsensitiver Anwendungen im Abschnitt 3.2. Gewöhnliche Nutzer haben in der Regel keine Vorstellung des adaptiven Verhaltens der Anwendung. Aus ihrer Sicht kann sich die Anwendung sogar nicht-deterministisch erscheinen. Vorsichtige Nutzer hingegen können das Verhalten der Anwendung nachvollziehen. Sie wagen es jedoch nicht dieses Verhalten durch Manipulation der für die Adaption der Anwendung verwendeten Kontextinformationen zu beeinflussen. Erfahrene Nutzer beeinflussen die verwendeten Kontextinformationen mit der Absicht, ein gewisses Reaktionsmuster der Anwendung zu erzwingen. Diese Informationen können für die Adaption einer Anwendung relevant sein. Sie werden insbesondere für die Zwecke der nachträgliche Anpassung der Logik der Adaption (Kalibrierung) der Anwendung benötigt (siehe Definition 2.5, [Fahrmair, 2005]). Erfahrene Nutzer versuchen Adaptionen der Anwendung zu erzwingen. Es besteht somit die Gefahr, dass die Anwendung für unbeabsichtigte Zwecke genutzt wird oder unerwünschte Nebeneffekte bei der Nutzung der Anwendung entstehen. Die Benutzungsdaten sollten daher bei der Benutzermodellierung im RE nicht unberücksichtigt bleiben.

Für die spätere Integration der Teilmodelle des Kontextes (siehe Abschnitt 4.8) ist es

[4] oder z.B. von der Höheneinstellung des Sitzes

wichtig, Integrationspunkte bzw. Bindeglieder (Entitätstypen in Faktenmodellen) in den einzelnen Teilmodellen, auch in dem Nutzermodell, zu identifizieren. Kandidaten für solche Integrationspunkte bei dem Nutzermodell sind die Benutzungsdaten. Sie werden mit den Aufgaben verbunden, die der Nutzer in den Nutzungssituationen durchführt. So wird beispielsweise bei der Durchführung der Aufgabe *Nutzung der Scheibentönungsfunktion* (siehe Abbildung 4.12) die Benutzungsdaten *vom Nutzer bevorzugter Tönungsgrad in Abhängigkeit von der äußeren Helligkeit* (siehe Abbildung 4.9 -d) berücksichtigt. Sie wird so als Bindeglied zwischen dem Nutzermodell und dem Aufgabenmodell bei der Integration der Modelle betrachtet. Die Benutzungsdaten drücken für jeden Nutzer aus, *wie* er die Aufgaben (am liebsten bzw. am gewöhnlichsten) durchführt. Sie können grundsätzlich auch als Attribute der Aufgaben in einem Aufgabenmodell vorkommen, allerdings wird dadurch ihre Abhängigkeit von dem betrachteten Nutzer nicht ausgedrückt und die erwähnte Integrationsmöglichkeit von dem Nutzermodell mit dem Aufgabenmodell nicht mehr gegeben.

4.4.2 Nutzermodellierung mittels ER-Modelle

Für die Modellierung der involvierten Nutzer schlagen wir, aufgrund der im Abschnitt 4.3 erarbeiteten Kriterien, Entity-Relationship-Modell (ER-Modelle) vor. In diesem Abschnitt führen wir zunächst die notwendigen Grundlagen der ER-Modellierung kurz auf und diskutieren anschließend ihre Eignung zur Erstellung von Nutzermodellen.

Einführung in die ER-Modellierung

Das Entity-Relationship-Modell (ER-Modell) wurde von Peter Chen als Datenmodellierungsmittel eingeführt [Chen, 1976]. Es wird zur Modellierung von Systemen und ihren Anforderungen in einem Top-Down-Ansatz verwendet. Grundlegend in ER-Modellierung sind die Konzepte von *Entität* (*entity*) und *Beziehung* (*relationship*). Entitäten sind Objekte (wie Personen, Plätze, Gebäude, Geräte oder ähnliches) und zwischen ihnen existieren Beziehungen.

Jedes ER-Modell enthält also Objekte, welche eindeutig durch einen (interner) Name gekennzeichnet werden. Objekte besitzen Attribute, die jeweils durch ein Tupel $(Name\, an, Wert\, av)$ beschrieben werden. Alle Attribute eines Objekts e_i können durch das Tupel $(e_i, an_1 : av_1, ..., an_k : av_k)$ beschrieben werden, wobei k die Anzahl der Attribute des Objekts e_i ist. Objekte mit gleichartigen Attributen und gleichartigen Beziehungen zu einander werden zu Objektmengen (*entity sets*) E_j zusammengefasst werden. Diese müssen allerdings nicht disjunkt sein.

Zwischen Objekten können verschiedene Arten von Beziehungen bestehen, wie z.B. Klassifikationsbeziehungen (etwa *is kind of, is a*), Bestandteilsbeziehungen (etwa *is part of*), Auftragsbeziehungen (etwa *bearbeitet Auftrag*), geometrische Beziehungen (etwa *liegt auf, ist verbunden mit, ist eingespannt in*), und Vorrangbeziehungen (etwa *muss benachrichtigt werden vor, kann nur gehandhabt werden nach*). Beziehungen haben Stelligkeiten: Eine m-stellige Beziehung ist durch ein Tupel $(rn_1 : e_1, ..., rn_m : e_m)$ gegeben, wobei e_i Objekte sind und rn_i Rollennamen der Objekte in der Beziehung sind. Die in einer Beziehung auftretenden Objekte sind disjunkt, allerdings nicht die Objektmengen, denen sie angehören. Die Rollennamen bezeichnen jeweils die Rolle, welche die

einzelnen Objekte in der Relation spielen. In einer zweistelligen Relation *besitzt* gibt es beispielsweise die Rollen *Nutzer* und *Gerät*. Üblicherweise wird als Rollenname rn_i die Name derjenigen Objektmenge verwendet, der das Objekt e_i angehört. Dies ist wegen der Eindeutigkeit möglich, solange eine Objektmenge in einer Relation nicht mehrfach auftritt. Einer Beziehung können Attribute zugeordnet werden, beispielsweise der Zeitpunkt ihrer Entstehung. Auch zwischen Objektmengen können Mengen von Beziehung bestehen. Diese werden entsprechend ihrer Semantik gruppiert und benannt. Sei $R(E_1, ..., E_m)$ eine solche Menge von $m-$stelligen Beziehungen mit gleicher Semantik, dann gilt:

$$\forall(r_1 : e_1, ..., r_k : e_k) \in R(E_1, ..., E_m) : \\ k = m \wedge \forall i, 1 \leq i \leq m : e_i \in E_i$$

ER-Modelle können als ER-Strukturgraph oder als ER-Tabellen dargestellt werden[5]. Abbildung 4.7 illustriert einen ER-Strukturgraph. Der Graph bildet eine *besitzt*-Beziehung zwischen den Entitäten *Nutzer* und *Gerät*. Die Entität *Nutzer* hat beispielsweise die Attribute *Name, Anschrift und Beruf* und die Beziehung *besitzt* hat einen Attribut *Art* (z.B. dauerhaft/temporär oder privat/beruflich). Die Entitäten werden als Rechtecke dargestellt, die Beziehungen als Raute und die Attribute als Ellipsen. Weitere Details zur ER-Modellierung, inklusive unterschiedlicher Notationen können beispielsweise aus [Thalheim, 2000] entnommen werden.

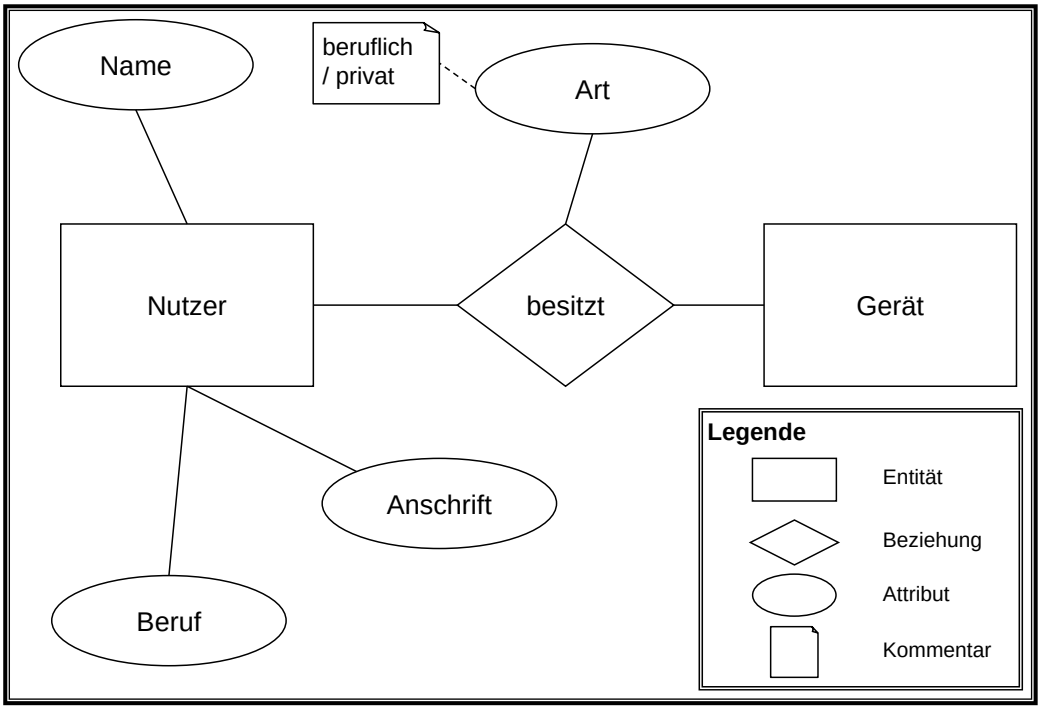

Abbildung 4.7: Beispiel eines ER-Modells

Der ER-Ansatz betrachtet eine Anwendungsdomäne unter dem Aspekt von Entitäten

[5]Jeder in einem ER-Strukturgraph abgebildete Sachverhalt lässt sich auch durch ER-Tabellen ausdrücken. Für jede Objektmenge und jede Beziehungsmenge wird je eine Tabelle angelegt. Der Name der Tabelle ist der Name der Menge. Die Zeilen der Tabelle sind die Tupel, die der zugeordneten Menge angehören. Die Spaltenüberschriften sind die Namen der Tupelelemente und der Spalteninhalt die Werte der Tupelelemente.

(also Objekten), die Attribute haben und an Beziehungen beteiligt sind. Diese Betrachtungsweise ist ziemlich intuitiv, und ist immer noch der populärste Ansatz zur Modellierung von Daten, Informationen und ihre Zusammenhänge. ER-Modelle werden während der Anforderungsanalyse eingesetzt, um Informationen zu beschreiben, welche das zu entwickelnde System verwenden wird. Sie können dazu verwendet werden, um einen Überblick oder eine Klassifikation der benötigten Informationen und ihre Beziehungen unter einander zu erarbeiten.

Eignung von Entity-Relationship zur Modellierung der Nutzer

Aus der Definition 4.1 geht hervor, dass ein Benutzermodell ein Datenmodell des Nutzers ist. Es soll dazu dienen, eine kontextsensitive Anwendung an die Charakteristika des Nutzers (z.B. bevorzugte Sprache, Lautstärke, Augenqualität inklusive Sehstärke oder Farbblindheit) anzupassen. Bei der Benutzermodellierung sind wir also daran interessiert, alle Charakteristika des Nutzers zu identifizieren und festzulegen, welche für die Adaption des Systems benötigt werden. Diese Charakteristika werden als Daten modelliert.

Der Vorschlag, die Nutzer mittels **Entity-Relationship** zu modellieren, basiert auf einer Bewertung der E/R-Modellierung anhand der im Abschnitt 4.3 aufgeführten Kriterien. E/R-Modelle eignen sich grundsätzlich für die Modellierung von Daten auf einer konzeptuellen Ebene [Kemper und Eickler, 1999], was mit dem ersten Kriterium verlangt wird. Bei der Benutzermodellierung haben wir in erster Linie Daten zu modellieren. Somit ist das Kriterium Modellierung auf einer konzeptuellen Ebene erfüllt (Kriterium Kr. 1). Darüber hinaus benötigen wir für die Zwecke der Benutzermodellierung im RE lediglich die Grundkonzepte von Entitäten, Beziehungen und Attributen. Demnach ist die Ausdrucksfähigkeit von ER-Modellen weitgehend ausreichend für unsere Zwecke (Kriterium Kr. 3). Die Bewertung von ER-Modellen hinsichtlich ihrer Einfachheit und Übersichtlichkeit ist etwas subjektiv. Sowohl die Einfachheit der Konstrukte zur Erstellung der Modellen als auch die Übersichtlichkeit der erstellten Modelle sind von dem Grundwissen des Modellierers abhängig (Kriterium Kr. 2). Da die benötigten Konstrukte jedoch sehr gering sind und hauptsächlich Grundlagen der ER-Modellierung sind, lässt sich das notwendige Grundwissen sehr schnell erlangen. Auch die Erweiterbarkeit der ER-Modellen ist grundsätzlich gegeben (Kriterium Kr. 4). Attribute können in Entitäten (und zusätzliche Beziehungen) umgewandelt werden, und bestehende Modelle können um weitere Entitäten, Beziehungen und Attribute erweitert werden. Ein entscheidendes Kriterium ist die Eignung zur Modellierung der Spezifika eines Benutzermodells. Als besonderes Spezifikum von Benutzermodellen gilt die Tatsache, dass es sich um die Modellierung charakteristischer Daten der Nutzer handelt [Langley, 1999]. Diese Daten können als bestimmte Entitäten mit geeigneten Attributen und Beziehungen untereinander modelliert werden, was in ER-Modellierung etwas Grundsätzliches ist. Somit ist das Kriterium Kr. 5 ebenfalls erfüllt.

Abbildung 4.8 illustriert die Benutzermodellierung mittels Entity-Relationship anhand der Fallstudie der kontextsensitiven Scheibentönung. Wesentliche Nutzer des Systems sind Fahrer und eventuelle Beifahrer. In diesem Ausschnitt des Nutzermodells ist abgebildet, dass der Fahrer eine gewisse Körpergröße hat, und für die einzelnen Scheiben (klassifiziert nach Windschutzscheibe, Heckscheibe sowie vorderen und hinteren Seitenscheiben) jeweils einen minimalen sowie einen maximalen Wert für den Grad der

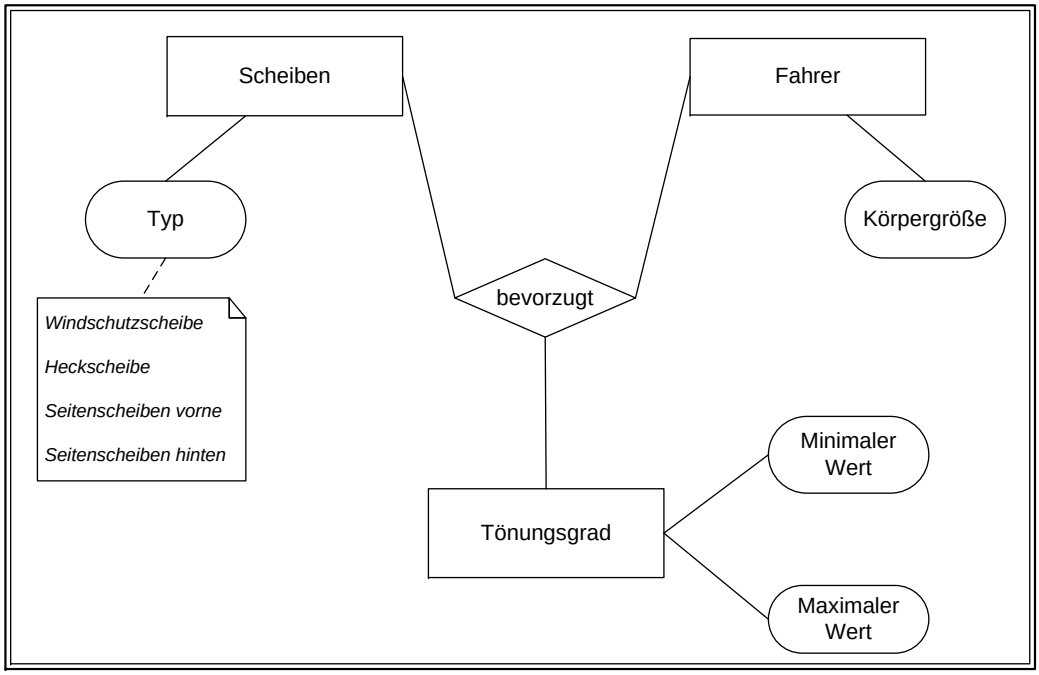

Abbildung 4.8: Ausschnitt des Benutzermodells des Scheibentönungssystems

Tönung (Lichtdurchlässigkeit) bevorzugt.

Zur Modellierung der Benutzerdaten, z.B. Benutzername oder Anschrift, ist die Eignung von E/R-Modellen offensichtlich, zumal diese Informationen grundsätzlich statischer Natur (sie ändern sich selten bis gar nicht in dem Lebenszyklus der Anwendung) sind. Die Eignung von E/R-Modellen zur Modellierung der Benutzungsdaten, z.B. Vorliebe oder Gewohnheiten der Benutzer bei der Nutzung der Anwendung, hingegen bedarf weitere Erläuterungen. Beispielsweise können wir für die Modellierung der Informationen über die Vorliebe des Nutzers bezüglich des Scheibentönungsgrads in Abhängigkeit mit der äußeren Helligkeit (Abbildung 4.9 -a), abgesehen von ER-Modellen (Abbildung 4.9 -b), Zustandsautomaten (Abbildung 4.9 -c) einsetzen. Durch den Einsatz von Zustandsautomaten an dieser Stelle würde beispielsweise ermöglicht, Aussagen über den Wechsel zwischen den einzelnen Zeilen aus der Tabelle in Abbildung 4.9 -a, d.h. Aussagen über Zustandsänderungen zu machen. Allerdings sind wir bei der Nutzermodellierung nicht daran interessiert, den Wechsel zwischen den Informationen zu erfassen. Wir sind lediglich an der Erfassung der Präferenzen der Nutzer interessiert, welche uns durch den Einsatz von ER-Modellen ermöglicht wird. Darüber hinaus haben wir mit ER-Modellen eine Möglichkeit die Informationen über die Benutzungsdaten samt den Benutzerdaten in einem Modell abzubilden (Abbildung 4.9 -d). Die Dynamik, die wir beispielsweise mit Zustandsautomaten abbilden können, geht weit über die, die wir bei der Nutzermodellierung erreichen wollen. Uns reicht es, wenn wir für jeden Nutzer seine Vorliebe und Gewohnheiten bei der Nutzung der Anwendung zusammen mit seinen Benutzerdaten mithilfe von ER-Modellen modellieren. Damit erhalten wir einen Teil (den Benutzer-bezogenen Teil) der charakteristischen Informationen der Nutzungssituationen der Anwendung, welche wir für die Adaption der Anwendung verwenden können.

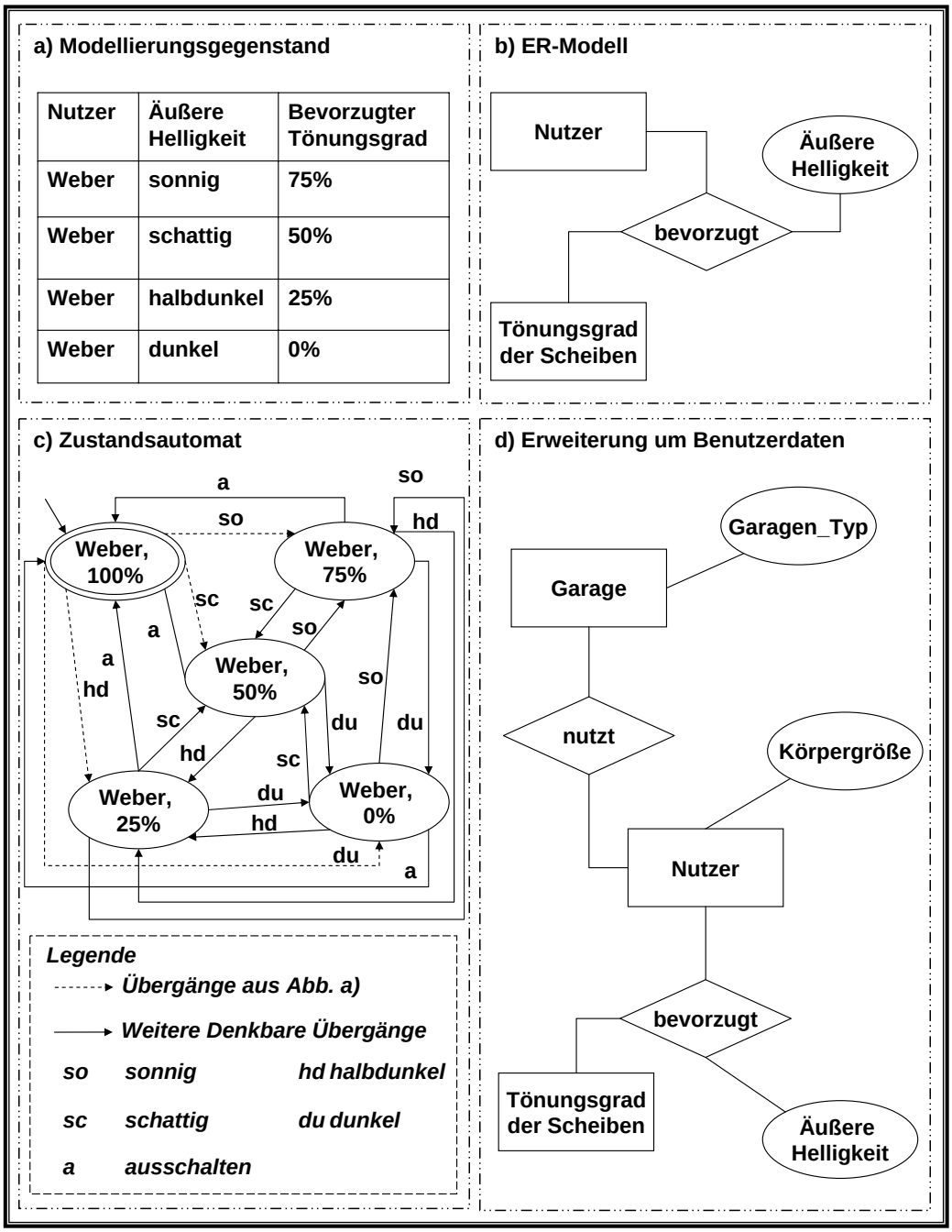

Abbildung 4.9: Modellierung von Benutzungsdaten

4.5 Modellierung der Aufgaben des Nutzers

In dem vorliegenden Framework bilden wir alle relevanten Informationen bezüglich den Aktivitäten in ein Aufgabenmodell ab. In diesem Abschnitt führen wir die in einem Aufgabenmodell zu erwarteten Inhalte auf und stellen eine Beschreibungstechnik vor, die sich zur Modellierung der Aufgaben eignet.

4.5.1 Informationsgehalt eines Aufgabenmodells

Aufgabenmodelle werden oft verwendet, um einerseits Ziele und Bedürfnisse der Nutzer zu verstehen, und andererseits Ergebnisse des Entwurfs zu kommunizieren sowie zu evaluieren [Hartson et al., 1990]. Im HCI-Bereich werden sie prinzipiell eingesetzt, um die unterschiedlichen Aufgaben im Zusammenhang mit der Nutzung einer Anwendung zu identifizieren, analysieren, evaluieren und sie zwischen den involvierten Stakeholder zu kommunizieren.

Die Wichtigkeit eines Aufgabenmodells in einem kontextsensitiven Anwendung wird mit einer Reihe entwickelter Prototypen gezeigt. Abhängig davon, welche Aufgaben der Nutzer gerade durchführt, wird die Anwendung (insbesondere die Art der Interaktion) angepasst. Beispielsweise wird der kontextsensitiven Terminplaner dem Nutzer lautlos benachrichtigen, wenn er gerade an einer Besprechung teilnimmt. In [Paradis et al., 2003] wird ein System (Portal zur Unterstützung von Reportern bei der Erfassung von Artikeln) vorgestellt, das ein Aufgabenmodell (Themendefinition, Ausarbeitung, Schreiben und Validierung des Artikels) verwendet. Abhängig davon welche Aufgabe der Reporter gerade durchführt, wird die Interaktion zwischen ihm und dem System umgestaltet und die vom System angebotenen Inhalte angepasst.

In der Kontextmodellierung zielt die Aufgabenmodellierung darauf, relevante Informationen über die Welt der Aufgaben, welche die Nutzer durchführen, in einem Aufgabenmodell (engl. *task models*) abzubilden. Die Aufgabenwelt (engl. *task world*) setzt grundsätzlich Aufgaben mit Zielen in Beziehung, und beschreibt, welche Objekte (bzw. Ressourcen) der Umgebung sie verwenden, welche Ereignisse sie gegebenenfalls auslösen und welche Nutzer sie durchführen. Eine anschauliche Ontologie der Aufgabenwelt ist beispielsweise in [van Welie et al., 1998] beschrieben. Dabei wird ein Aufgabenmodell wie folgt definiert:

„A task model is a step by step description of a goal to achieve. The description consists of a hierarchical decomposition of a main goal into a set of sub-goals. Each goal is represented by a task or a set of sub-tasks that reflect the different ways to perform the goal."

Grundlegend in dieser Definition ist, dass Aufgaben Ziele verfeinern. Dabei bezeichnen Ziele einen Zustand der realen Welt, welche der Nutzer erreichen möchte, und eine Aufgabe notwendige Aktivitäten zur Erreichung der Ziele bezeichnen. Dieser Zusammenhang wird beispielsweise in [Simon, 1969], [van Welie, 2001] und [Liebermann, 2005] näher erläutert. Aufgaben können ziemlich abstrakt (etwa eine Strategie zur Lösung eines Problems ausdenken) aber auch ziemlich konkret (etwa eine Route auswählen) sein.

Definition 4.2 Aufgabenmodell für kontextsensitive Anwendungen

☞ Ein Aufgabenmodell (auch Task Modell) in einer kontextsensitiven Anwendung ist ein Modell, das Annahmen über alle Aktivitäten des Nutzers bei der Nutzung der Anwendung enthält, die für die Adaption der Anwendung relevant sind. Es ist ein potentieller Bestandteil des Kontextmodells.

Bei der Definition 4.2 beschränken wir uns auf die von den Nutzern durchgeführten Aktivitäten (mit oder ohne Kooperation mit der Anwendung), und modellieren nur

bedingt die internen Aktivitäten der Anwendung. Beispiele für Aktivitäten, die der Nutzer ohne Kooperation mit der Anwendung durchführt, sind *sich entscheiden an einem Termin teilzunehmen und sich entscheiden ein Telefonat durchzuführen*. Aktivitäten, die der Nutzer in Zusammenarbeit (Kooperation oder Interaktion) mit der Anwendung durchführt, sind beispielsweise *Anwendung starten (Terminplaner) bzw. System einschalten (Scheibentönungssystem), Termine bearbeiten, und Termine Anzeigen*. Einige Beispiele für Aktivitäten, die das System intern durchführt, sind *Termin arrangieren, Benachrichtigung verschicken, Scheiben tönen, und Scheiben aufhellen*. Mögliche Aktivitäten, die durchzuführen sind, um Ziele eines Nutzers zu erreichen, werden in einem Aufgabenmodell beschrieben. Dabei sind die temporalen oder semantischen Beziehungen zwischen den Aufgaben nicht zu vernachlässigen. Aufgaben können beispielsweise parallel, sequentiell oder wiederholt durchgeführt werden, um ein Ziel zu erreichen. Darüber hinaus können zur Erreichung eines Ziels alternative Aufgaben durchgeführt werden.

In einem Aufgabenmodell halten wir Ziele (abstrakt), Aktivitäten (etwas konkret) und durchgeführten Aufgaben (ziemlich konkret) fest. Wir modellieren Aufgaben der Art *Termin vereinbaren, an Besprechung teilnehmen oder auch Anruf entgegennehmen.* Bei der Aufgabenmodellierung sind wir daran interessiert solche Aufgaben zu identifizieren, und ihre charakteristischen Merkmale für die Zwecke der Charakterisierung der Nutzungssituationen einer Anwendung zu beschreiben. Dabei ist es wichtig, die identifizierten Aufgaben in eine Hierarchie einzuordnen, d.h. sie mit übergeordneten Zielen bzw. Aufgaben, sowie untergeordnete Aufgaben zu verbinden. Einige Beispiele für charakteristische Merkmale einer Aufgabe sind, abgesehen von ihren übergeordneten Ziele/Aufgaben und untergeordneten Aufgaben in der Aufgabenhierarchie, die Dauer ihrer Durchführung, ihre Priorität sowie ihre Unterbrechbarkeit. Auch die benötigten Interaktionen zur Durchführung einer Aufgabe sowie die dadurch manipulierten Objekte[6] gehören zu den charakteristischen Merkmalen.

4.5.2 Aufgabenmodellierung mittels ConcurTaskTrees

Für die Modellierung der durchgeführten Aufgaben schlagen wir, aufgrund der im Abschnitt 4.3 erarbeiteten Kriterien, ConcurTaskTrees (CTT) vor. In diesem Abschnitt führen wir zunächst die notwendigen Grundlagen der CTT kurz auf und diskutieren anschließend ihre Eignung zur Erstellung von Aufgabenmodellen.

Grundlagen der ConcurTaskTrees

Zur Modellierung von Aufgaben existieren eine Reihe von Techniken. Eine davon ist die ConcurTaskTrees [Paternò et al., 1997, Paternò, 1999].

Abbildung 4.10 illustriert das CTT Meta-Modell. Demnach werden Aufgaben in einem Aufgabenmodell hierarchisch strukturiert und Rollen zugeordnet. Außerdem können Einschränkungen bei den Aufgaben bestehen. Für die Erledigung einer Aufgabe werden Aktionen durchgeführt, und dabei gewisse Objekte manipuliert. Zentrale Merkmale von CTT sind:

[6] Die Objekte gehören teilweise zu der Umgebung, und werden in dem Umgebungsmodell abgebildet. Sie sind aber auch teilweise Bedienelemente des Systems und werden bei Bedarf in einem Präsentationsmodell abgebildet.

Operator	***Kurzbeschreibung***
Independent Concurrency ($T1|||T2$)	*T1 und T2 können ohne Einschränkung in beliebiger Reihenfolge durchgeführt werden, z.B. Angaben des Navi folgen und Scheibentönungssystem einschalten.*
Choice ($T1[]T2$)	*Eine der Aufgaben wird ausgewählt, und sobald die Auswahl gemacht worden ist, kann die ausgewählte Aufgabe durchgeführt werden. Dabei wird die andere Aufgabe mindestens bis zur Beendigung der durchgeführten Aufgabe nicht verfügbar/durchführbar.*
Concurrency with Information exchange ($T1|[]|T2$)	*T1 und T2 können gleichzeitig durchgeführt werden, allerdings müssen sie miteinander synchronisieren, um Informationen auszutauschen.*
Order Independence ($T1| = |T2$)	*T1 und T2 sind durchzuführen. Allerdings, wenn die eine durchgeführt wird, muss sie beendet werden, bevor die andere durchgeführt werden kann.*
Deactivation ($T1[> T2$)	*Sobald T2 begonnen hat, wird T1 abgebrochen und beendet.*
Enabling ($T1 >> T2$)	*Sobald Task 1 abgeschlossen ist, wird Task 2 begonnen.*
Enabling with Information passing ($T1[] >> T2$)	*Sobald T1 abgeschlossen ist, wird T2 begonnen, und dabei gibt T1 Informationen an T2 weiter.*
Suspend-resume ($T1| > T2$)	*T2 kann T1 unterbrechen, und wenn T2 abgeschlossen ist, kann T1 von dem vor der Unterbrechung erreichten Zustand aus weitergeführt werden. z.B. Termine bearbeiten und Termine abspeichern.*
Iteration (T^*)	*T wird wiederholt durchgeführt, bis sie von einer anderen Aufgabe deaktiviert wird.*
Finite iteration ($T(n)$)	*T wird n mal wiederholt.*
Optional Task ($[T]$)	*T kann, muss aber nicht, durchgeführt werden. Dies ist häufig beim Ausfüllen von Formularen.*
T Recursion	*T taucht in ihrer eigenen Durchführung auf.*

Tabelle 4.2: Temporale Operatoren in ConcurTaskTrees

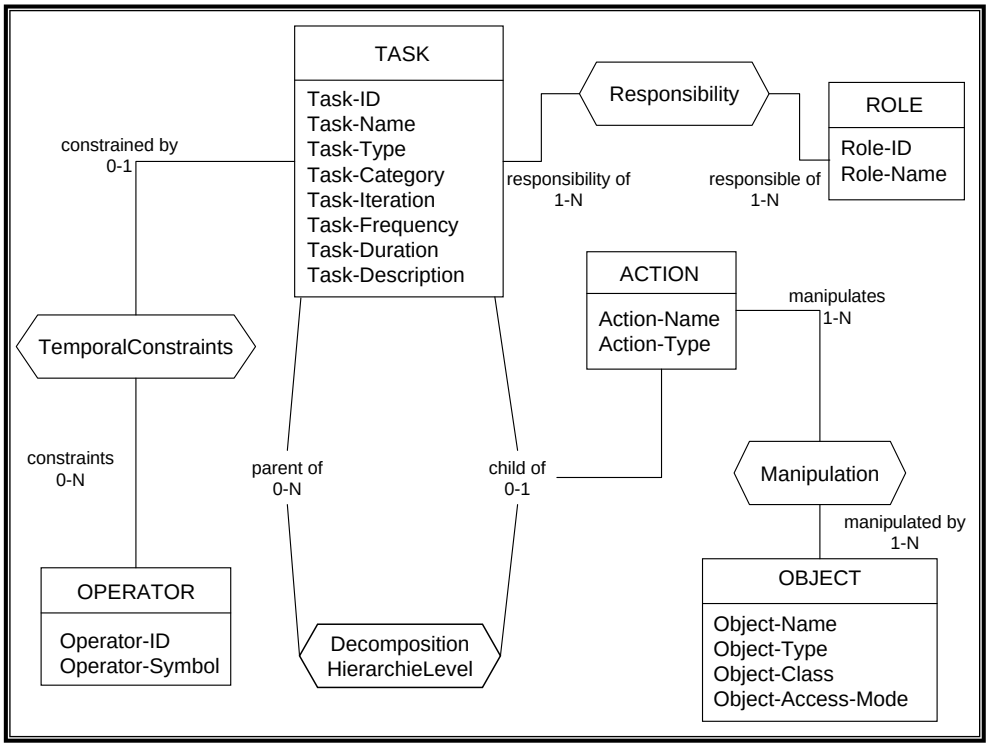

Abbildung 4.10: ConcurTaskTrees Task Meta-Modell ([Limbourg et al., 2001])

1. Fokus auf Aktivitäten: CTT stellt die Aktivitäten, die die Nutzer durchzuführen haben, im Vordergrund und versucht weg von Implementierungsdetails zu abstrahieren.
2. Hierarchische Struktur: Mit ihrer hierarchischen Struktur bietet CTT eine Möglichkeit die Modellierung der Nutzeraktivitäten nah an der menschlichen Intuition zu halten. Beim Problemlösen tendiert der Mensch dazu, das zu lösende Problem in kleinere Teilprobleme zu zerlegen und anschließend einzeln zu lösen, und dabei die Beziehungen zwischen den Teilproblemen und den Teillösungen nicht aus den Augen zu verlieren.
3. Temporale Operatoren: CTT Operatoren (siehe Tabelle 4.2) ermöglichen es, temporale Beziehungen zwischen Aufgaben einer Hierarchie-Ebene zu definieren. Dadurch werden wichtige Informationen, die bei der Analyse von Aufgaben implizit betrachtet werden, explizit ausgedrückt.
4. Kategorisierung der Aufgaben: Bei der Modellierung von Aufgaben mittels CTT besteht die Möglichkeit, Kategorien der Aufgaben festzulegen. Somit wird bestimmt, welche Rolle eine Aufgabe durchzuführen hat. Dabei unterstützt CTT vier Kategorie: *Nutzeraufgabe, Anwendungsaufgabe, Interaktionsaufgaben und abstrakte Aufgabe* (siehe Abbildung 4.11).

Eignung von CTT zur Modellierung der Aufgaben

ConcurTaskTrees finden Verwendung in der Aufgabenanalyse *(Task Analysis)* im RE mit dem Ziel Anforderungen zu identifizieren, welche genügt werden müssen, um be-

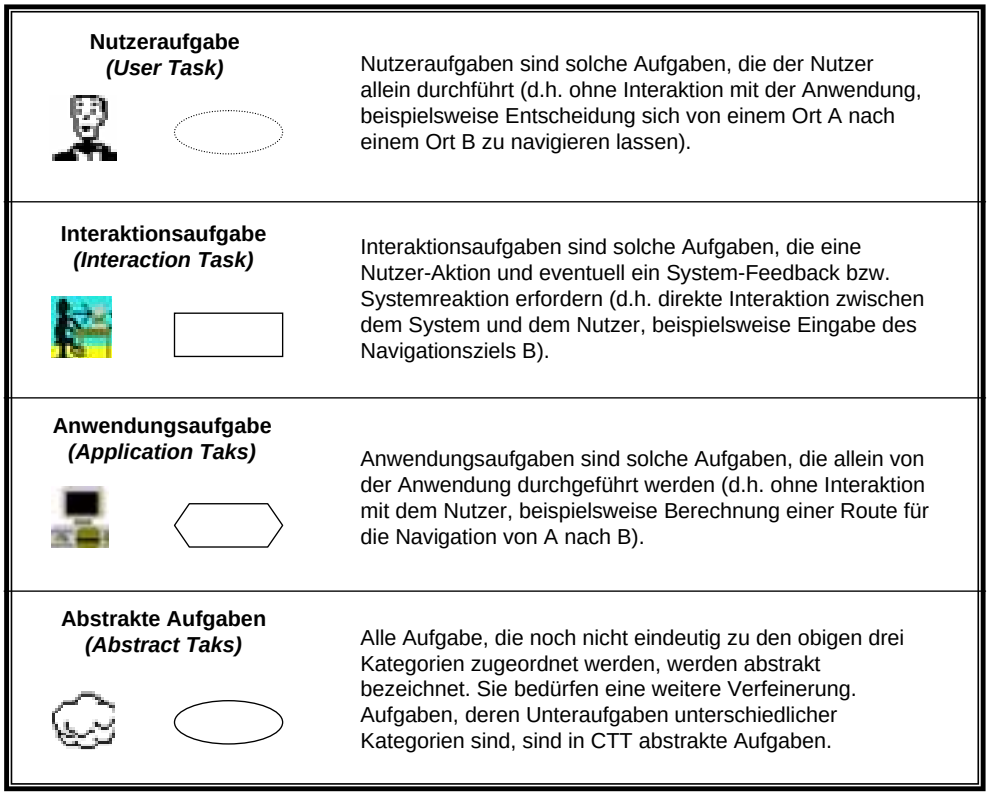

Abbildung 4.11: Kategorien von Aufgaben in ConcurTaskTrees

stimmte Aufgaben durchzuführen bzw. um die Durchführung bestimmter Aufgaben zu unterstützen [van Welie, 2001]. Mit dem besonderen Fokus auf Aktivitäten und ihrer hierarchischen Strukturierung eignet sich CTT zur Modellierung von Aufgaben auf einer konzeptuellen Ebene (Kriterium Kr. 1).

Mittels CTT werden Aktivitäten auf einer sehr intuitiven Art und Weise hierarchisch strukturiert und modelliert. Dies ist nah an der Art und Weise, wie der Mensch Aufgaben löst: um eine Aufgabe zu lösen, tendiert er sie in kleinere Aufgaben zu zerlegen und zu lösen, und hierbei die Beziehungen zwischen den einzelnen Teilaufgaben und Lösungen beizubehalten. Auch, die in CTT verwendeten Konstrukten sind relativ intuitiv und lassen sich leicht erlernen (Kriterium Kr. 2). Bei der Modellierung von Aufgaben, als Teil der Kontextmodellierung, werden einzelne Aufgaben und Teilaufgaben identifiziert und miteinander in Beziehung gesetzt. Dabei sind hauptsächlich an Aufgaben interessiert, die lediglich von den Nutzern, mit oder ohne Interaktion mit der Anwendung, durchgeführt werden. Diesbezüglich unterstützt CTT die Modellierung vier unterschiedlicher Kategorien von Aufgaben (siehe Abbildung 4.11). Auch die hierarchische Strukturierung von Aufgaben ist grundlegend in der Modellierung mittels CTT. Die Definition der Beziehungen zwischen Aufgaben kann allerdings sehr umständlich sein, dafür existiert jedoch eine Werkzeugunterstützung[7]. Die Ausdrucksfähigkeit von CTT ist an dieser Stelle völlig gegeben (Kriterium Kr. 3). Ergebnis der Aufgabenmodellierung ist ein Wald, der aus Bäumen besteht (hierarchische Struktu-

[7] Die Erstellung von Aufgabenmodellen mittels CTT wird mit dem Werkzeug CTTE (ConcurTaskTrees Environment) unterstützt. Wir verweisen auf [Mori et al., 2002, Paternò et al., 2001] für eine detaillierte Beschreibung der Konzepte von CTTE.

rierung der Aufgabe). Die Erweiterung, sowohl der einzelnen Bäume, als auch des gesamten Waldes, lässt sich mittels den Konstrukten von CTT durchführen (Kriterien Kr. 4). Wesentlich für die Aufgabenmodellierung ist die Identifikation von Zielen und notwendigen Aktivitäten sowie ihre Verfeinerung in kleinere Aufgaben, was ebenfalls bei CTT zentraler Natur ist [Mori et al., 2002] (Kriterium Kr. 5). Es gibt eine Reihe von Techniken zur Modellierung von Zielen, Aktivitäten und Aufgaben, die ebenfalls die hier aufgestellten Kriterien grundsätzlich genügen. Der Vorschlag, CTT zu verwenden, wurde jedoch dadurch erleichtert, dass diese Technik und eine Reihe weiterer Techniken zur Modellierung von Aufgaben bereits in [Limbourg et al., 2001] miteinander verglichen wurden. Wichtigste Kriterien, die diesem Vergleich zugrunde liegen, waren die Unterstützung der betrachteten Modellierungstechniken bei Aufgabenplanung und Operationalisierung, die Existenz einer festgelegten untersten Ebene bei der Zerlegung der Aufgaben, und die Spezifikation von Objekten, die bei der Durchführung einer Aufgaben benötigt werden. CTT hat bei diesem Vergleich sehr gut abgeschnitten. Die bei dem genannten Vergleich verwendeten Kriterien sind zwar nicht identisch zu unseren Kriterien, allerdings decken sie die Spezifika von Aufgabenmodellen sehr gut ab (Kriterium Kr. 5).

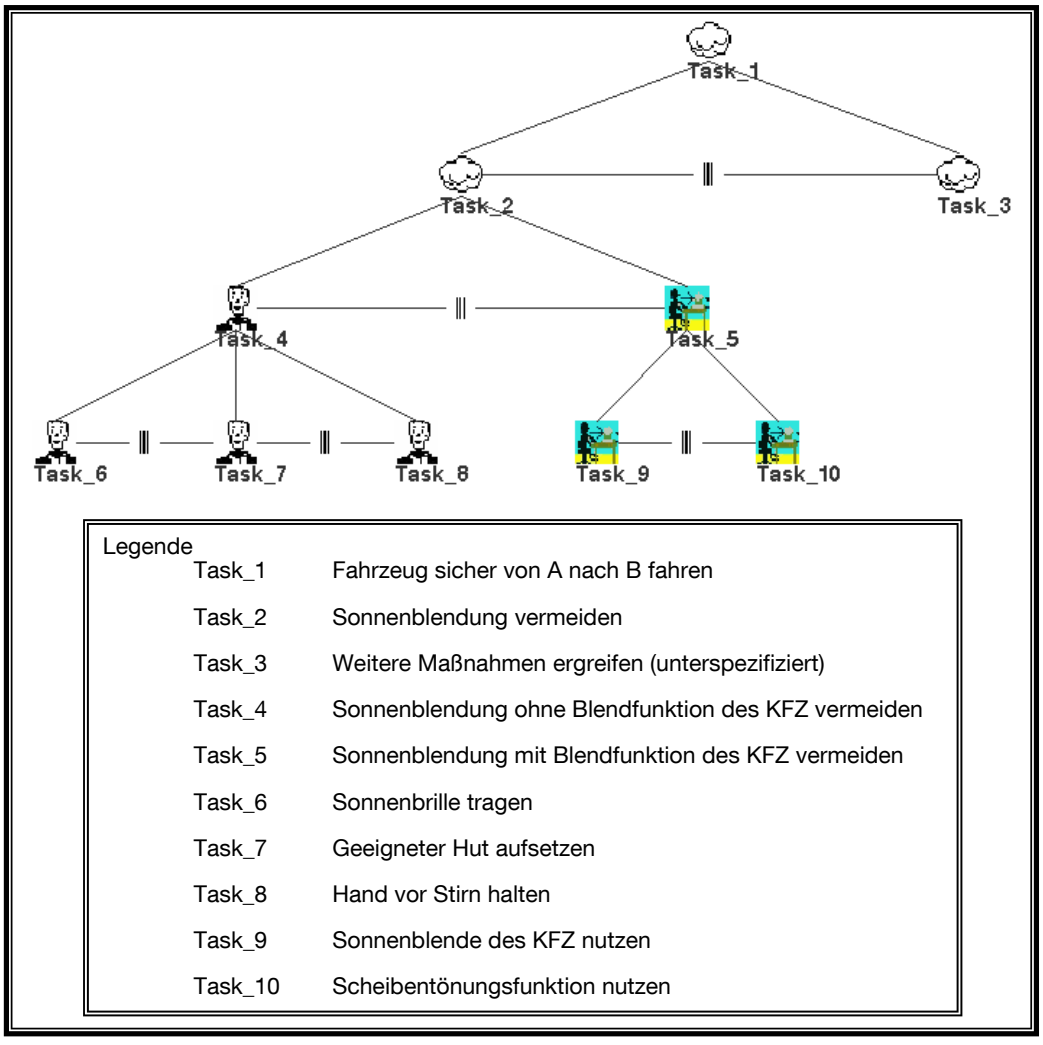

Abbildung 4.12: Ausschnitt des Aufgabenmodells des Scheibentönungssystems

Abbildung 4.12 illustriert die Aufgabenmodellierung mittels ConcurTasktrees anhand der Fallstudie des kontextsensitiven Scheibentönungssystems. An der Wurzel dieses Modellausschnitts steht die Aufgabe *Fahrzeug sicher von A nach B zu fahren*. In diesem Ausschnitt des Aufgabenmodells wird eine Verfeinerung dieser Aufgabe abgebildet. Eine Aufgabe, die aus diese Verfeinerung resultiert, ist die Vermeidung von Sonnenblendung beispielsweise durch Nutzung der Blendfunktionen des Fahrzeugs (Sonnenblende oder Scheibentönungsfunktion). Weitere Aufgaben sind: Sonnenbrille tragen, geeigneter Hut Tragen und Hand vor der Stirn halten.

Das Aufgabenmodell bietet prinzipiell eine gute Möglichkeit zur Integration aller Teilmodelle des Kontextes. Die modellierten Aufgaben werden beispielsweise abhängig von dem verwendeten Plattform durchgeführt, oder ihre Durchführung hängen von der Interaktionsfreiheit des Nutzers ab. Eine solche Interaktionsfreiheit wird in ein Interaktionsmodell abgebildet, und wir adressieren dadurch die Fragestellung, *welche Benutzerinteraktionen werden bei der Durchführung einer Aufgabe benötigt?* Wir kommen auf diese Integrationspunkte später im Abschnitt 4.8 zurück. Zunächst beschäftigen wir uns mit der Modellierung der Einsatzumgebung der Anwendung, d.h. *in welcher Umgebung werden die Nutzer die Anwendung nutzen –, um die modellierten Aktivitäten durchzuführen?*

4.6 Modellierung der Einsatzumgebung

Wir bilden in dem vorliegenden Framework die relevanten Informationen bezüglich der Umgebung, in der die Nutzer ihre Aktivitäten durchführen, in ein Umgebungsmodell ab. In diesem Abschnitt führen wir die in einem solchen Modell zu erwarteten Inhalte auf und schlagen eine Beschreibungstechnik vor, die sich zu ihrer Modellierung eignet.

4.6.1 Informationsgehalt eines Umgebungsmodells

Kontextmodellierung wird in erster Linie als Umgebungsmodellierung verstanden, insbesondere als Modellierung des Einsatzortes einer Anwendung[8]. Mit dem Umgebungsmodell erstellen wir ein Modell, das den Kontextaspekt der Einsatzumgebung widerspiegeln soll. Dabei sind wir insbesondere daran interessiert, Informationen über Objekte in der Einsatzumgebung zu identifizieren, welche für die Adaption der Anwendung von Bedeutung sind.

Die Einsatzumgebung setzt sich grundsätzlich aus zwei nicht notwendigerweise disjunkten Bereiche zusammen: einem Bereich bestehend aus allen für den Nutzer manipulierbaren und/oder beobachtbaren Objekten *(Bereich Nutzer)*, und einem Bereich bestehend aus allen für das System wahrnehmbaren Objekten *(Bereich System)*. Dabei kann der Bereich Nutzer leer sein, und der Bereich System, aufgrund der Notwendigkeit der Informationen über die Einsatzumgebung zur Adaption der Anwendung, jedoch nicht. Bei der Kontextmodellierung ist der Bereich System der Einsatzumgebung möglichst vollständig zu identifizieren und in ein Umgebungsmodell abzubilden. Ab-

[8] In unserem grundlegenden Modell des Kontextes ist die Umgebung, in der die Anwendung verwendet wird, lediglich ein Aspekt des Kontextes.

bildung 4.13 illustriert diese Differenzierung, welche einen methodischen Hintergrund hat. Wir kommen in Kapitel 5 darauf zurück[9]. In diesem Abschnitt sind wir erstmal lediglich daran interessiert, Informationen aus der Einsatzumgebung zu modellieren, welche für die Adaption der Anwendung relevant sind.

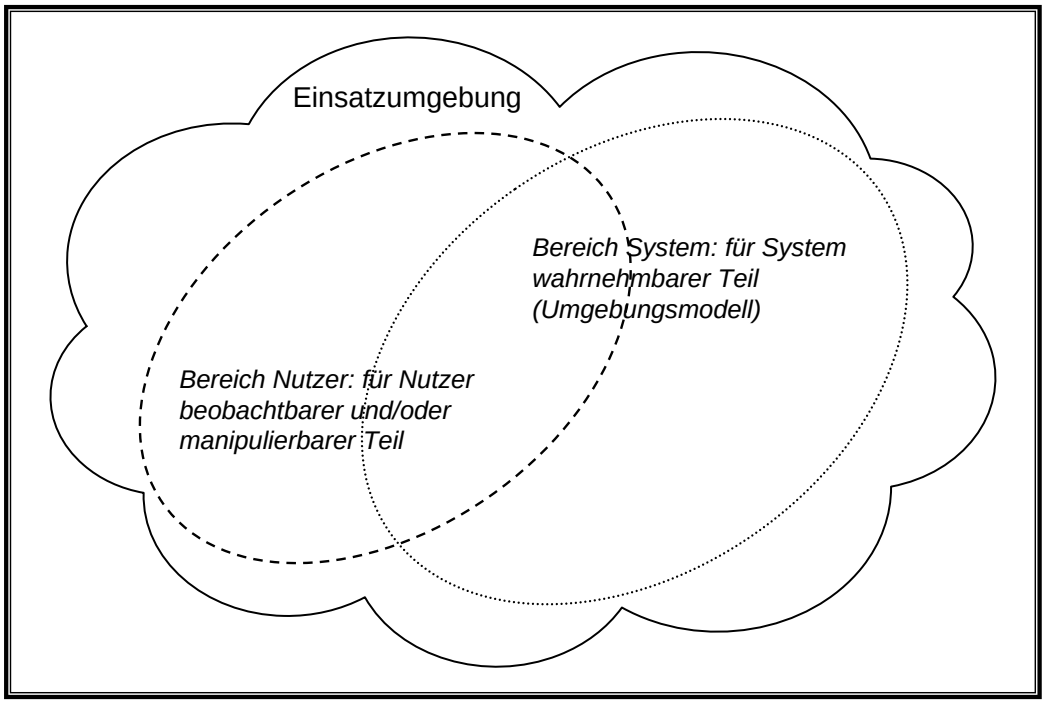

Abbildung 4.13: Bereiche der Einsatzumgebung einer Anwendung

Auf dem Gebiet des Context Aware Computing hat sich [Nicklas, 2005] intensiv mit der Erstellung von Umgebungsmodellen für die Zwecke der Adaption kontextsensitiver Anwendungen beschäftigt. Dabei wird ein Umgebungsmodell wie folgt definiert:

„Ein Umgebungsmodell besteht aus Daten, welche die Umgebung eines Benutzers abbilden. Zusätzlich kann es auch Referenzen oder Metaphern für digitale Informationen enthalten, die für die Umgebung des Benutzers relevant sind. Der Benutzer kann selbst Teil des Umgebungsmodells sein. Mehrere Benutzer können das selbe Umgebungsmodell verwenden.“

Diese Definition stellt die Umgebung des Benutzers in Vordergrund. Dabei werden Informationen bzgl. der Nutzung der Anwendung nicht berücksichtigt, obwohl sie zentral für die Adaption einer kontextsensitiven Anwendung sind. Für die Zwecke der Umgebungsmodellierung als Teil der Kontextmodellierung führen daher die Definition 4.3 ein. Bei dieser Definition bestehen wir darauf, dass es sich um die Umgebung handelt, in der die Anwendung eingesetzt bzw. genutzt wird. Wir sehen es zwar ein, dass der Nutzer Teil der Umgebung ist, allerdings klammern wir ihn bei der Umge-

[9] Wir sind also lediglich an die Inhalte des Umgebungsmodells, d.h. des Bereiches System der Einsatzumgebung, interessiert. Dadurch, dass die Schnittmenge der Bereiche System und Nutzer der Einsatzumgebung nicht notwendiger leer ist, ist es möglich, dass der Nutzer Objekte des Bereiches System manipuliert, um das beobachtbare Verhalten der Anwendung zu beeinflussen: Nutzung der Anwendung über nicht vorhergesehene Kanäle (siehe Abschnitt 3.2). Diese Erkenntnisse sollten beim Entwurf der Anwendung berücksichtigt werden. Auch für die Zwecke der Transparenz der Adaption, ist diese Differenzierung methodisch von wichtiger Bedeutung.

bungsmodellierung aus und verweisen auf seine explizite Betrachtung bei der Nutzermodellierung (siehe Abschnitt 4.4).

Definition 4.3 Umgebungsmodell für kontextsensitive Anwendungen

☞ Ein Umgebungsmodell (auch Modell der Einsatzumgebung) in einer kontextsensitiven Anwendung ist ein Modell, das Annahmen über alle Aspekte der Umgebung enthält, in der die Anwendung eingesetzt bzw. genutzt wird (Einsatzumgebung), welche für die Adaption der Anwendung relevant sind. Es ist ein potentieller Bestandteil des Kontextmodells.

Prinzipiell existiert keine Grenze der Charakterisierung der Einsatzumgebung, allerdings gilt bei der Umgebungsmodellierung die Relevanz der einzelnen Aspekte festzustellen und adäquate Annahmen über sie zu machen. Die Einsatzumgebung einer Anwendung umfasst also Annahme über Aspekte wie Ort der Nutzung der Anwendung, Netzwerkinfrastruktur, Geräte und Kommunikationsmöglichkeiten, Orientierung von Entitäten (Personen oder Gegenstände), physikalische Faktoren (wie Temperatur, Licht, Feuchtigkeit, oder Geräusch). Kontextinformationen der Kategorie Einsatzumgebung sind in erster Linie Ortsinformationen (Position und Lage) von Objekten wie Geräten, Fahrzeugen, Gebäuden, Bäumen, Straßen, Gewässer oder auch Wiesen. Eng verwandt mit der Position eines Objektes ist seine Orientierung, wie es beispielsweise in [Priyantha et al., 2001] aufgeführt ist. Es könnte für die Adaption einer Anwendung wichtig sein, nicht nur die Position oder Lage eines Objektes zu wissen, sondern auch wie es sich orientiert. Auch die physikalischen Eigenschaften der Objekte können charakteristisch für die Einsatzumgebung sein. Beispielsweise kann das Objekt Tunnel in dem Umgebungsmodell aufgenommen werden, und dabei seine Länge die relevante Eigenschaft für die Adaption ist. Weitere Informationen dieser Kategorie sind Zustände der Geräte und Objekte, mit welchen die Nutzer interagieren. Beispiele für solche Geräte sind Notebooks, PDAs oder Mobiltelefone. Mögliche Zustände der Geräte sind funktionsfähig, ein/aus und betriebsbereit (wartend auf Nutzereingabe). Zustände von Objekten wie Gebäude oder Fahrzeuge fallen ebenfalls unter diese Kategorie. Zu den physikalischen Aspekten der Umgebung gehören Faktoren wie Helligkeit, Temperatur, Druck und Geräusch. Auch Kommunikationsinfrastruktur und Interaktionsgeräte werden als relevante Information der Einsatzumgebung klassifiziert. Als Eigenschaften kommen insbesondere die Verfügbarkeit der Geräten (z.B. Mobiltelefone, PDA), ihre Besonderheiten wie Bildschirmgröße/Auflösung, und Besonderheiten der Kommunikationsnetzen (z.B. Bluetooth, GPRS, Wireless LAN) wie ihre Bandbreite in Betrachtung.

4.6.2 Umgebungsmodellierung mittels ER-Modelle

Aus der Definition 4.3 geht hervor, dass ein Umgebungsmodell ein Datenmodell der Einsatzumgebung ist, welches dazu dienen soll, charakteristische Merkmale der Einsatzumgebung bei der Adaption der Anwendung zu berücksichtigen. Bei der Modellierung der Einsatzumgebung sind wir also daran interessiert, für die Adaption relevante Merkmale der Einsatzumgebung festzustellen.

Zur Erstellung von Umgebungsmodellen auf einen konzeptuellen Ebene schlagen wir,

genau wie im Falle von Nutzermodellen, **Entity-Relationship-Modelle** vor. Dieser Vorschlag basiert ebenfalls auf einer Bewertung der E/R-Modellierung anhand der im Abschnitt 4.3 aufgeführten Kriterien. Die Begründung ist ähnlich wie bei der Benutzermodellierung. Wie sie in Abschnitt 4.4.2 aufgeführt wurden, eignen sich E/R-Modelle grundsätzlich für die Modellierung von Daten auf einer konzeptuelle Ebene, was mit dem ersten Kriterium verlangt wird. Bei der Umgebungsmodellierung haben wir in erster Linie Daten über den Einsatzumgebung zu modellieren, damit ist dieses Kriterium erfüllt. Auch hier benötigen wir lediglich die Konzepte von Entitäten, Beziehungen und Attribute der ER-Modellierung. Dadurch ist das Kriterium des Ausdrucksfähigkeit ebenfalls erfüllt (Kriterium Kr. 3). Die Eignung von ER-Modellen hinsichtlich der Einfachheit und Übersichtlichkeit, sowie der Flexibilität und Erweiterbarkeit wird mit den selben Begründungen wie bei der Benutzermodellierung positiv bewertet (Kriterien Kr. 3 und Kr. 4). Das entscheidende Kriterium ist die Eignung zur Modellierung der Spezifika eines Umgebungsmodells (Kriterium Kr. 5). Eine Besonderheit von Umgebungsmodellen ist, dass sie Annahmen über Sachverhalten verschiedener Natur (konkrete Objekte und physikalische Faktoren) enthalten. Die Sachverhalte können miteinander in Beziehung stehen. Bei den Annahmen stellen wir bestimmte Eigenschaften der Sachverhalte in Vordergrund. Wir modellieren die Sachverhalte als Entitäten mit bestimmten Attributen und bei Bedarf mit Beziehungen untereinander.

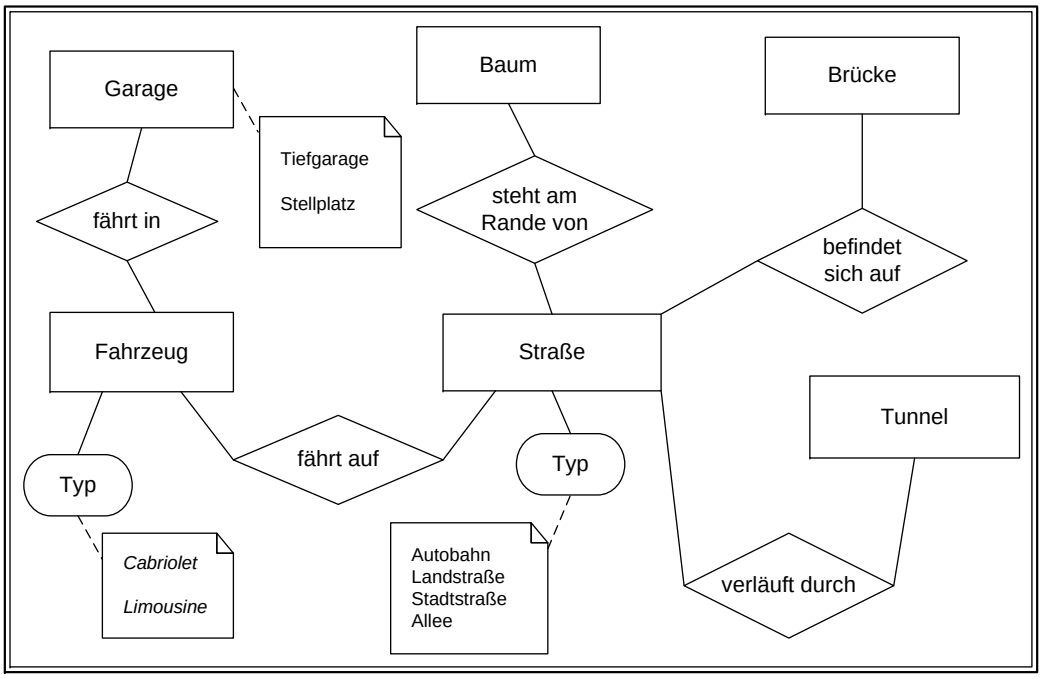

Abbildung 4.14: Ausschnitt des Umgebungsmodells des Scheibentönungssystems

Abbildung 4.14 illustriert die Umgebungsmodellierung mittels Entity-Relationship anhand der Fallstudie des kontextsensitiven Scheibentönungssystems. Zentrales Element in diesem Modell ist die Entität Fahrzeug. In diesem Ausschnitt des Umgebungsmodells wird abgebildet, dass ein Fahrzeug auf einer Straße fährt, welche eine Autobahn, eine Landstraße oder eine Allee sein kann. Am Rande einer Straße können Bäume stehen und eine Straße kann durch Tunnel verlaufen.

4.7 Unterstützende Modelle

Mit der Zielsetzung die Erstellung umfassender Kontextmodelle zu bezwecken, muss das vorliegende Framework flexibel gehalten werden. Dies ist besonders wichtig, damit die Integration neuer Kontextaspekte und die Hervorhebung spezifischer Teilaspekte gewährleistet bleibt.

Die drei zentralen Kontextaspekte Beteiligte, Aktivitäten und Einsatzumgebung sind in dem vorliegenden Framework durch die bislang aufgeführten Modelle abgedeckt worden. Je nach Anwendungsfall können Annahme über zusätzliche bzw. spezifische Aspekte des Kontextes benötigt werden, um nützlichere Adaptionsentscheidungen herbeizuführen[10]. Beispielsweise ist es wichtig im Bereich mobiler webbasierter Anwendungen, Informationen über das Endgerät bei der Adaption der Anwendung zu berücksichtigen [Ziegler et al., 2005]. Dies kann zwar als Teil des Umgebungsmodells betrachtet werden, allerdings wird bei ihrer getrennten Modellierung mit angemessener Sorgfalt vorgegangen. Für diese Zwecke führen wir drei unterstützende Modelle ein: ein Plattformmodell, ein Interaktionsmodell und ein Präsentationsmodell.

Die drei unterstützende Modelle sind je nach Anwendungsfall als Teil des Kontextmodells zu betrachten. Mit diesen Modellen decken wir weitere Aspekte des Kontextes ab, die nicht durch die drei zentrale Teilmodelle des Kontextes explizit abgedeckt bzw. nicht ausführlich genug betrachtet werden können.

4.7.1 Plattformmodell

In kontextsensitiven Anwendungen kann die physikalische Infrastruktur (insbesondere die Geräte), auf der die Anwendung läuft eine entscheidende Rolle für die Adaptionsentscheidung spielen. Wenn die Anwendung mit variablen physikalischen Infrastrukturen genutzt wird, wird besonders wichtig, Informationen bzgl. den unterschiedlichen Infrastrukturen bei den Adaptionsentscheidungen zu berücksichtigen. Dafür bilden wir alle relevanten Informationen bzgl. den physikalischen Infrastruktur in ein Plattformmodell ab.

Die Bedeutung eines Plattformmodells ist in der Literatur unumstritten. Das Modell wird teilweise als *Computing Context* ([Schilit et al., 1994], [Dey, 2000]), *Infrastructure Context* ([Rodden et al., 1998], [Dix et al., 2000]) oder *Device Context/Profile* ([Houssos et al., 2003, Lemlouma und Layaida, 2004, Ziegler et al., 2005]) klassifiziert. Mit Plattformodellen soll der Tatsache Rechnung tragen werden, dass mobile kontextsensitive Systeme grundsätzlich über eingeschränkte Ressourcen verfügen. Auch die Tatsache, dass die Anwendungen meist in einer verteilten Umgebung genutzt werden und die Nutzung den Einsatz einer Kette von Geräten beanspruchen kann, ist bei der Modellierung der Plattform zu berücksichtigen.

Sachverhalte, die in ein Plattformmodell abgebildet werden, sind im Wesentlichen charakteristische Informationen über die physikalische Infrastruktur inklusive den Bezie-

[10] Die zusätzlichen Informationen, die diese Modelle enthalten, können teilweise in das Aufgabenmodell (z.B. die notwendige Interaktionen zur Durchführung der Aufgaben) oder das Umgebungsmodell (z.B. physikalische Infrastruktur) abgebildet werden. Allerdings ist es hilfreich diese Informationen in andere Modelle abzubilden. Dadurch wird die Gefahr vermindert, dass relevante Informationen für die Adaptionsentscheidung vernachlässigt werden.

hungen zwischen den verwendeten Geräten. Wesentliche Annahmen über die Geräte werden in einem Plattformmodell abgebildet. Diese Annahmen betreffen zum Beispiel der Typ der verwendeten Geräte (Desktop PC, PDA, Mobiltelefon ...), die Größe und Auflösung des Display, sowie die unterstützten Kommunikationsmöglichkeiten.

Für die Plattformmodellierung schlagen UML-Komponentendiagramme [OMG UML Partners, 2005, Rumbaugh et al., 2004] vor[11]. Ein Grund für diesen Vorschlag ist, dass UML-Komponentendiagramme grundsätzlich für die Modellierung technischer Komponente, die zur Ausführungszeit benötigt werden, eingesetzt werden können. Mit dem UML-Komponentendiagramm können wir die verschiedensten Teile einer Plattform und ihre Beziehungen unter einander modellieren. Dabei benötigen wir die Konzepte von Komponenten und Schnittstellen. Jede Komponente benötigt bestimmte Schnittstellen und bietet welche an.

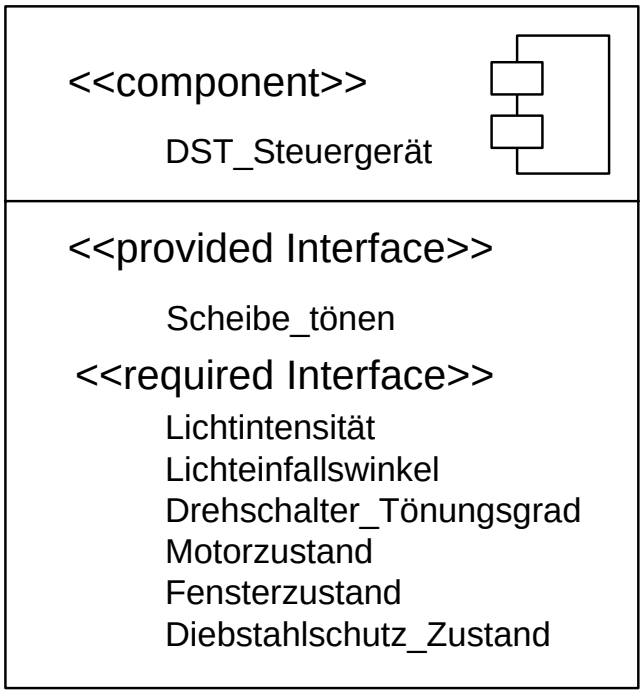

Abbildung 4.15: Ausschnitt des Plattformodells des Scheibentönungssystems

Abbildung 4.15 illustriert die Plattformmodellierung mittels UML-Komponentendiagramm anhand der Fallstudie der kontextsensitiven Scheibentönung. In diesem Ausschnitt des Plattformmodells wird abgebildet, dass das Scheibentönungssteuergerät die Lichtintensität bei dem Helligkeitssensor abfragt. Im Fall des Scheibentönungssystem ist die das Plattformmodell als Teil des Kontextmodell nicht sehr relevant, da das Plattform fest ist. Bei dem kontextsensitiven Terminplaner wird die Bedeutung des Plattformmodell höher. Dort wird nämlich zwischen Clients unterschieden, die auf Pocket PC laufen und welche, die auf Desktop PC laufen. Je nach dem welches Plattform gerade verwendet wird, werden angemessene Adaptionsentscheidungen getroffen.

[11] Grundsätzlich würden jedoch auch strukturierte Tabelle oder Entity-Relationship-Modelle (Bestandteile der Plattform als Entitäten mit Beziehungen unter einander und mit bestimmten Eigenschaften als Attribut) zur Modellierung der Plattform passen.

4.7.2 Interaktionsmodell

Zur Nutzung einer Anwendung finden zwischen dem Nutzer und dem System eine Reihe von Interaktionen statt. Im Fall kontextsensitiver Anwendungen können die notwendigen Interaktionen zur Durchführung einer Aufgabe situationsabhängig sein. Beispielsweise in einer lauten Umgebung ist es angebrachter, eine Telefonnummer über die Tastatur zu wählen als über die Spracheingabe. Der Dialog zwischen dem Nutzer und dem System wird in einem Interaktionsmodell/Dialogmodell abgebildet.

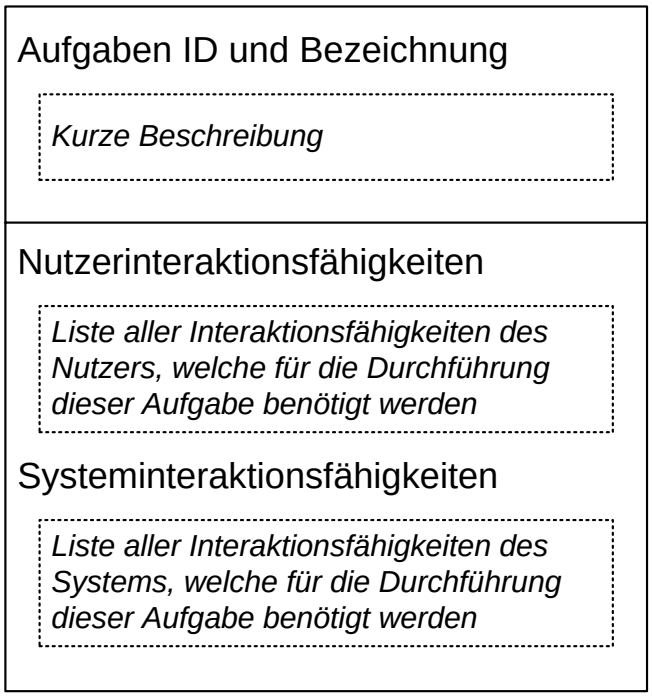

Abbildung 4.16: Template zur Erstellung von Interaktionsmodellen

In einem Aufgabenmodell sind eine Reihe von Aktivitäten abgebildet worden, die der Nutzer in Zusammenarbeit mit dem System durchführt (so genannten Interaktionsaufgaben). Zur Erstellung des Interaktionsmodell ist der Fokus auf diese Aufgaben zu setzen. Die Möglichkeiten der Partner (System und Nutzer), diese Aufgaben kooperativ durchzuführen, werden modelliert. Bei der Interaktionsmodellierung ist es daher wichtig, die Interaktionen sowohl aus Sicht des Nutzers als auch aus Sicht des Systems zu betrachten und dabei ihre Interaktionsfähigkeiten zu beachten. Die Betrachtung der Interaktionen aus Sicht des Systems, setzt allerdings voraus, dass ein erstes Systemkonzept (erster Entwurf) vorhanden ist.

Es gibt eine Tendenz Aufgabenmodelle und Interaktionsmodelle miteinander zu integrieren [Luyten et al., 2003]. Dies ist grundsätzlich nicht falsch. Allerdings besteht die Gefahr gewisse Interaktionen zu übersehen, und dadurch Adaptionsentscheidungen zu treffen, welche aufgrund der momentanen Interaktionsfähigkeit des Nutzers nicht angebracht sind. Eine Trennung der Modelle zielt darauf, die benötigten Interaktionen zur Durchführung einer Aufgaben explizit zu modellieren, und die Interaktionsfreiheit des Nutzers gezielt bei der Adaptionsentscheidung zu berücksichtigen.

Zur Erstellung der Interaktionsmodelle schlagen wir lediglich strukturierte Tabelle

vor. In einer solchen strukturierten Tabelle listen wir für jede Aufgabe, die bei der Aufgabenmodellierung als „Interaktionsaufgabe“ identifiziert wurde, die Interaktionen sowohl aus Sicht des Nutzers als auch aus Sicht des Systems auf. Abbildung 4.16 zeigt ein Template zur Modellierung möglicher Interaktionen, die notwendig für die Durchführung einer Aufgabe sind.

4.7.3 Präsentationsmodell

Für die Interaktionen zur Durchführung bestimmter (Interaktions-) Aufgaben können kontextsensitive Anwendungen je nach Anwendungsfall unterschiedliche Darstellungen der Interaktionselemente bzw. -Objekte verwenden. Die Objekte werden während der Durchführung der Aufgaben manipuliert.

Mit dem Präsentationsmodell (auch Darstellungsmodell) wird ein ähnliches Ziel verfolgt, wie im Falle des Interaktionsmodells. Für jede Interaktionsaufgabe aus dem Aufgabenmodell sind Objekte zu identifizieren, die bei der Durchführung dieser Aufgabe manipuliert werden. Wir konzentrieren uns auf Bedienelemente des Systems, sowie Objekte, die von System verwendet werden, um dem Nutzer Feedback bzgl. der Interaktion zu geben. Sie sind visuelle, haptische und audio-Elemente, welche mit der Durchführung einer Aufgabe betroffen werden können. Die Betrachtung der Interaktionsobjekte setzt voraus, dass ein erstes Systemkonzept vorhanden ist.

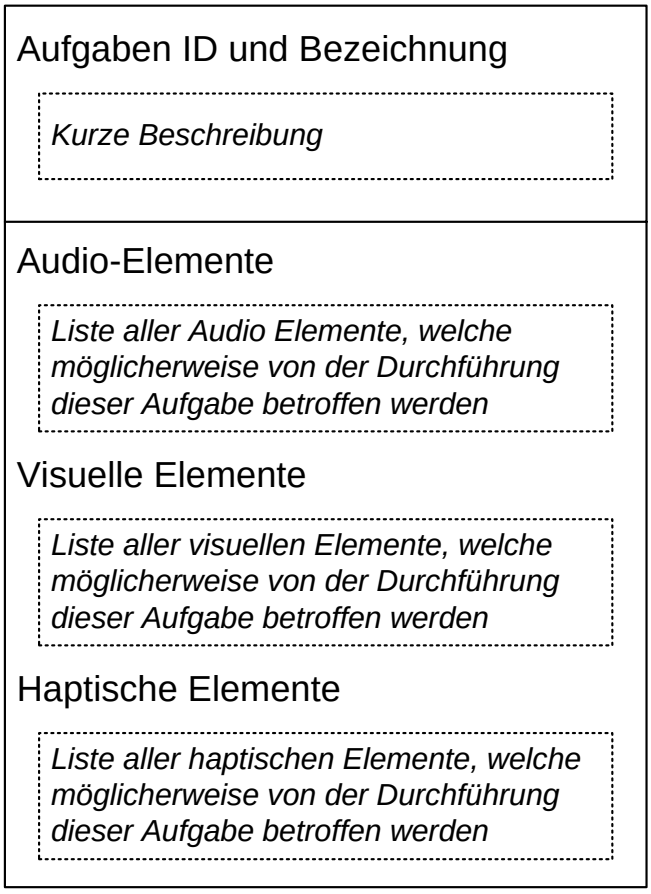

Abbildung 4.17: Template zur Erstellung von Präsentationssmodellen

Zur Erstellung der Präsentationsmodelle schlagen wir ebenfalls strukturierte Tabelle vor (siehe Abbildung 4.17). Darin listen wir für jede Aufgabe, die bei der Aufgabenmodellierung als Interaktionsaufgabe identifiziert wurde, die Objekte auf, die bei ihrer Durchführung betroffen werden können.

Die bislang aufgeführten Modelle der Teilaspekte des Kontextes sollen dazu dienen ein umfassendes integriertes Kontextmodell im RE zu erstellen. Dabei wird kein Anspruch auf ihre Eignung zur unmittelbaren technischen Realisierung eines Kontextmodells erhoben. Aufgrund der Tatsache, dass die Menge der vorstellbaren kontextsensitiven Anwendungen riesig ist, kann kein universales Kontextmodell geben, welches für jede Zwecke passend ist. Nichtsdestotrotz soll mit den beschriebenen Modellen ermöglicht werden, die häufig genannten Aspekte des Kontextes abzudecken. Ein übergreifendes Kontextmodell soll die aufgeführten Modelle integrieren, und sie bei Bedarf auf spezifische Forderungen der Anwendung zuschneiden. Im nächsten Abschnitt führen wir einen Ansatz zur Integration der Modelle auf.

4.8 Integration in ein umfassendes Kontextmodell

Kontextmodellierung im Requirements Engineering zielt darauf, Nutzungssituationen einer Anwendung zu identifizieren und sie in ein umfassendes Kontextmodell abzubilden. Dabei ist jede Nutzungssituation eindeutig durch eine Belegung des Kontextmodells gekennzeichnet[12]. Die Menge aller Ausprägungen des Kontextmodells bildet dann die Gesamtheit der Situationen, in denen die Anwendung genutzt wird.

Wie im Abschnitt 4.2.1 aufgeführt ist, sind die einzelnen Aspekten des Kontextes miteinander in Wechselwirkung (die so genannten Inter-Aspekt Wechselwirkung). Demnach werden sie miteinander in Beziehung gesetzt und die einzelnen Nutzungssituationen der Anwendung dadurch charakterisiert. Wir betrachten die Nutzungssituationen als Instanzen des Kontextmodells, welches sich aus den in den Abschnitten 4.4, 4.5, 4.6 und 4.7 aufgeführten Modellen zusammen setzt. Bislang haben wir jedoch nicht aufgeführt, wie diese Modelle miteinander integriert werden.

4.8.1 Fakten in einer Situation als Mittel der Integration

Eine Möglichkeit zur Integration der Modelle ist der Einsatz technischer Lösungen auf Ebene der Realisierung. Aufgrund unserer Zielsetzung, die Kontextmodelle im RE einzusetzen – um zu analysieren, welche Funktionalität einer Anwendung, und dementsprechend welche Anforderungen, in einer Nutzungssituation angebracht ist bzw. erfüllt werden soll – hilft uns eine Integration der Teilmodelle erst auf einer Realisierungsebene kaum. Es ist wichtig, dass die Integration der Modelle auf einer ähnlichen konzeptuellen Ebene wie ihre Erstellung erfolgt. Über die Nutzungssituationen der Anwendung haben wir eine Möglichkeit diese Forderung zu genügen, mit dem Vorteil nah genug an unserer Zielsetzung im RE zu sein.

Nutzungssituationen werden grundsätzlich durch Fakten über die Nutzung der Anwendung zu einer gegebenen Zeit charakterisiert. Fakten sind elementare Aussa-

[12] Wir betrachten eine Situation als Schnappschuss der Modelle inkl. den Wechselbeziehungen zwischen ihnen.

gen, die in einem Modell, beispielsweise dem des Kontextes einer kontextsensitiven Anwendung, gelten. Sie bekommen einen Zeitstempel, womit eindeutig festgelegt wird, wann sie gelten. Die Sammlung der vorhandenen Fakten wird beispielsweise in der Künstlichen Intelligenz als Faktenbasis bezeichnet [Simon, 1969, Russell und Norvig, 2003]. Alle in einem Faktenbasis enthaltenen Fakten sind gültig, allerdings nicht zwangsläufig gleichzeitig. Basierend auf einem solchen Faktenbasis kann eine Reihe von Regeln definiert werden, welche eine unmittelbare Nachbildung des menschlichen „Wenn ... dann...“-Schließens erlauben. Dies ist tatsächlich das, was in Adaptionsentscheidungen abgebildet wird, welche als Logik der Adaption realisiert werden – Adaptionslogik. Die Fakten enthalten Informationen über die Nutzungssituationen der Anwendung. Ausgehend von diesen Informationen bestimmt die Adaptionslogik, welche Funktionalität der Anwendung in einer Situation angebracht ist. Mit Fakten beschreiben wir übergreifende Zusammenhängen zwischen den einzelnen Modellen und erzielen dadurch ihre Integration.

Die betrachteten Fakten sind der Form: ***Nutzer*** führen ***Aktivitäten*** unter bestimmten ***Umgebungsbedingungen*** durch. Bei der Durchführung der Aktivitäten kann ein System – das zu entwickelnde kontextsensitiven System – genutzt werden. Die Durchführung der Aktivität führt dazu, dass ***Interaktionen*** stattfinden. Dabei werden eine Reihe von Objekten manipuliert. Die Objekte können sowohl Teil der Umgebung, als auch Teil des unterstützenden Systems sein. Die Objekte, die Teil des Systems sind, werden für die ***Darstellung*** der Kommunikation zwischen dem Nutzer und der Anwendung verwendet. Die Anwendung läuft auf einem gegebenen ***Plattform (Infrastruktur)***. Die Modellierung des Nutzers, der Aktivitäten, der Umgebung, der Interaktionen, der Darstellungsmöglichkeiten und der Infrastruktur ist in den Abschnitten 4.4, 4.5, 4.6 und 4.7 aufgeführt.

Die Fakten widerspiegeln Wechselwirkungen zwischen diesen Teilmodellen des Kontextes. Sie bilden grundlegende Beziehungen zwischen den einzelnen Modellen. Die Informationen über die Beziehungen bzw. Wechselwirkungen sind charakteristisch für die Nutzungssituationen. Als grundlegende Beziehungen zwischen Nutzern und Einsatzumgebung gelten beispielsweise Beziehungen zwischen Arbeiter und Arbeitsstelle, Student und Universität, sowie Fahrer und Fahrzeug. Beispiel einer Beziehung zwischen Nutzer und Aktivität ist die Beziehung zwischen Wissenschaftler und Forschen oder auch Fahrer und Fahren.

Eine Abstraktion der Sammlung aller Fakten ergibt die Menge der Fakttypen, aus den das Faktenmodell besteht. Ein solches Faktenmodell fungiert als Kontextmodell der Anwendung. Jede Instanz des Kontextmodells charakterisiert eine Nutzungssituation. Die möglichen Inhalte einer Instanz des Kontextmodells sind demzufolge die Sammlungen aller Fakten. Sie beschreiben die Informationen über den Nutzer (Nutzerdaten und Nutzungsdaten), seine aktuell durchgeführten Aktivitäten, die aktuelle Umgebung, in der er die Anwendung nutzt bzw. seine Aktivitäten durchführt, die Interaktionen, die er benötigt, um seine Aktivitäten durchzuführen, die Plattform, auf der die Anwendung läuft, sowie die aktuellen Objekte zur Repräsentation der Interaktion zwischen dem System und dem Nutzer. Mit Hilfe des Kontextmodells stellt das System fest, welche Nutzungssituation vorliegt und zeigt entsprechendes Verhalten bzw. erfüllt abhängig davon definierte Anforderungen[13]. Abbildung 4.18 illustriert das

[13]Unter Umstände können die bestehenden Kontextinformationen nicht für die Entscheidung ausreichen, welches Verhalten das System zeigen soll. In solchen Fällen ist es notwendig, zusätzliche Kontex-

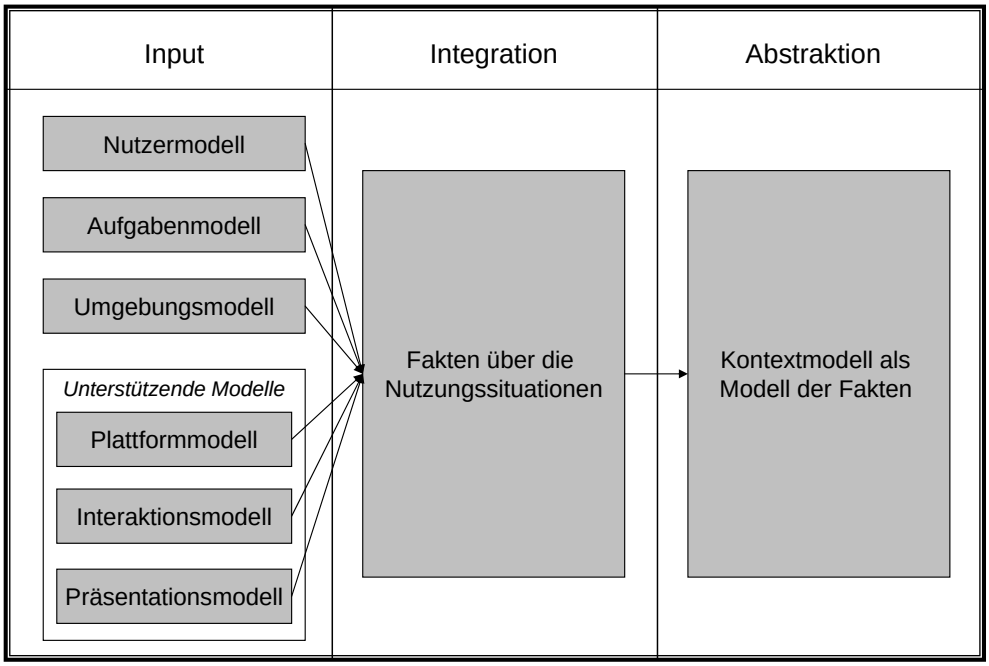

Abbildung 4.18: Kontextmodellierung: Integration über Fakten

Grundkonzept der Kontextmodellierung. Dieses Konzept ist insbesondere für die Erstellung des Kontextmodells einer Anwendung mit Hilfe der RE-CAWAR Methodik (siehe Kapitel 5) grundlegender Natur.

4.8.2 Faktenmodellierung mittels Object-Role-Modeling

Eine bekannte Technik zur Modellierung von Fakten – und zur Analyse von Informationen auf einer konzeptuellen Ebene – ist das *Object-Role Modeling (ORM)*. Sie wurde Mitte der 1970er Jahren eingeführt [Falkenberg, 1976], und seither für unterschiedliche Zwecke überarbeitet. Ein Überblick über ORM ist in [Halpin, 1996, Halpin, 1998] gegeben, und eine detaillierte Beschreibung seiner Grundkonzepte ist in [Halpin, 2001] gegeben. ORM wird hauptsächlich auf einer konzeptuellen Ebene in der Datenmodellierung verwendet, und beschreibt Informationen über Objekte in Form von Fakten. Dabei spielt die Fragestellung, wie Fakten in Strukturen wie Objektklassen gruppiert werden, keine Rolle. Viel mehr geht es in ORM darum, die Fragestellung zu adressieren, wie Entwickler und Experten der Anwendungsdomäne ihre Kenntnisse miteinander sinnvoll beschreiben, kommunizieren und validieren können. Mittels des relationalen Mappings von ORM können die erstellten Faktenmodelle unmittelbar in Datenbanken umgesetzt werden[14].

Faktenorientierung in ORM bedeutet, dass Beispiele von Datensätzen, den so genann-

tinformationen, wie etwa die Historie der Fakten, zu betrachten. Die Historie der Fakten ist dann als eigenständige Kontextinformation zu modellieren und in die Bedingungen der Entscheidungsregeln aufzunehmen.

[14] Die Abbildung der Faktenmodellen auf relationalen Datenbanken geht über die Ziele der vorliegenden Arbeit hinaus. Für eine tiefer gehende Beschreibung der ORM-Konzepte, inkl. des relationalen Mapping, verweisen wir auf [Halpin, 2001].

ten *Data Use Cases*, als elementare Fakten in natürlich sprachlichen Sätzen ausgedrückt werden. Die Beschreibungen werden durch eine relativ intuitive graphische Notation unterstützt. Die Validierung der Faktenmodelle erfolgt mittels Beispiele von Datensätzen. Eine bekannte Variante von ORM ist das *Formal ORM (FORM)* [Halpin, 2001], mit den Konzepten von *Fakttyp, Constraint und Ableitungsregel*. Eine ausführliche Diskussion dieser Konzepten fällt außerhalb der Rahmen dieser Arbeit. Wir verweisen für diese Zwecke auf [Halpin, 2001] und erklären im Folgenden die grundlegenden Konzepte von ORM.

Grundlegend in ORM sind die Konzepte von *Rollen (roles)*, *Objekte (objects)* und *Fakttypen (fact types)*. Objekte und Rollen werden in Fakten miteinander in Beziehung gesetzt. Ein Fakt in diesem Zusammenhang ist eine Aussage darüber, dass ein Objekt eine bestimmte Eigenschaft hat (d.h. es spielt eine Rolle mit sich selbst) oder sich ein oder mehrere Objekte an einer Beziehung beteiligen (d.h. sie spielen eine von einander unterschiedliche Rolle in einer Beziehung) und spielen darin ihre jeweiligen Rollen. Mit Fakttypen werden Fakten der selben Kategorie gruppiert (d.h. Fakten sind Instanzen von Fakttypen). Fakttypen bestehen also aus einer oder mehreren von einander unterschiedlichen Rollen. Die Anzahl der Rollen in einem Fakttyp wird Stelligkeit (*engl. arity*) des Fakttypen genannt. Jede Rolle ist mit einem Objekt assoziiert, welches durch einen Name und eine Referenz charakterisiert ist. Fakttypen werden jeweils mit einer eindeutigen Prädikaten versehen.

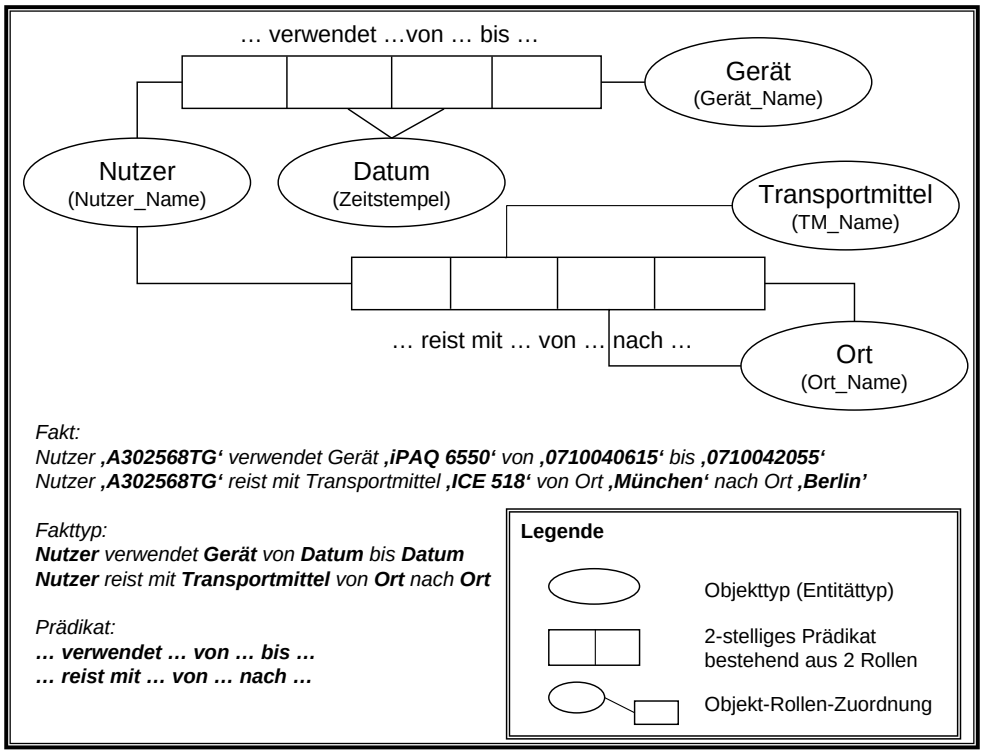

Abbildung 4.19: Beispiel eines ORM-basierten Faktenmodells

Abbildung 4.19 illustriert ein einfaches ORM Faktenmodell. Das Modell bildet zwei Fakttypen ab, welche eine Gruppierung unterschiedlicher Fakten darstellen.

Ausgehend von dieser kurzen Einführung in ORM, lässt sich diese Technik hinsicht-

lich ihrer Eignung zur Modellierung von Fakten im RE kontextsensitiver Anwendungen anhand der Kriterien aus Tabelle 4.1 relativ schnell bewerten. Wie bereits zu Beginn dieses Abschnittes erwähnt, wird ORM auf einer konzeptuellen Ebene in der Systementwicklung eingesetzt, u.a. mit dem Ziel die Kommunikation zwischen Experten der Anwendungsdomäne und Entwickler zu erleichtern. Demnach ist das erste Kriterium (Modellierung auf einer konzeptuellen Ebene) erfüllt. ORM-Konstrukte, und selbst die seiner neueren Erweiterungen wie dem *Context Modeling Language* [Henricksen, 2003, Henricksen et al., 2005], können relativ schnell erlernt werden. Die erstellten Modelle sind ähnlich ER-Modelle, und ihre Übersichtlichkeit ist relativ gegeben (Kriterium Kr. 2). Für unsere Zwecke der Integration der Teilmodelle, reicht es, wenn wir die Wechselwirkungen zwischen den unterschiedlichen Aspekten des Kontextes beschreiben können, die in den Modellen abgebildet werden. Mittels Prädikaten zwischen den Entitätstypen in ORM-Modelle lassen sich diese Wechselwirkungen abbilden. Die Ausdrucksfähigkeit von ORM ist somit gegeben (Kriterium Kr. 3). ORM-Modelle lassen sich mit der Einführung neuer Fakttypen und/oder Erweiterung bestehender Fakttypen um weitere Entitätstypen erweitert. Lediglich die Einführung entsprechender Rollen und Prädikate ist notwendig. Allerdings wird abgeraten, mehr als vier Rollen in einem Fakttypen abzubilden. Trotzdem ist die Erweiterbarkeit von ORM-Modellen gegeben (Kriterium Kr. 4). Als Besonderheit von Situationen (Kriterium Kr. 5) gilt die Tatsache, dass ihre Charakterisierung unterschiedliche Aspekte des Kontextes gleichzeitig betreffen kann. Diesem Umstand wird durch die Modellierung der Situationen als (Teilmodell-)übergreifende Fakten Rechnung getragen.

4.8.3 Aktualisierung des Kontextes

Das Kontextmodell, wie es in diesem Kapitel aufgeführt ist, soll dazu dienen, die Nutzungssituationen der Anwendung anhand vorliegender Kontextinformationen zu identifizieren und abzubilden. Eine offene Frage bleibt die Fragestellung der Aktualisierung des Kontextes, die bislang aufgrund der Zielsetzungen dieser Arbeit bewusst in den Hintergrund gestellt wurde. Die Aktualisierung des Kontextes betrifft hauptsächlich die Aktualisierung der Kontextinformationen, allerdings ist dabei die Aktualisierung des eigentlichen Kontextmodells (beispielsweise aufgrund einer Kalibrierung [Fahrmair, 2005, Fahrmair et al., 2006a]) nicht auszuschließen. In diesem Abschnitt führen wir auf, wie die Aktualisierung von Kontextinformationen erfolgen kann.

Wie sie bislang aufgeführt sind, dienen Kontextinformationen der Charakterisierung von Nutzungssituationen einer Anwendung. Dabei stellt eine Nutzungssituation den Zustand der realen Welt zu einer gegebenen Zeit oder über ein Zeitintervall. Sie wird mit einem oder mehreren Fakten über die reale Welt charakterisiert. Um die Nutzungssituationen strukturiert zu erfassen, haben wir ein Kontextmodell konzipiert, welches die unterschiedlichen Aspekten der realen Welt der Nutzung einer Anwendung abdecken. Demnach stellt jede Instanz des Kontextmodells eine Nutzungssituation dar. Die Nutzungssituationen bezogen auf der realen Welt bleiben allerdings nicht den gesamten Lebenszyklus der Anwendung lang unverändert. Sie müssen daher aktuell gehalten werden. Die Aktualisierung der Nutzungssituationen wird in diesem Zusammenhang gleich der Aktualisierung der Kontextinformationen gesetzt.

Im Bereich des Context-Aware Computing wird die reale Welt mit physischen

und logischen Sensoren überwacht, welche die notwendigen Daten zur Bestimmung von Kontextinformationen liefern [Schilit et al., 1994, Dey, 2000, Schmidt, 2002, Samulowitz, 2002, Fahrmair, 2005]. Um die Aktualisierung der Kontextinformationen effizient zu gestalten, wurde eine Schicht-Architektur im Esprit Project 26900 – TEA ([Esprit Project 26900, 1998, Schmidt et al., 1999b, Schmidt, 2002]) eingeführt. Abbildung 4.20 illustriert diese Architektur. Dabei wird zwischen den Abstraktionsebenen *Sensor, Cue und Kontext* unterschieden.

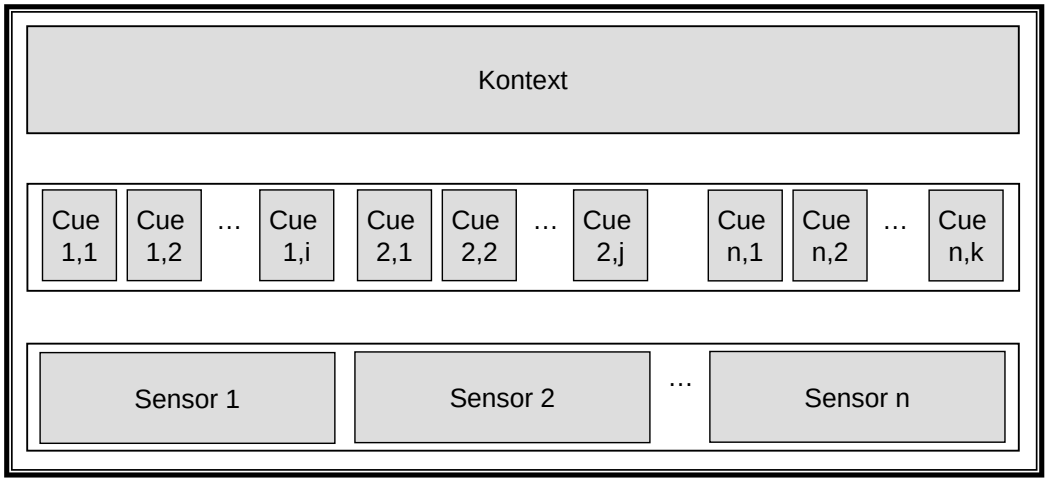

Abbildung 4.20: Konstruktion und Aktualisierung von Kontextinformationen

Auf der Sensor-Ebene befinden sich sowohl so genannte physische als auch logische Sensoren. Physische Sensoren bezeichnen Hardwareeinheiten (wie Licht- und Temperatursensoren), die eine bestimmte physikalische Größe der Umgebung messen. Logische Sensoren sind Einheiten, welche Informationen liefern, die nicht unmittelbar aus der Umgebung entnommen sind, aber Informationen über die reale Welt darstellen (z.B. Uhr als Sensor, der die Uhrzeit liefert, oder ein Server, der den Wechselkurs von Währungen liefert).

Auf der Cue-Ebene werden von den physischen und logischen Sensoren abstrahiert. Jede Cue hängt genau von einem Sensor ab, aber mittels den von einem Sensor gelieferten Daten können mehrere Cues berechnet werden (1:n Beziehung zwischen Sensoren und Cues). Sie sind ein Mittel zur Reduzierung der Datenmenge, die von den Sensoren produziert werden. Dabei abstrahieren sie von unnötigen Details der Sensoren. Somit ermöglichen sie auch den Austausch von Sensoren, ohne dass Änderungen auf der Kontext-Ebene erforderlich wären. Im Rahmen des TEA-Projekt sind Funktionen vorgeschlagen, welche eingesetzt werden können, um die anfallenden Daten zusammenzufassen und notwendige Kontextinformationen daraus abzuleiten. Die Wahl der Funktionen hängt von der geplanten Anwendung bzw. der eingesetzten Sensoren ab. Zur Charakterisierung einer Messfolge kann beispielsweise kontinuierlich ein gleitender Mittelwert gebildet werden. Alternativ zum Mittelwert kann die erste Ableitung (zur Ermittlung von Änderungen) oder die Standardabweichung (zur Bestimmung der Stabilität eines Signals) berechnet werden.

Auf der Kontext-Ebene befinden sich alle Beschreibungen von Kontextinformationen, die für die Anwendung relevant und modelliert sind. Bei der Konstruktion und Aktualisierung der Kontextinformationen werden die Cues eingesetzt, um Beobachtungs-

daten (d.h. von den Sensoren gesammelte Daten) vorab zu verarbeiten. Aus den resultierenden Daten werden dann Kontextinformationen zur Beschreibung der Situationen schrittweise abgeleitet und in dem Kontextmodell fortgeschrieben. Situationen, die auf das Kontextmodell abgebildet werden können, werden als Nutzungssituationen bewertet und entsprechend gehandelt. In einem System, das durch eine geringe Anzahl von Sensoren bzw. zu bestimmender Fakt-/Kontexttypen gekennzeichnet ist, kann die Ableitung von Kontextinformationen durch einfache Regeln gefasst werden.

Eine ähnliche Architektur findet man beispielsweise auch in [Samulowitz, 2002] und [Fahrmair, 2005] wieder. In [Samulowitz, 2002] wird von Sensoren/Signalgeber, Interpretationsmodell und Situation/Kontext geredet. An Stelle von Cues werden in [Fahrmair, 2005] von Kontext-Interpreter geredet, also entsprechend den drei Schichten aus Abbildung 4.20 wird von Sensor-Kontext, Interpreter-Kontext und Adaption-Kontext geredet. Solche Architekturen sind grundlegend im Bereich der Datenfusion in Multi-Sensor Systemen [Abidi, 1992]. Datenfusion, als solche, behandelt die synergetische Kombination von Daten diverser Quellen mit der Absicht, ein besseres Verständnis der beobachteten Szenerie zu gewinnen.

Die Interpretation der Sensor-Daten und die Ableitung der Kontextinformationen, wie es beispielsweise von den Cues durchgeführt wird, hängt zum einen von den eingesetzten Sensoren, und zum anderen von der Anwendungsdomäne ab. Die Entscheidung, welche Sensoren tatsächlich eingesetzt werden, wird oft erst im Systementwurf getroffen. Lediglich Rahmenbedingungen, die dabei gelten, werden im RE berücksichtigt. Die Berücksichtigung der Rahmenbedingung im RE, beispielsweise bei der Verfeinerung der Stakeholder-Ziele, ist allerdings nicht spezifisch für kontextsensitive Anwendungen, und werden entsprechend von der RE-CAWAR Methodik im Kapitel 5 nicht in den Vordergrund gestellt.

Der zweite Aspekt der Aktualisierung des Kontextes betrifft die Änderung (Hinzufügen und/oder Entfernen) der Elemente eines Kontextmodells (d.h. der Fakttypen im Ganzen, oder der beteiligten Entitättypen, Prädikate bzw. Rollen). Eine solche Änderung erfolgt ähnlich wie die Konstruktion des eigentlichen Modells im Rahmen des Conceptual Context Modeling Schema – CCMS (siehe Abschnitt 5.3.2).

4.9 Zusammenfassung

Zentrale Bedeutung eines Kontextmodells in kontextsensitiver Anwendungen liegt in ihrer Verwendung bei der Entscheidung, wann welche Adaption angebracht ist. Die Erstellung der Kontextmodelle können sehr komplex werden, insbesondere wenn die Adaptionsentscheidungen viele Faktoren berücksichtigen müssen. Je mehr Kontextinformationen betrachtet werden, desto optimaler werden die Adaptionsentscheidung, und dementsprechend können die Anwendungen nützlicher werden. Ein Navigationssystem kann als optimale Route auf Basis des Nutzerziels, schnell ans Ziel zu kommen, Autobahn verwenden. Aber die Nützlichkeit der Anwendung kann dadurch erhöht werden, wenn sie Verkehrsflussinformationen bei ihrer Entscheidungen mitberücksichtigt. In einem Anwendungsfall wie etwa „sobald eine Nachricht eingetroffen ist, soll der Nutzer benachrichtigt werden“, spielen eine Reihe von Kontextinformationen eine wichtige Rolle. Beispiele hierfür sind die Nutzerpräferenzen bzgl. der Benachrichtigungsart, seine Beziehung zu anderen Personen in seiner unmittelbarer Nä-

he, sowie die für die Zwecke der Benachrichtigung zur Verfügung stehenden Geräte. Mit etwas mehr Sorgfalt stellt man weitere relevante Kontextinformationen für die Benachrichtigungsanwendung fest.

Zum Kontext der Nutzung einer Anwendung zählen also grundsätzlich Informationen, wie den Aufenthaltsort des Benutzers, Geräusche und Helligkeit am Nutzungsort, anwesende Personen (das soziale Umfeld), verfügbare Geräte und deren Eigenschaften, Geste und Blickrichtung des Benutzers, Aufgaben und Ziele des Benutzers und viel mehr. Eine Klassifikation dieser Informationen würde es ermöglichen, sie systematisch und zielgerichtet zu erarbeiten. Auch die Feststellung der Relevanz der einzelnen Informationen für die Adaption der Anwendung würde dadurch systematischer erfolgen. Präzisere Adaptionsentscheidungen können dann getroffen werden, und Abweichungen zwischen Nutzererwartung und Systemverhalten können seltener vorkommen.

In diesem Kapitel haben wir ein Framework zur Modellierung des Kontextes vorgeschlagen. Kern des Frameworks ist ein grundlegendes Verständnis des Kontextes, bei dem seine zentralen Aspekte *Nutzer, Aktivität und Einsatzumgebung sowie ihre Wechselwirkungen unter einander und ihre Veränderungen über die Zeit* hervorgehoben werden.

Um die Aspekte des Kontextes adäquat abzudecken, wurden in diesem Framework sechs Modelle eingeführt, die allesamt Teil eines umfassenden Kontextmodells sein können: ein Nutzermodell, ein Aufgabenmodell, ein Umgebungsmodell, ein Plattformmodell, ein Interaktionsmodell und ein Präsentationsmodell. Dabei sind die ersten drei Modelle zentraler Natur und die letzten drei als unterstützende Modelle zu betrachten. Grundlegend in diesem Framework ist ebenfalls die Integration der einzelnen Modelle in ein umfassendes Kontextmodell.

Bei dem vorliegenden Framework steht eine Beschreibung der möglichen Inhalte der Modelle im Vordergrund. Geeignete Modellierungstechniken wurden vorgeschlagen. Ein Konzept zur Integration der Modelle in ein umfassendes Kontextmodell wurde ebenfalls vorgeschlagen. Es wurde mehrfach betont, dass die Erstellung der Modelle diszipliniert im RE erfolgen soll. Eine sorgfältige Modellierung des Kontextes – und damit einhergehend die Identifikation der Nutzungssituationen der Anwendung – ist notwendig, insbesondere um feststellen zu können:

1. In welcher Situation kann die Anwendung genutzt werden (d.h. was sind die gedachten Nutzungssituationen)?
2. Welche Anwendungsfunktionen sind in welchen Nutzungssituationen angebracht (Grundlage der Adaptionsentscheidung)?

Wir haben bislang außen vorgelassen, wie die aufgeführten Modelle methodisch im Requirements Engineering erstellt werden, und haben uns lediglich auf ihre möglichen Inhalten konzentriert. Im Kapitel 5 gehen wir näher auf die methodische Erstellung der Modelle ein. Dort beschreiben wir im Rahmen einer RE-Methodik einen Ansatz zur Erstellung der Modelle, charakteristisch für die Feststellung der Nutzungssituationen kontextsensitiver Anwendungen. Wir beschreiben ebenfalls einen Ansatz zur integrierten Erfassung, Analyse und Dokumentation der Anforderungen sowie der Situationen, in denen die Anforderungen erfüllt sein sollen.

Kapitel 5

Die RE-CAWAR Methodik

Es ist nicht genug zu wissen, man muss auch anwenden;
Es ist nicht genug zu wollen, man muss auch tun.

Johann Wolfgang von Goethe

Inhaltsangabe

5.1 Einleitung

Im RE kontextsensitiver Anwendungen haben wir zwei grundlegende Fragestellungen zu adressieren. Zum einen, wie kommt man zu den im Kapitel 4 aufgeführten charakteristischen Kontextinformationen einer Nutzungssituation? Und zum anderen, welche Systemreaktion bzw. Funktion wird in welcher Nutzungssituation erwartet, und soll dementsprechend von der Anwendung angeboten werden? Die erste Fragestellung adressiert geeignete Methoden zur Erstellung von Kontextmodellen. Dabei ist es wichtig, dass die Kontextinformationen die voraussichtlichen Nutzungssituationen eindeutig charakterisieren. Die zweite Fragestellung erfordert geeignete Methoden zur Ermittlung, Analyse und Spezifikation funktionaler Anforderungen kontextsensitiver Anwendungen aus Sicht der Anwendungsnutzung. Hierbei sind die Erwartungen der Nutzer zu berücksichtigen. Bei der Spezifikation der Anforderungen an kontextsensitive Anwendungen sollen jedoch nicht lediglich die Funktionalitäten der Anwendungen

beschrieben werden, sondern auch gleichzeitig die Situationen, in denen sie aufgerufen oder genutzt bzw. angeboten werden.

Um die obigen Fragestellungen zu adressieren, führen wir im diesem Kapitel eine Methodik für das Requirements Engineering kontextsensitiver Anwendungen ein (RE-CAWAR – *Requirements Engineering for Context-Awareness*). Die RE-CAWAR Methodik bietet einen Ansatz, mit Hilfe dessen die Kontextmodelle, und damit einhergehend die charakteristischen Kontextinformationen der Nutzungssituationen einer Anwendung, schrittweise erstellt bzw. identifiziert werden. Die Methodik bietet ebenfalls einen Ansatz zur Bestimmung der funktionalen Anforderungen, die in einer gegebenen Nutzungssituation von der Anwendung gefordert sind. Die Kontextmodellierung und die Ermittlung der funktionalen Anforderungen erfolgen in einer integrierten Kontext- und Szenarioanalyse.

Im Abschnitt 5.2 geben wir einen Überblick der RE-CAWAR Methodik. Die Methodik ist in drei Phasen gegliedert: eine Anfangsphase mit Scoping-Aktivitäten, eine Hauptphase mit Aktivitäten der integrierten Kontext- und Szenario-Analyse und eine Endphase mit Aktivitäten der Konsolidierung der Ergebnisse. Die einzelnen Phasen der Methodik werden in diesem Abschnitt beschrieben. Anschließend werden die durchzuführenden Schritte der Methodik im Abschnitt 5.3 erläutert. Bei der Beschreibung der Phasen wird zunächst einen Überblick der Methodik gegeben. Weiterführende Details werden bei der Beschreibung der durchzuführenden Schritte der Methodik gegeben. Zu den einzelnen Schritten werden die grundlegenden Konzepte und durchzuführende Aktivitäten beschrieben. Illustrierende Beispiele aus den begleitenden Fallstudien des kontextsensitiven Terminplaners und des kontextsensitiven Scheibentönungssystems (siehe Abschnitt 2.5) werden an entsprechenden Stellen angegeben. In diesem Kapitel wird ein Modellbasierter RE-Ansatz vorgestellt. Dabei stehen die Artefakte (d.h. die zu erarbeiteten Ergebnisse) im Vordergrund, und nicht etwas die Prozesse zur Erstellung dieser Artefakte.

5.2 Überblick der RE-CAWAR Methodik

Requirements Engineering für eine kontextsensitive Anwendung bedeutet in erster Linie die Berücksichtigung des Nutzungskontextes bei der Ermittlung, Analyse und Spezifikation der Anforderungen [Kolos-Mazuryk et al., 2006, Sutcliffe et al., 2006, Goldsby und Cheng, 2006, Sitou und Spanfelner, 2007, Welsh und Sawyer, 2008, Seyff et al., 2008]. Dies setzt eine sorgfältige und systematische Modellierung des Nutzungskontextes voraus. Für die Zwecke der Modellierung des Nutzungskontextes haben wir im Kapitel 4 ein Framework konzipiert. Darin haben wir die möglichen Inhalte eines Kontextmodells beschrieben – ohne anzugeben, wie diese im einzelnen herausgearbeitet werden. Mit anderen Worten fehlt bislang ein konstruktiver Ansatz zur Modellierung der Kontextinformationen. Im Kapitel 3 haben wir als Ergebnis der Analyse der Herausforderungen im RE kontextsensitiver Anwendungen u.a. festgestellt, dass die Anforderungen integriert mit dem Nutzungskontext zu analysieren sind. Diese Erkenntnis bildet einen Grundsatz der RE-CAWAR Methodik, die in drei Phasen *(Anfangsphase, Hauptphase, Endphase)* gegliedert ist.

Den Ausgangspunkt der Aktivitäten in der Methodik-Hauptphase bilden Stakeholder-Ziele, insbesondere Nutzer- und Geschäftsziele. Diese werden in der Anfangsphase

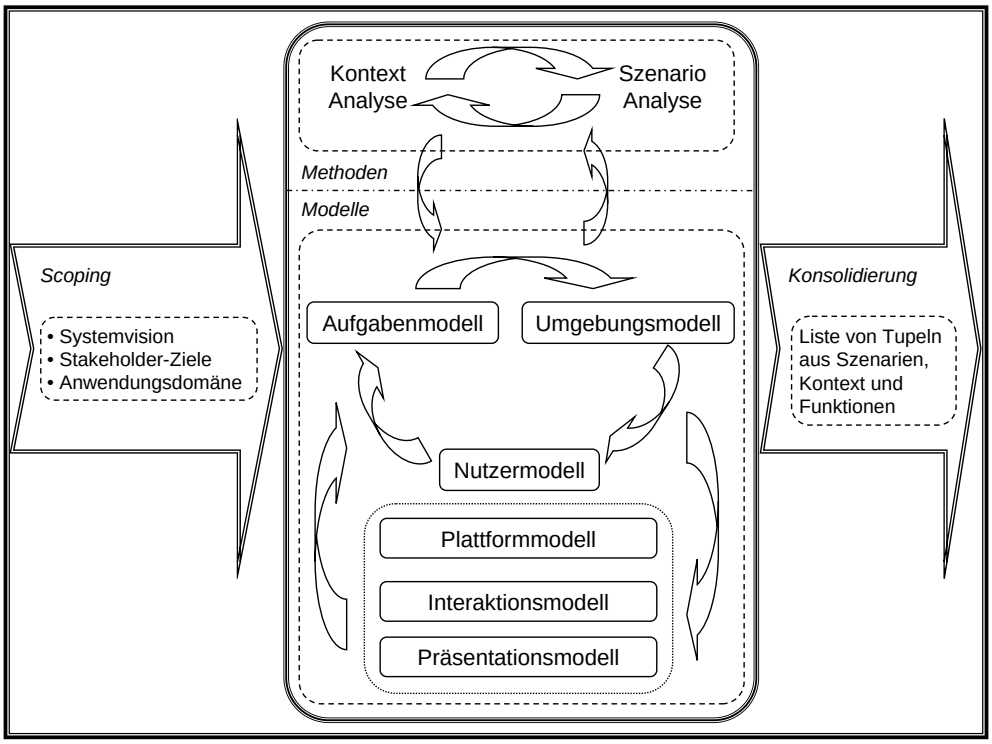

Abbildung 5.1: Überblick der RE-CAWAR Methodik

der Methodik im Rahmen eines Scopings identifiziert. Zusätzlich werden im Scoping die Systemvision definiert, sowie Anwendungsdomäne und Systemgrenzen festgelegt.

Grundlegend für die Hauptphase der RE-CAWAR Methodik sind die Methoden der Szenario- und Kontextanalyse sowie die dabei erarbeiteten Teilmodelle des Nutzungskontextes und die funktionalen Anforderungen des Systems. Unter Anwendung der Methoden der Kontext- und Szenario-Analyse werden die Teilmodelle des Kontextes konstruiert. Die funktionalen Anforderungen an das System werden ebenfalls in dieser Phase der Methodik identifiziert.

Das Ergebnis der Anwendung der RE-CAWAR Methodik ist eine konsolidierte Liste von Anforderungen an das zu entwickelnde System. Zu jeder Anforderung dieser Liste wird stets angegeben, welcher Nutzungssituation sie zuzuordnen ist bzw. in welcher Situation die realisierende Funktion anzubieten ist. Zudem wird jeweils ein illustrierendes Szenario zur Nutzung der Funktion angegeben. Es ergibt sich also eine Liste von Tupeln aus funktionalen Anforderungen, charakteristischen Kontextinformationen einer Nutzungssituation und illustrierenden Szenarien (so genannten *Contextual Requirements Chunks*). Eine Abstraktion aller Nutzungssituationen bildet das grundlegende Kontextmodell zur Nutzung der Anwendung.

Abbildung 5.1 illustriert den Überblick der RE-CAWAR Methodik. Im Folgenden gehen wir genauer auf die einzelnen Phasen der Methodik ein: die Anfangsphase im Abschnitt 5.2.1, die Hauptphase im Abschnitt 5.2.2 und die Endphase im Abschnitt 5.2.3. Im Abschnitt 5.3 werden ausführliche Beschreibungen der Konzepte der durchzuführenden Schritte zusammen mit geeigneten Illustrationen anhand der begleitenden Fallstudien angegeben.

5.2.1 Anfangsphase: Scoping

Um das Scoping zu verdeutlichen, überlegen Sie sich folgendes Beispiel: Ein kontextsensitives Software-System soll entwickelt werden, das in unterschiedlichen Situationen des täglichen Lebens genutzt wird. Das beobachtbare Verhalten des Systems soll dabei an die jeweiligen Nutzungssituationen angepasst werden. Zu diesem Zwecke wird die RE-CAWAR Methodik eingesetzt, um das Requirements Engineering systematisch durchzuführen. Dabei wird die Kontextsensitivität des Systems explizit behandelt, insbesondere bei der Modellierung der Nutzungsszenarien des Systems und der Analyse der Anforderungen an das System. Der Kontext, in dem das System eingesetzt wird, wird explizit modelliert und fungiert als essentielle Grundlage der Adaption der Anwendung.

Die Methodik beginnt mit einem Scoping mit den wesentlichen Aktivitäten: die Definition der Systemvision, die Identifikation der Stakeholder und ihre Ziele sowie die Festlegung der Anwendungsdomäne und der Systemgrenzen. Der Scoping-Prozess ist leider nur selten ein einfacher Prozess, bei dem alles von vorne herein eindeutig ist [Sommerville und Sawyer, 1997]. Dies hängt damit zusammen, dass Nutzer prinzipiell nicht immer wissen, was sie wollen. Auch sind ihre Wissen über das zu entwickelnde System oft sehr vage. Sie haben kaum Erfahrung mit neuartigen Anwendungen, was bei kontextsensitiven Anwendungen grundsätzlich der Fall ist. Die Festlegung der Anwendungsdomäne sowie der Grenzen des Systems kann ebenfalls nicht in einem Schritt präzise vorgenommen werden. Es empfiehlt sich daher den Scoping-Prozess iterativ zu gestalten. In jeder Iteration werden neue Erkenntnisse gewonnen, die dazu dienen die Systemvision, die Stakeholder-Ziele und die Anwendungsdomäne sowie die Systemgrenzen präziser zu beschreiben.

Zur Durchführung des Scoping-Prozesses existieren eine Reihe von Techniken, von denen Brainstorming und Workshops (siehe [Sommerville und Sawyer, 1997, Kotonya und Sommerville, 1998]) mit Beteiligung potentieller Stakeholder am geläufigsten sind. Unabhängig davon, welche Technik eingesetzt wird, ist es wichtig, systematisch vorzugehen: relevante Stakeholder und ihre Ziele sind zu identifizieren, die Anwendungsdomäne ist festzulegen, die Systemvision ist zu definieren. Relevante Stakeholder haben stets einen Einfluss auf die Systementwicklung bzw. sie verfolgen mit der Systementwicklung stets ein gewisses Ziel. Alle relevanten Stakeholder, ihre Einflüsse und Ziele sind dabei zu identifizieren. Zur Identifikation bzw. Festlegung der Anwendungsdomäne werden Experten der Anwendungsdomäne und notwendiges Domänenwissen mit in die Analyse einbezogen.

Bei der Entwicklung kontextsensitiver Anwendungen ist es wichtig, dass den Stakeholder, insbesondere den Nutzern und Nutzergruppen, die Einsatzgrenze der Anwendung kommuniziert wird. Eine solche Grenze kann an dieser Stelle zwar nicht endgültig bestimmt werden, allerdings sollten erste Erkenntnisse hierzu festgehalten und kommuniziert werden. Diese Maßnahme erfüllt den Zweck, unerfüllbaren bzw. nicht vorgesehenen Erwartungen seitens der Stakeholder bereits in frühen Phasen vorzubeugen. Anschließende Aktivitäten in der Hauptphase der Methodik werden durch ein sorgfältige Scoping – mit einer eindeutigen Festlegung der Systemgrenzen, einer ausführlichen Beschreibung der Stakeholder-Ziele und der Systemvision – zielgerichtet durchgeführt.

5.2.2 Hauptphase: Integrierte Szenario-/Kontext-Analyse

Die RE-CAWAR Methodik ist ein modellbasierter Ansatz des RE kontextsensitiver Anwendungen. Das dabei verfolgte Ziel ist, die Anforderungen an die Anwendungen im Form einer konsolidierten Liste von Tupeln aus Anwendungsszenarien, Nutzungskontext und Systemfunktionen zu spezifizieren. Ein umfassendes Modell des Nutzungskontextes wird bei der Anwendung der Methodik explizit entwickelt. Kern der Methodik ist eine integrierte Kontext- und Szenario-Analyse, die in der Hauptphase vorgenommen wird. Bei der Durchführung der integrierten Kontext- und Szenario-Analyse werden Teilmodelle des Nutzungskontextes, wie sie im Kapitel 4 konzipiert und aufgeführt wurden, erstellt und miteinander integriert. Durch die Integration der Teilmodelle werden die Nutzungssituationen der Anwendung in Form eines Kontextmodells systematisch modelliert.

Nach der Identifikation der Nutzer samt ihrer Ziele in der Anfangsphase der Methodik werden Anwendungsfälle modelliert und analysiert. Dabei werden für die einzelnen Nutzer bzw. Nutzergruppen jeweils modelliert, wie sie die Anwendung nutzen werden, um ihre Ziele zu erreichen (auch Anwendungs- bzw. Nutzungsszenarien). In einer *Szenario-Analyse* werden Anwendungsfälle modelliert und analysiert. Ausgehend von den Anwendungsfällen werden funktionalen Anforderungen abgeleitet.

Die Umstände, unter welchen die Nutzer die Anwendung nutzen, um ihre Ziele zu erreichen, werden ebenfalls modelliert und analysiert. Eventuell sind die Anwendungsfälle so detailliert beschrieben, dass sie eine Beschreibung der Umstände bzw. Bedingungen der Nutzung der Anwendung bereits enthalten. In diesem Fall ist nur noch die Analyse dieser Umstände bzw. Gegebenheiten der Nutzung nötig: die so genannte *Kontext-Analyse*. Dadurch werden Kontextinformationen explizit (und nicht versteckt in den Szenarien) analysiert und modelliert. Eine Besonderheit des RE kontextsensitiver Anwendungen liegt in der expliziten Behandlung des Nutzungskontextes. Dementsprechend spielt die Kontext-Analyse eine zentrale Rolle in der Methodik.

Während der Kontext-Analyse werden die Teilmodelle des Kontextes – das Nutzermodell, das Aufgabenmodell und das Umgebungsmodell sowie die unterstützenden Modelle (Plattformmodell, Interaktionsmodell und Präsentationsmodell) – sukzessive aufgebaut. Eine ausführliche Beschreibung der möglichen Inhalte dieser Modelle ist im Kapitel 4 gegeben. Die Erstellung der Teilmodelle erfordert, dass jedes Anwendungsszenario mit Informationen über die Gegebenheiten der Nutzung angereichert wird. Mit der Erstellung eines Teilmodells werden Erkenntnisse gewonnen, die dazu führen, dass neue Anwendungsszenarien identifiziert werden.

Nach der Erstellung der Teilmodelle erfolgt ihre Integration in ein umfassendes Modell des Kontextes. Für diese Zwecke werden die Wechselwirkungen zwischen den Aspekten des Kontextes (siehe Abschnitt 4.2) untersucht. Die einzelnen Nutzungssituationen werden dabei identifiziert. Jede Nutzungssituation ist ein Fakt über die Inhalte der Teilmodelle des Kontextes. Eine Abstraktion aller möglichen Nutzungssituationen stellt das Kontextmodell der Anwendung dar.

Die Kontext-Analyse und die Szenario-Analyse werden in der RE-CAWAR Methodik integriert durchgeführt. Neue Erkenntnisse aus der Analyse des Kontextes erfordern meist eine Überarbeitung der Szenarien und umgekehrt. Aus diesem Grund ist es wichtig, die Aktivitäten in dieser Phase der Methodik iterativ durchzuführen.

5.2.3 Endphase: Konsolidierung der Ergebnisse

Im Anschluss an die Hauptphase ist eine Integration und Konsolidierung der Ergebnisse vorgesehen. Sie erfolgen in der Endphase der RE-CAWAR Methodik. Die identifizierten funktionalen Anforderungen werden mit den jeweiligen Nutzungssituationen gekoppelt. Diese Koppelung beschreibt folgende Tatsache: *wenn* die spezifizierte Nutzungssituation auftritt, *dann* soll diejenige Systemfunktion angeboten werden, welche die angegebene funktionale Anforderung realisiert.

Zur Illustration der Kopplung der einzelnen Nutzungssituationen mit den jeweiligen funktionalen Anforderungen werden Szenarien angegeben. Die illustrierenden Szenarien beschreiben beispielhaft die Nutzung der Funktionen, welche die angegebene funktionale Anforderung realisieren – innerhalb der jeweils angegebenen Nutzungssituationen. Dementsprechend enthalten sie sowohl Informationen über die Nutzungssituationen als auch über die funktionalen Anforderungen.

Definition 5.1 Contextual Requirements Chunk – CRC

☞ Jedes Tripel bestehend aus den charakteristischen Informationen über die Nutzungssituationen, den funktionalen Anforderungen sowie den illustrierenden Szenarien nennen wir *Contextual Requirements Chunk – CRC*.
Sei CRC die Menge aller Contextual Requirements Chunks, REQ die Menge aller funktionalen Anforderungen, SIT die Menge aller Nutzungssituationen und SCE die Menge aller illustrierenden Szenarien. Es gilt:

$$CRC = REQ \times SIT \times SCE$$

Jedes Contextual Requirements Chunk x ist dementsprechend der Form $x \in REQ \times SIT \times SCE$. Prinzipiell sind beliebige Kombinationen von Anforderungen, Situationen und Szenarien in den CRCs, wie sie in Definition 5.1 aufgeführt sind, erlaubt. Es gilt allerdings, dass jedes Szenario in einem CRC sowohl von der in dem CRC angegebenen Anforderung als auch von der Situation abhängig ist. Auch gilt, dass viele Anforderungen in einer gleichen Situation vorkommen können. Es gilt allerdings (spätestens bei der Konsolidierung der Contextual Requirements Chunks) die Einschränkung, dass widersprechende Anforderungen nicht in einer gleichen Situation vorkommen dürfen.

Auf die Charakterisierung der Mengen REQ, SIT und SCE kommen wir später zurück. Wir merken an dieser Stellen an, dass dadurch, dass die Contextual Requirements Chunks für jede Nutzungssituation die geforderte funktionale Anforderung enthalten, beschreiben sie auf Ebene der Problemstellung (also in der Sprache der Nutzer) die Logik der Adaption. Denn die CRCs sind von der Form:

WENN $Nutzungssituation$ **DANN** $Funktionale\ Anforderung$
MIT $Illustration\ anhand\ eines\ Anwendungsszenarios$

Die Chunks werden in einer Tabelle festgehalten, welche das Kernstück im Anforderungsdokument einer kontextsensitiven Anwendung (*User Requirements Specification* nach [Wiegers, 1999], [Geisberger, 2005] oder auch [Geisberger et al., 2006]) bildet.

Abbildung 5.2 illustriert die tabellarische Darstellung der CRCs. Die angegebenen funktionalen Anforderungen werden durch Systemfunktionen realisiert (*Sy-*

stem Requirements Specification nach [Wiegers, 1999], [Geisberger, 2005] oder auch [Geisberger et al., 2006])[1]. Die Nutzung der Systemfunktionen erfolgt unter den Gegebenheiten, die anhand der Nutzungssituationen abgebildet sind. Die illustrierenden Szenarien beschreiben beispielhaft mögliche Abläufe bei der Nutzung der Funktionen, die die funktionalen Anforderungen realisieren.

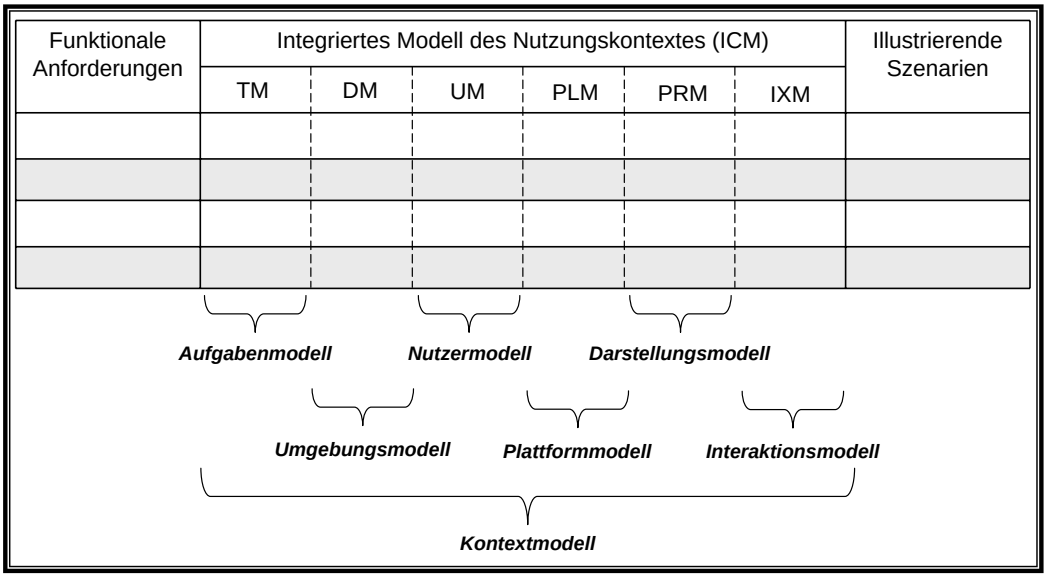

Abbildung 5.2: Konsolidierte Contextual Requirements Chunks

Definition 5.2 Nutzungssituation in einem CRC

☞ Eine Nutzungssituation $sit \in SIT$ ist je nach Anwendungsfall durch Informationen über (1) den Nutzer, (2) die Aufgaben, die der Nutzer durchführt, (3) die Umgebung, in der der Nutzer die Anwendung nutzt, (4) die Plattform, auf der die Anwendung läuft, (5) die Interaktionen, die den Nutzer benötigt, um die Aufgaben durchzuführen, und (6) die Art/Modalität der Präsentation bzw. Darstellung der Elemente der Interaktionen zwischen dem Nutzer und dem System charakterisiert.
Sei USR die Menge aller Informationen über den Nutzer, TSK die Menge aller Informationen über die Aufgaben, die der Nutzer durchführt, ENV die Menge aller Informationen über die Einsatzumgebung, PLT die Menge aller Informationen über die Plattform, auf der die Anwendung läuft, ITX die Menge aller Informationen über die Interaktionen, die den Nutzer benötigt, um die Aufgaben durchzuführen, und PRE die Menge aller Informationen über die Präsentation der Elemente der Interaktionen zwischen dem Nutzer und dem System. Wir fordern:

$$\forall sit \in SIT: \; sit = (usr, tsk, env, plt, itx, pre) \; mit \; (usr \cup tsk \cup env \neq \emptyset)$$

Dabei gilt:

$$usr \subseteq USR, tsk \subseteq TSK, env \subseteq ENV, plt \subseteq PLT, itx \subseteq ITX, pre \subseteq PRE$$
$$\wedge \; SIT = USR \times TSK \times ENV \times PLT \times ITX \times PRE$$

[1] Das Thema der System Requirements Specification ist ein eigenes Thema für sich und wird im Rahmen der RE-CAWAR Methodik nicht behandelt.

Beispiel 5.1 *Contextual Requirements Chunk - Vibration*

Funktionale Anforderung – Req. RQ01
Der Nutzer soll ausschließlich per Vibrationsalarm benachrichtigt werden.

Nutzungssituation – Sit. US01

a) ***UM01:*** *Nutzer ist Journalist; Er besucht gern Sport-PK; Er will jeder Zeit möglichst sofort benachrichtigt werden, sobald eine Sport-PK betreffende Mitteilung veröffentlicht wird.*
b) ***TM01:*** *An Meeting teilnehmen.*
c) ***EM01:*** *Meetingsraum verfügt über einen flächendeckenden GSM-Empfang.*

Illustrierendes Szenario – ISce. ISC01
Herr Weber ist gerade in einem Meeting und während dessen wird die Absage einer geplanten Pressekonferenz bekannt gemacht. Aufgrund seines Profils stuft sein CATS-Client dieses Event als interessant für ihn ein. Um weitere Teilnehmer in der näheren Umgebung des Nutzers nicht zu stören, benachrichtigt er ihn per Vibrationsalarm.

Beispiel 5.2 *Contextual Requirements Chunk - Audio*

Funktionale Anforderung – Req. RQ02
Der Nutzer soll ausschließlich per Audio-Signal benachrichtigt werden.

Nutzungssituation – Sit. US02

a) ***UM01:*** *Nutzer ist Journalist; Nutzer besucht gern Sport-PK; Nutzer will jeder Zeit möglichst sofort benachrichtigt werden, sobald eine Sport-PK betreffende Mitteilung veröffentlicht wird.*
b) ***TM02:*** *Auto fahren.*
c) ***EM:*** *–*

Illustrierendes Szenario – ISce. ISC02
Herr Weber fährt gerade Auto und während dessen wird die Absage einer geplanten Pressekonferenz bekannt gemacht. Aufgrund seines Profils stuft sein CATS-Client dieses Events als interessant für ihn ein. Um sicher zu gehen, dass Herr Willi davon erfährt, benachrichtigt ihn sein CATS Client per Audio-Signal.

Beispiel 5.3 *Contextual Requirements Chunk - Audio und Vibration*

Funktionale Anforderung – Req. RQ03
Der Nutzer soll gleichzeitig per Audio-Signal und Vibrationsalarm benachrichtigt werden.

Nutzungssituation – Sit. US03

a) ***UM01:*** *Nutzer ist Journalist; Er besucht gern Sport-PK; Er will jeder Zeit möglichst sofort benachrichtigt werden, sobald eine Sport-PK betreffende Mitteilung veröffentlicht wird.*
b) ***TM03:*** *Einkaufstour machen.*
c) ***EM02:*** *Die Einkaufstour findet in der Innenstadt statt. Die Innenstadt ist gerade voller Passanten. Es ist laut.*

Illustrierendes Szenario – ISce. ISC03
Herr Weber macht gerade eine Einkaufstour in der Innenstadt und während dessen wird eine Mitteilung bzgl. einer Sport-Pressekonferenz veröffentlicht. Aufgrund seines Profils stuft sein CATS-Client dieses Event als interessant für ihn ein. Um sicher zu gehen, dass Herr Willi davon erfährt, benachrichtigt ihn sein CATS Client gleichzeitig per Audio-Signal und Vibrationsalarm.

In der Definition 5.2 wird gefordert, dass für die Charakterisierung einer Nutzungssituation mindestens eine Angabe über den Nutzer, seine durchgeführten Aufgaben oder die Umgebung der Nutzung der Anwendung, benötigt wird. Diese Forderung lehnt sich an die drei zentralen Aspekte des Kontextes (siehe Abschnitt 4.2). Dadurch wird sichergestellt, dass jede Nutzungssituation durch Informationen aus mindestens einem der zentralen Aspekte des Kontextes charakterisiert ist. Eine Abstraktion aller Informationen, die in den Nutzungssituationen vorkommen, ergibt das Kontextmodell der Anwendung. $CTX\_Modell = Abstraktion(SIT)$, wobei die Abstraktion auf die konkrete Anwendung bezogen ist.

Die Beispiele 5.1, 5.2 und 5.3 zeigen Contextual Requirements Chunks aus der Fallstudie des kontextsensitiven Terminplaners.

Für die Spezifikation der Anforderungen an kontextsensitive Anwendungen aus Sicht der Problemstellung („User Requirements Specification“) wurde im Rahmen dieser Arbeit einen Gliederungsvorschlag erarbeitet (siehe Anhang B). Die Grundidee hinter dieser Gliederung besteht darin, sowohl die Anforderungen als auch deren zugehörige Nutzungssituationen gemeinsam (d.h. integriert miteinander) zu entwickeln. Im Kapitel B.7 eines Anforderungsdokuments einer kontextsensitiven Anwendung werden die CRCs zusammengefasst. Sie dienen als Grundlage für die weiteren Entwicklungsaktivitäten, insbesondere für die Aktivitäten beim Übergang von RE zum Design aber auch für die Kommunikation der Fähigkeiten der Anwendung an zukünftige Nutzer. Die CRCs bilden den Ausgangspunkt der Spezifikation der Systemanforderungen (System Requirements) inklusive der Adaptionsfunktion.

5.3 Durchzuführende Schritte

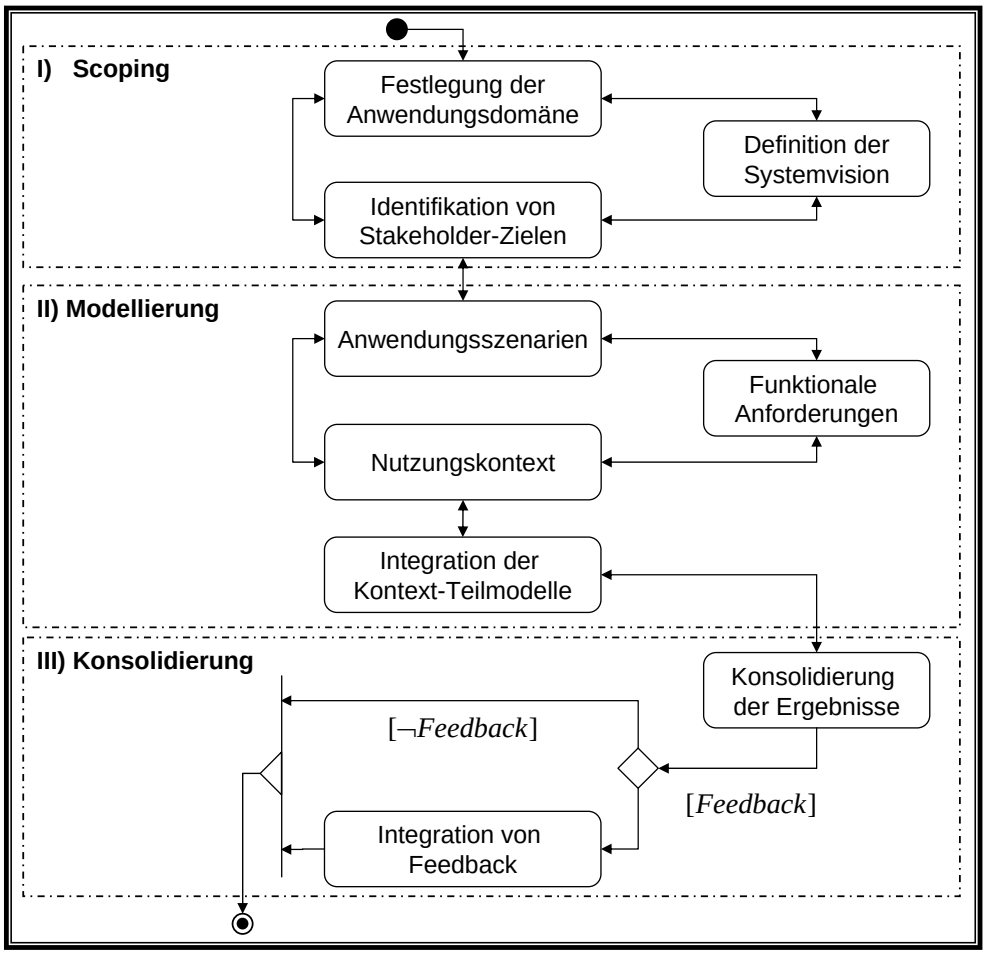

Abbildung 5.3: Schritte der RE-CAWAR Methodik

Wesentlich in der RE-CAWAR Methodik sind die Schritte, die durchgeführt werden, um von dem Ausgangspunkt (die Ziele der relevanten Stakeholder) zu den Ergebnissen (die konsolidierten Contextual Requirements Chunks sowie das Modell des Nutzungskontextes) zu gelangen. Im einzelnen sind es:

1. Scoping bestehend aus:
   (a) Identifikation der Stakeholder und ihrer Ziele
   (b) Festlegung der Anwendungsdomäne und Anwendungsgrenze
   (c) Definition der Systemvision
2. Modellierung bestehend aus:
   (a) Modellierung der Anwendungsszenarien ausgehend von den Zielen
   (b) Ableitung der funktionalen Anforderungen aus den Szenarien
   (c) Identifikation der Nutzungssituationen und Modellierung des Teilaspekte des Nutzungskontextes der Anwendung

(d) Integration der Teilmodelle des Nutzungskontextes in einem umfassenden Kontextmodell der Anwendung

3. Konsolidierungsaktivitäten bestehend aus

   (a) Konsolidierung der Ergebnisse aus den Modellierungsaktivitäten in Form von *Contextual Requirements Chunks*

   (b) Abarbeitung von Feedback (aus jeglicher Abstimmungsaktivität, z.B. nach neuen Erkenntnissen aus dem Systementwurf) und Iteration über die gesamte Methodik.

Abbildung 5.3 zeigt die einzelnen Schritte inklusive ihrer Übergänge[2]. In den nächsten Abschnitten beschreiben wir diese Schritte und illustrieren sie jeweils mit entsprechenden Beispielen aus den begleitenden Fallstudien.

5.3.1 Scoping-Aktivitäten

Ein erster Schritt in der RE-CAWAR Methodik, ist die Erklärung der Absicht, die mit dem zu entwickelnden System verfolgt wird. Es wird gleich zu Beginn eine Abgrenzung des Vorhabens vorgenommen. Dabei wird unter anderem eine Stakeholder-Analyse vorgenommen – mit dem Ziel, die relevanten Stakeholder und ihre Ziele zu identifizieren. Es wird ebenfalls die Anwendungsdomäne festgelegt und die Vision des Systems definiert (siehe Abbildung 5.4).

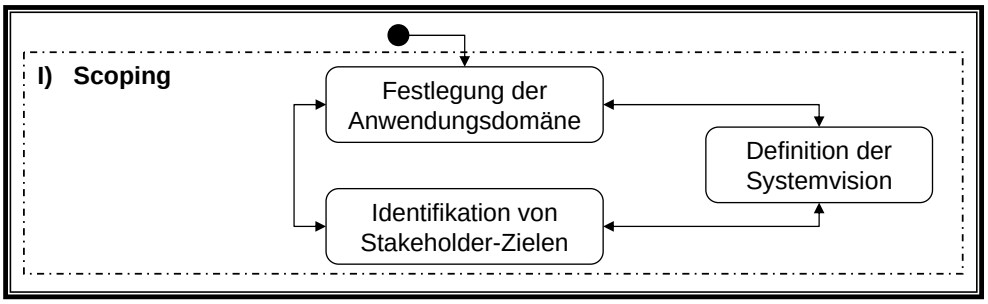

Abbildung 5.4: Scoping-Aktivitäten in RE-CAWAR

Schritt 1: Festlegung der Anwendungsdomäne

Die Scoping-Aktivitäten in der RE-CAWAR Methodik beginnen mit der Festlegung der Anwendungsdomäne des zu entwickelnden Systems. Ein wichtiger Schritt bei der Entwicklung von Software-Systemen im allgemeinen ist das Verstehen des Anwendungsgebiets. Damit wird u.a. die Abgrenzung des Anforderungsgegenstandes zielgerichtet vorgenommen.

Mögliche Anwendungsgebiete sind persönliche Assistenz am Beispiel von PDA („Personal Digital Assistant“), Laptop und Desktop, Unterhaltung am Beispiel von Musik, Video und Spiele, SmartHome, Fahrer-Assistenz, Fahrer-Information, Telekommunikation, Medizin oder auch Gebäudeautomatisierung.

[2] Die *bi*direktionalen Pfeile in der Abbildung sind lediglich aufgrund der Übersichtlichkeit verwendet worden. Sie stehen stellvertretend für zwei *uni*direktionale Pfeile in entgegengesetzter Richtung

Mit der Festlegung der Anwendungsdomäne wird das zu lösende Problem eingegrenzt [Sutcliffe, 1996, Jackson und Zave, 1993]. In Konsequenz wird die Identifikation relevanter Stakeholder sowie die Definition der Systemvision erleichtert. Je genauer die Anwendungsdomäne festgelegt ist, desto leichter fällt die Identifikation der relevanten Stakeholder sowie die Definition der Systemvision.

Schritt 2: Identifikation von Stakeholder-Zielen

Ein wichtiges Ziel der Softwareentwicklung, insbesondere im Fall kontextsensitiver Anwendungen, ist die Entwicklung eines Softwaresystems, das die Wünsche und Bedürfnisse der relevanten Stakeholder erfüllt. Bei den Stakeholder wird im Fall kontextsensitiver Anwendungen ein besonderes Augenmerk auf die Nutzer gelegt. Eine Spezifikation (bzw. Modellierung) der Anforderungen aus Sicht der unterschiedlichen Stakeholder ist ein essentieller Teil der Softwareentwicklung. Dabei bezeichnet ein Stakeholder (auch Interessenvertreter) eine Einzelperson, eine Gruppe von Personen oder eine Organisation, die Anteil oder Interesse an dem System, und dementsprechend an den Anforderungen, haben [Mitchell et al., 1997, Sharp et al., 1999, Nuseibeh und Easterbrook, 2000]. Für die Identifikation der Ziele verwenden wir das Template 5.1.

Template 5.1 *Ziel-Schablone*

Bezeichner: *Gmn:* $m, n \in \mathbb{N}$
Kurzbezeichnung: *Eine aussagekräftige Bezeichnung des Ziels*
Beschreibung: *Die Beschreibung des identifizierten Ziels*
Übergeordnetes Ziel: *Das übergeordnete Ziel falls zutreffend*
Stakeholder: *Die Interessenvertreter dieses Ziels*
Anmerkung: *Jegliche Anmerkung zu dem Ziel*

In der Stakeholder-Analyse werden die relevanten Stakeholder bereits in den frühen Phasen des Requirements Engineering identifiziert und in den Entwicklungsprozess einbezogen. Stakeholder können beispielsweise die Nutzer, die Entwickler, das Management, das Wartungspersonal, der Gesetzgeber oder auch Sponsoren sein. Sie haben allesamt Ziele, die sie mit dem zu entwickelnden System erreichen wollen. Wir fassen diese Ziele unter Nutzer- und Geschäftszielen zusammen. Damit betonen wir, dass insbesondere die Ziele aus Sicht der Nutzer und des Geschäftes einen hohen Stellenwert in der Methodik haben.

Im RE kontextsensitiver Anwendungen sind wir daran interessiert, die Ziele der Nutzer zu identifizieren. Damit verbunden sind ihre Wünsche und Bedürfnisse. Aus den Zielen werden im Hauptteil der Methodik mittels einer integrierten Kontext- und Szenario-Analyse insbesondere Nutzungssituationen und funktionale Anforderungen an das System ermittelt und analysiert. Die Szenario-Analyse erfolgt angelehnt an die identifizierten Stakeholder-Ziele.

Schritt 3: Definition der Systemvision

Beispiel 5.4 *Systemvision am Beispiel des Scheibentönungssystems*

Die Sichtbehinderung durch die blendende Sonne stellt eine Gefährdung im Straßenverkehr dar und führt häufig zu Unfällen. In den Wintermonaten wird der Fahrer durch die tief stehende Sonne geblendet, im Sommer ist die Sicht des Fahrers aufgrund der hohen Lichtintensität stark eingeschränkt.

Sonnenbrillen sind ein weit verbreitetes Mittel um der Sichtbehinderung durch die Sonne entgegenzuwirken und auch die Sonnenblenden innerhalb des Fahrzeugs verbessern die Lichtsituation. Dennoch sind beide Möglichkeiten lediglich suboptimal, da ihr Einsatz die Konzentration des Fahrers (zumindest kurzzeitig) beeinflusst:

1. *Der Fahrer muss die Sonnenbrille während der Fahrt aufsetzen; bei einem Fahrer mit Sehschwäche damit auch ein Brillenwechsel verbunden, d.h. seine Sehkraft ist kurzzeitig nicht ausreichend*
2. *Die Sonnenblende muss ebenfalls während der Fahrt herunter geklappt werden.*

Besonders deutlich wird dieser Aufwand an kognitiver Aufmerksamkeit bei häufig wechselnden Lichtverhältnissen, d.h. Wechsel zwischen hell und dunkel in zeitlich kurzen Abständen. Der Fahrer kann sich dann nicht mehr ausreichend auf die Verkehrssituation und das eigentliche Fahren konzentrieren und somit steigt das Unfallrisiko.

Das System der dynamischen Scheibentönung soll den Fahrer vor einer Blendung durch die Sonne schützen, indem es die Scheiben des Fahrzeugs entsprechend der Lichtverhältnisse (Lichtintensität, Lichteinfallswinkel) tönt.

Darüber hinaus kann die Scheibentönung beim geparkten Fahrzeug als Diebstahlschutz fungieren, indem die Scheiben bis zu einer Lichtdurchlässigkeit von 0% getönt werden und somit der Einblick von außen in das Fahrzeug verhindert wird.

Die Entwicklung kontextsensitiver Anwendungen sollte genau wie die Entwicklung jedes Software-System von einer Vision getrieben sein.

Die Systemvision legt die Zielsetzungen der Systementwicklung fest. In der Systemvision soll die langfristige Zielsetzung des zu entwickelnden Systems zusammengefasst sein. Bei der Definition der Systemvision sollen die Ziele der Systementwicklung und möglichst die Kernfunktionalität beschrieben werden. Die Systemvision dient als Grundlage für Entscheidungen während der gesamten Systementwicklung.

Ein gemeinsames Vorstellung des zu behandelnden Problems und der Zielsetzung des zu entwickelnden Systems ist die Basis für die weiteren Schritte im Entwicklungsprozess. Es ist daher notwendig bereits in den frühen Phasen die Systemvision zu definieren. Das Beispiel 5.4 zeigt die Systemvision des Scheibentönungssystems.

Eine Systemvision enthält mindestens eine kurze Einführung in das Anwendungsgebiet, eine Motivation für die Entwicklung und den Einsatz des Systems, die Nutzer des Systems und die Ziele, die mit der Entwicklung des Systems erreicht werden sollen.

5.3.2 Modellierungsaktivitäten

Die RE-CAWAR Methodik hat ihre wesentlichen Besonderheiten in den Modellierungsaktivitäten. Dabei geht es nicht um die reine Modellierung allein, sondern auch die einleitende Ermittlung und die begleitende Analyse. Diese Aktivitäten betreffen zum einen die Anwendungsszenarien und die funktionalen Anforderungen, und zum anderen die Kontextinformationen, zur Charakterisierung der unterschiedlichen Nutzungssituationen der Anwendung (siehe Abbildung 5.5).

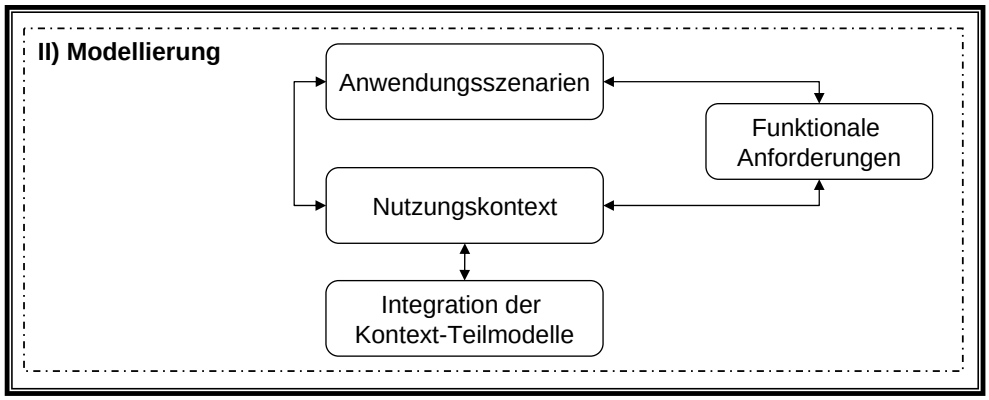

Abbildung 5.5: Modellierungsaktivitäten in RE-CAWAR

Schritt 4: Modellierung der Anwendungsszenarien

Eine Herausforderung bei der Entwicklung komplexer Software-Systeme ist die Ermittlung und Spezifikation der Anforderungen. Ein Hilfsmittel dabei ist der Einsatz von Szenarien [Amyot und Eberlein, 2003, Sutcliffe, 1997]. Dies gilt auch für kontextsensitive Anwendungen, insbesondere aufgrund der Tatsache, dass Nutzer meist nur vage Vorstellungen derartiger Anwendungen haben. Schließlich haben sie noch keine Erfahrungen mit diesen Anwendungen sammeln können. Sie wissen nicht immer, was sie wollen; ihre Wünsche können sie meist nur schwer einschätzen und artikulieren. Kontextsensitive Anwendungen sollen jedoch diese Wünsche in entsprechenden Nutzungssituationen erfüllen. Es bietet sich an, die Wünsche der Nutzer bezüglich der Nutzung der Anwendungen über die Entwicklung von Anwendungsszenarien (kurz Szenarien) zu erfassen.

Aus einer Untersuchung der Eignung aktueller RE-Ansätze zur Ermittlung von Anforderungen an kontextsensitive Anwendungen (siehe [Kolos-Mazuryk et al., 2006]) wurde festgestellt, dass Zielmodellierungsansätze (siehe beispielsweise [Dardenne et al., 1993], [Antón, 1997], [Yu, 1997], [Darimont et al., 1997], [van Lamsweerde, 2001] oder auch [Kavakli, 2002]) bei der Ermittlung der Nutzerwünsche und Anforderungen an das System hilfreich sein können. Diese Erkenntnis wurde bei der Konzeption der RE-CAWAR Methodik genutzt. Wie beispielsweise

in [Amyot und Eberlein, 2003] aufgeführt, können Szenarien für unterschiedliche Zwecke in der Systementwicklung eingesetzt werden. In der RE-CAWAR Methodik werden sie eingesetzt, um Ziele und Wünsche der Nutzer sowohl zu entdecken als auch zu konkretisieren. Abstrakt formulierte Ziele und Wünsche der Nutzer werden mittels Szenario-Modellierung konkretisiert. Durch diese Konkretisierung können weitere Ziele und Wünsche identifiziert werden.

Szenarien in der RE-CAWAR Methodik haben die in Abbildung 5.6 gegebene grundlegende Struktur. Diese Struktur ist angelehnt an die Szenario-Struktur aus dem ESPRIT CREWS Projekt [Rolland et al., 1998, Rolland et al., 1999]. Dabei wird eine Szenario definiert als *„a possible behavior limited to a set of purposeful interactions taking place among several agents“* [Plihon et al., 1998].

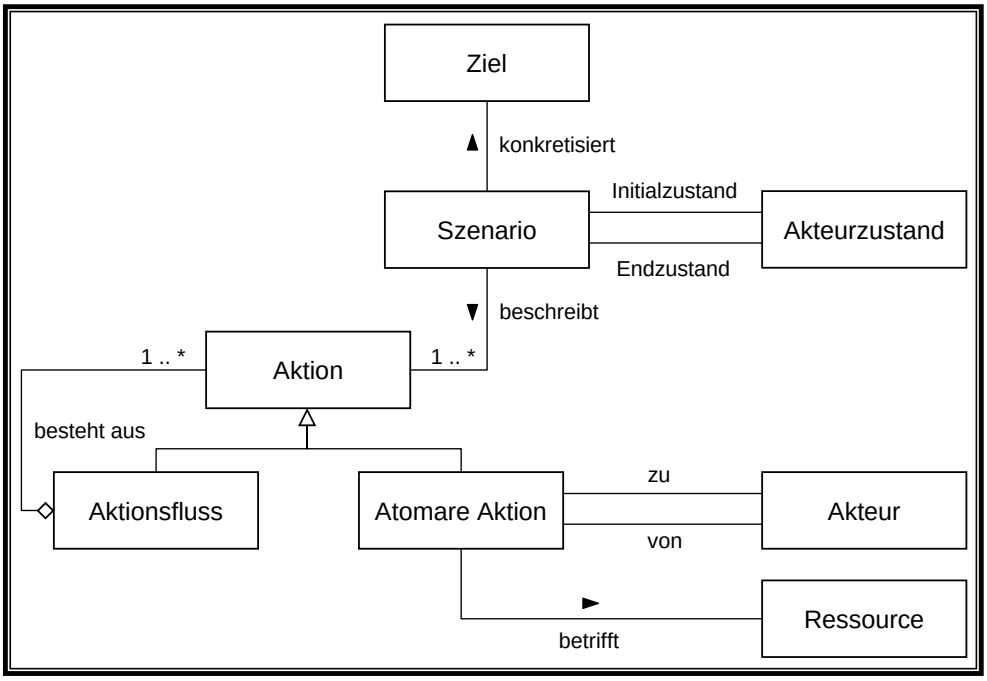

Abbildung 5.6: Szenarien in der RE-CAWAR Methodik

Jedes Szenario beschreibt eine Folge von Aktionen. Aktionen können atomar sein, d.h. sie sind nicht in weitere Aktionen zerlegbar. Atomare Aktionen beschreiben Interaktionen zwischen zwei Akteuren. Die Interaktionen betreffen Ressourcen, welche jegliche Objekte in der Anwendungsdomäne sein können. Ein Satz wie *„Der Nutzer steckt den Schlüssel in das Zündschloss“* ist eine atomare Aktion. Er beschreibt eine Interaktion zwischen den Akteuren *Nutzer und Zündschloss*, und betrifft die Ressource *Schlüssel*.

Aktionen können aber auch ein Fluss einzelner Aktionen sein. Sie bestehen dann aus einer Menge weiterer Aktionen. Die in einem Fluss von Aktionen vorkommenden Aktionen können sequenziell, alternativ, wiederholt oder gleichzeitig ausgeführt werden. Charakteristisch für Szenarien ist die Tatsache, dass sie einen Initialzustand und einen Endzustand haben (Zustand bzgl. den beteiligten Akteuren). Ein Initialzustand in einem Szenario gibt eine Vorbedingung an den Anstoß des Szenarios an. Ein Endzustand gibt eine Nachbedingung an, die nach der Ausführung der Aktionen des Szenarios herrscht. Für die Modellierung der Szenarien verwenden wir das Template 5.2.

Template 5.2 *Szenario-Schablone*

Bezeichner: *USCmn:* $m, n \in \mathbb{N}$
Kurzbezeichnung: *Eine aussagekräftige Bezeichnung für das Szenario*
Ziel: *Das Ziel, das mit dem Szenario konkretisiert wird*
Vorbedingung: *Vor Anstoß der Szenario-Schritte gilt:*
1. *Bedingung 1*
2. *Bedingung 2*
3. *...*

Aktionen: *Folgende Aktionen (Szenario-Schritte) werden ausgeführt:*
1. *Aktion*
2. *...*

Nachbedingung: *Nach Ablauf der Szenario-Schritte gilt:*
1. *Bedingung 1*
2. *Bedingung 2*
3. *...*

Anmerkung: *Jegliche Anmerkung zu dem Szenario.*

Beispiel 5.5 *Anwendungsszenario am Beispiel des kontextsensitiven Scheibentönungssystems*

Bezeichner: *USC01*
Kurzbezeichnung: *Scheiben des geparkten Fahrzeugs vollständig tönen*
Ziel: *G01 - Einblick von außen in das Fahrzeug beim geparkten Fahrzeug verhindern*
Vorbedingung:
1. *Das Fahrzeug ist geparkt*
2. *Das Scheibentönungssystem ist aktiv*

Aktionen:
1. *Der Fahrer drückt auf den Knopf zur Auslösung der vollständigen Tönung der Scheiben.*
2. *Das System tönt die Scheiben des Fahrzeugs bis zu einer Lichtdurchlässigkeit von 0%*

Nachbedingung:
1. *Die Scheiben des Fahrzeugs sind getönt*
2. *Die Lichtdurchlässigkeit bei den Scheiben des Fahrzeugs ist 0%*
3. *Das Scheibentönungssystem ist aktiv*

Anmerkung: *Vollständige Tönung der Scheiben := Tönung bis zu einer Lichtdurchlässigkeit von 0%.*

Die Modellierung der Anwendungsszenarien resultiert in einer Liste von Szenarien mit Angabe der durchzuführenden Aktionen (siehe Beispiel 5.5). Die Bedingungen, unter welchen diese Aktionen ablaufen, können zum Teil aus den Vorbedingungen entnommen werden. Sie sind jedoch an dieser Stellen nicht notwendigerweise ausführlich angegeben. Sie werden später in dem methodischen Schritt der Modellierung des Nut-

zungskontextes explizit ausgearbeitet. Es empfiehlt sich aufgrund der Übersichtlichkeit, die Szenarien in Anwendungsfälle mit Angabe von Akteuren zu gruppieren und ein Übersichtsdiagramm (in Form eines Use Case Diagramms [Jacobson, 2004]) zu erstellen. Eine solche Gruppierung bzw. Übersicht erleichtert die nachfolgende Analyse-Aktivität, welche in der Modellierung der funktionalen Anforderungen resultiert.

Schritt 5: Modellierung der funktionalen Anforderungen

Nach der Initiierung der Modellierung der Anwendungsszenarien beginnt der Schritt der Modellierung der funktionalen Anforderungen. Dieser Schritt begleitet die Szenario-Modellierung. Sie soll dazu dienen, funktionale Anforderungen an das System aus den Szenarien zu identifizieren. Allerdings werden durch die Identifikation neuer Anforderungen, die Modellierung zusätzlicher Szenarien angestoßen.

Die funktionalen Anforderungen beschreiben das beobachtbare Verhalten des Systems. Oft werden sie im Zusammenhang mit Sequenzen von Nutzer-Aktionen und System-Reaktionen betrachtet. Sie beschreiben, was das System leisten soll. Jede Aussage der Form *„Der Nutzer soll in der Lage sein, eine bestimmte Funktion aufzurufen“* oder *„Das System soll ein bestimmtes Verhalten aufzeigen“* ist wahrscheinlich eine funktionale Anforderung (siehe Template 5.3). Sie sind aus den Szenarien abzuleiten. Jede Aktion in einem Szenario ist ein potentieller Ursprung einer Anforderung. Dabei können die Vor- und Nachbedingungen Präzisionen in der Formulierung der Anforderungen herbeiführen.

Für jede Anforderung wird gefordert, dass mindestens ein Szenario angegeben wird, aus dem sie abgeleitet ist. Aufgrund dessen, dass die Anforderungen aus den in den Szenarien vorkommenden Aktionen abgeleitet werden, und eine Aktion in mehr als einem Szenario vorkommen kann, ist nicht ausgeschlossen, dass eine Anforderung aus mehr als einem Szenario abgeleitet wird.

Template 5.3 *Funktionale Anforderungen*

Bezeichner: *RQmn mit* $m, n \in \mathbb{N}$
Beschreibung: *Die Beschreibung hat eine der folgenden Formen:*

> *<Der Nutzer soll die Funktion* f_1 *des Systems aufrufen können>*
>
> *<Der System soll die Funktion* f_2 *anbieten>*
>
> *<Das System soll ein bestimmtes Verhalten aufzeigen>*
>
> *...*

Abgeleitet aus Szenarien: *Jeweilige Bezeichner und Kurzbeschreibung*
Anmerkung: *Jegliche Anmerkung bzw. Erläuterung zu der Anforderung*

Das Ergebnis dieses Schritts in der RE-CAWAR Methodik ist eine Liste funktionaler Anforderungen an das System (siehe Beispiel 5.6). Die aufgelisteten funktionalen An-

forderungen sind für die spätere Identifikation der Systemfunktionen maßgeblich. Allerdings fehlen noch die Kontextinformationen zur Charakterisierung der Situationen, in der die Funktionen angeboten werden sollen. Die Modellierung dieser Informationen erfolgt im nächsten Schritt der RE-CAWAR Methodik, dem Schritt der Modellierung des Nutzungskontextes.

Beispiel 5.6 *Funktionale Anforderungen am Beispiel des kontextsensitiven Scheibentönungssystems*

RQ01: Beschreibung: *Das System soll einen Bedienknopf zum Anstoß der vollständigen Tönung der Scheiben des Fahrzeugs haben.*
Abgeleitet aus Szenarien: *USC01 (Scheiben des geparkten Fahrzeugs vollständig tönen).*

RQ02: Beschreibung: *Der Nutzer soll per Knopf-Druck die vollständige Tönung der Scheiben des Fahrzeugs anstoßen können.*
Abgeleitet aus Szenarien: *USC01 (Scheiben des geparkten Fahrzeugs vollständig tönen).*

RQ03: Beschreibung: *Das System soll die Lichtdurchlässigkeit der Scheiben des Fahrzeugs messen.*
Abgeleitet aus Szenarien: *USC01 (Scheiben des geparkten Fahrzeugs vollständig tönen).*

...

RQ12: Beschreibung: *Das System soll die Scheiben des Fahrzeugs bis zu einer Lichtdurchlässigkeit von 0% tönen.*
Abgeleitet aus Szenarien: *USC01 (Scheiben des geparkten Fahrzeugs vollständig tönen – Leeres Fahrzeug).*
Anmerkung: *Mit einer Scheibentönung bis 0% Lichtdurchlässigkeit wird die vollständige Tönung der Scheiben erreicht.*

RQ13: Beschreibung: *Das System soll die Scheiben des Fahrzeugs bis zu einer Lichtdurchlässigkeit von 50% tönen.*
Abgeleitet aus Szenarien: *USC02 (Scheiben des geparkten Fahrzeugs vollständig tönen – Nicht leeres Fahrzeug).*

RQ14: Beschreibung: *Das System soll die Scheiben des Fahrzeugs abhängig von der äußeren Lichtintensität bis zu einem Tönungsgrad von n% tönen.* $n \in \{0, 25, 50, 75\}$
Abgeleitet aus Szenarien:
USC03 (Tönung abhängig von der äußeren LI – sonnig)
USC04 (Tönung abhängig von der äußeren LI – schattig)
USC05 (Tönung abhängig von der äußeren LI – halbdunkel)
USC06 (Tönung abhängig von der äußeren LI – dunkel)
Anmerkung: *LI steht für Lichtintensität*
sonnig, d.h. $LI \geq 30000\ Lux \rightarrow n = 75$
schattig, d.h. $LI \geq 15000\ Lux \rightarrow n = 50$
halbdunkel, d.h. $LI \geq 5000\ Lux \rightarrow n = 25$
dunkel, d.h. $LI \leq 3500\ Lux \rightarrow n = 0$

Schritt 6: Modellierung des Nutzungskontextes

Der Schritt 6 der RE-CAWAR Methodik befasst sich mit der Modellierung des Kontextes angeführt von der Identifikation der Nutzungssituationen der Anwendung. Nach der Modellierung der Anwendungsszenarien erfolgt in diesem Schritt die Modellierung der Teilaspekte des Kontextes. Die Kontextmodellierung wird von einer zielgerichtete Anreicherung der Szenarien mit Kontextinformationen begleitet, welche die jeweiligen Nutzungssituationen charakterisieren.

Der gesamte Schritt der Modellierung des Nutzungskontextes ist komplex und wird in feinere Schritte untergliedert, welche iterativ durchgelaufen werden. Abbildung 5.7 stellt eine Übersicht der feineren Schritte der Kontextmodellierung dar. Diese Schritte erfolgen iterativ, so dass die Erkenntnisse aus dem einen Schritt in den anderen Schritt einfließen und umgekehrt.

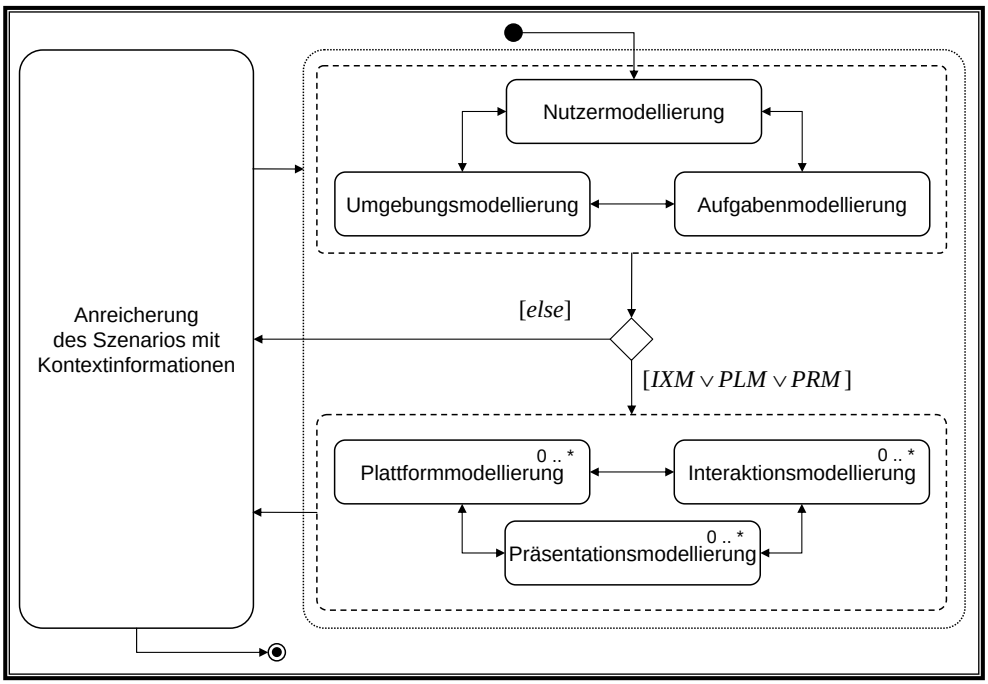

Abbildung 5.7: RE-CAWAR Schritt 6: Modellierung des Nutzungskontextes

Für jedes Szenario werden charakteristische Kontextinformationen identifiziert. Diese Informationen werden entsprechend den Bestandteilen des Kontextes klassifiziert und dienen der Erstellung der jeweiligen Teilmodelle des Kontextes (siehe Abbildung 5.8). Das betroffene Szenario wird dabei mit Kontextinformationen zur Charakterisierung der Nutzungssituationen angereichert (siehe Abbildung 5.9).

Wir erinnern an dieser Stelle, dass eine Besonderheit kontextsensitiver Anwendungen die Berücksichtigung des Kontextes ist, in dem sie genutzt werden. Der Kontext beeinflusst das beobachtbare Verhalten der Anwendung. Die Spezifikation der Anforderungen an die Anwendung geht Hand in Hand mit der Modellierung des Kontextes. Die zu realisierenden Reaktionsmuster der Anwendung sind entsprechend der Ausprägungen des Kontextmodells zu spezifizieren. Dementsprechend ist Schritt 6 von großer Bedeutung in der Methodik.

a- Modellierung des Kontextes

Die Grundlagen der Kontextmodellierung in der RE-CAWAR Methodik wurden mit dem in Kapitel 4 konzipierten Framework gelegt. Dabei bedeutet Kontextmodellierung, Annahmen über die möglichen Bestandteile des Kontextes explizit in entsprechende Modelle abzubilden.

Um systematisch vorgehen zu können, werden die möglichen Bestandteile zunächst getrennt voneinander modelliert. Die Integration erfolgt in einem weiteren Schritt der Methodik. Abbildung 5.8 stellt eine Übersicht der Teilmodelle des Nutzungskontextes dar. Eine kurze Beschreibung der möglichen Inhalte der Teilmodelle ist dabei angegeben. Diese Beschreibungen sind Grundlage der Fragen, die pro Nutzungsszenario gestellt werden, um die Kontextinformationen zu identifizieren und die Teilmodelle zu erstellen.

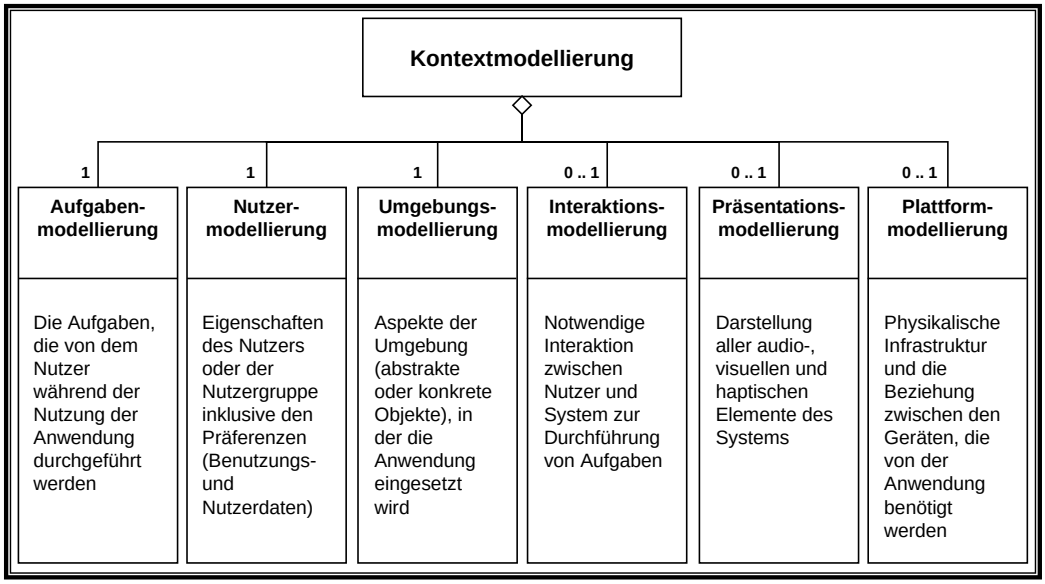

Abbildung 5.8: Kontextmodellierung in der RE-CAWAR Methodik

Als Teil der Kontextmodellierung gilt die Nutzermodellierung, die Aufgabenmodellierung, die Umgebungsmodellierung, die Plattformmodellierung, die Interaktionsmodellierung sowie die Präsentationsmodellierung. Die letzten drei Bestandteile sind im Gegensatz zu den ersten drei nicht zwingend und können weggelassen werden. Eine ausführliche Beschreibung dieser Modelle inkl. ihrer möglichen Inhalte und Beispiele ist im Kapitel 4 gegeben.

Jeder Teil der Kontextmodellierung besteht darin, die entsprechenden Teilmodelle des Kontextes zu erarbeiten. Dazu werden die ausgearbeiteten Szenarien sorgfältig betrachtet, und zielgerichtet mit notwendigen Kontextinformationen angereichert. Dabei wird pro Szenario gezielt nach Informationen gefragt, welche in die Teilmodelle abgebildet werden können. Sukzessiv werden die erhaltenen Informationen strukturiert und abstrahiert.

Bei der Abstraktion werden geeignete Typen der in den Informationen vorkommenden Entitäten, ihre Attribute und die Beziehungen zwischen ihnen, die Nutzeraufgaben und ihre charakteristische Merkmale als Attribute identifiziert.

Aus dieser Modellierungsaktivität ergeben sich die Artefakte *Nutzermodell, Aufgabenmodell, Umgebungsmodell*, und eventuell auch die Artefakte *Plattformmodell, Interaktionsmodell und Präsentationsmodell* (siehe Abschnitte 4.4, 4.5, 4.6 und 4.7). Die Integration dieser Modelle in ein umfassendes Kontextmodell der Anwendung folgt in dem nächsten Schritt der RE-CAWAR Methodik. Das grundlegende Konzept zur Anreicherung der Szenarien mit den notwendigen Kontextinformationen ist im Folgenden beschrieben.

b- Anreicherung der Szenarien mit Kontextinformationen

Anwendungsszenarien, wie sie im Schritt 4 der RE-CAWAR Methodik modelliert wurden, sind ideale Mittel zur Konkretisierung der meist abstrakten Stakeholder-Ziele [Amyot und Eberlein, 2003, Pohl und Haumer, 1997]. Sie modellieren Abfolgen von Aktionen, die zwischen den beteiligten Akteuren (hier Nutzer und Anwendung) stattfinden. Wir sind allerdings im RE kontextsensitiver Anwendung an zusätzlichen Informationen interessiert, welche den Kontext, in dem die Aktionen eines jeden Szenarios stattfinden, charakterisieren.

Obwohl Vor- und Nachbedingungen in einem Szenario in gewisser Hinsicht Informationen über den Kontext des Szenarios beschreiben (siehe z.B. [Regnell et al., 1996]), reichen sie jedoch für die Charakterisierung der Nutzungssituationen kontextsensitiver Anwendungen nicht aus. Vor- und Nachbedingungen beschreiben lediglich die Zustände der beteiligten Akteure (Benutzer oder weiteren Systeme). Umfangreiche Kontextinformationen werden in den Szenarien nicht berücksichtigt. Es ist jedoch erforderlich, den Kontext, in dem eine kontextsensitive Anwendung genutzt wird, bereits im RE zu berücksichtigen und zu modellieren. Mit einer expliziten Modellierung des Nutzungskontextes bietet RE-CAWAR eine Möglichkeit dieser Forderung nachzukommen.

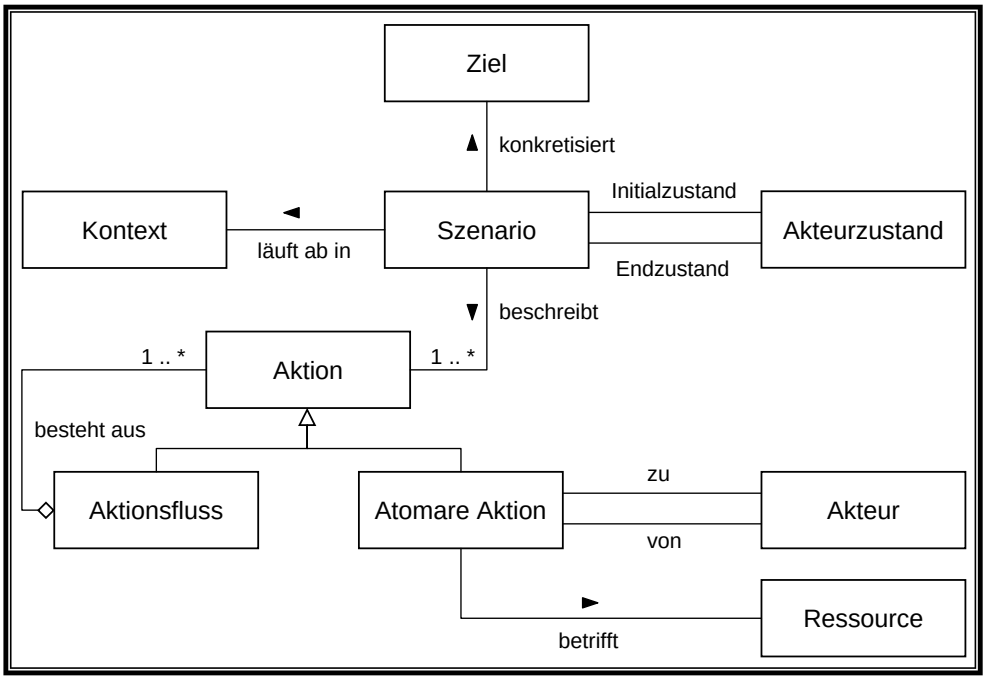

Abbildung 5.9: Anreicherung von Szenarien mit expliziten Kontextinformationen

Konzeptuell erweitern wir das grundlegende Modell der Anwendungsszenarien aus Abbildung 5.6 um das Konzept *Kontext* (siehe Abbildung 5.9). Aufgrund der Übersichtlichkeit lassen wir in dieser Abbildung jegliche Detaillierung des Kontextes weg. Hinter den Kontext steckt jedoch ihre bislang betrachteten möglichen Bestandteile, d.h. den Nutzer, seine durchgeführten Aufgaben, die Umgebung der Nutzung der Anwendung, etc. (siehe Abbildung 4.6). Für die Anreicherung der Szenarien verwenden wir das Template 5.4.

Template 5.4 *Schablone für erweiterte Szenarien*

Bezeichner: *ISCmn:* $m, n \in \mathbb{N}$
Kurzbezeichnung: *Eine aussagekräftige Bezeichnung für das Szenario*
Ziel: *Das Ziel, das mit dem Szenario konkretisiert wird*
Kontext: *Die Aktionen in diesem Szenario finden im folgenden Kontext statt:*
1. *Kontextinformation (z.B. Fakten über den Nutzer)*
2. *Kontextinformation (z.B. Fakten über die vom Nutzer durchgeführten Aufgaben)*
3. *Kontextinformation (z.B. Fakten die Umgebung der Nutzung der Anwendung)*
4. *...*

Vorbedingung: *Vor Anstoß der Szenario-Schritte gilt:*
1. *Bedingung*
2. *...*

Aktionen: *Folgende Aktionen (Szenario-Schritte) werden ausgeführt:*
1. *Aktion*
2. *...*

Nachbedingung: *Nach Ablauf der Szenario-Schritte gilt:*
1. *Bedingung*
2. *...*

Anmerkung: *Jegliche Anmerkung zu dem erweiterten Szenario.*

Die Anreicherung der Szenarien mit Kontextinformationen erfolgt strukturiert nach den in Abbildung 5.8 differenzierten möglichen Bestandteilen eines Kontextmodells. Das Szenario wird sukzessiv mit Informationen bzgl. der Teilaspekte des Kontextes angereichert. Zunächst werden nach Informationen bzgl. des Nutzers, inkl. Nutzerdaten und Nutzungsdaten, gefragt. Das Szenario wird mit den identifizierten Informationen angereichert. Gleichzeitig werden die Typen der Informationen im Sinne der Nutzermodellierung (siehe 4.4) abstrahiert. Selbige gilt für die Informationen bzgl. der Aufgaben, die der Nutzer durchführt, sowie für die Informationen über die Umgebung, in der die Anwendung eingesetzt/genutzt wird. Falls zutreffend wird das Szenario ebenfalls auf die selbe Art mit Informationen angereichert, welche die Plattform, auf der die Anwendung laufen wird, die benötigten Interaktionen zur Durchführung der Aufgaben sowie die Modalität der Darstellung der Interaktionen charakterisieren.

Damit die Identifikation der anreichernden Informationen systematisch abläuft, werden die einzelnen Informationen jeweils in ein Nutzermodell, ein Aufgabenmodell,

ein Umgebungsmodell, ein Plattformmodell, ein Interaktionsmodell und ein Präsentationsmodell abgebildet. Genauere Beschreibungen der Informationsgehalte dieser Modelle einschließlich Beispiele und jeweils geeigneter Beschreibungstechniken sind im Kapitel 4 aufgeführt. Jedes im Schritt 4 der Methodik modellierte Anwendungsszenario wird mit den Kontextinformationen angereichert. Mit jedem Anreicherungsschritt werden die Teilmodelle des Kontextes präziser und letztendlich vollständiger.

Alle Kontextinformationen, die in einem angereicherten Szenario vorkommen, charakterisieren zusammen die Nutzungssituation, in der das beschriebene Szenario stattfindet. Durch das explizite Hinzufügen der Kontextinformationen können einzelne Aktionen aus den ursprünglichen Szenarien (Interaktionen zwischen den Akteuren) als optional eingestuft werden, und eventuell bei der späteren Konsolidierung der Ergebnisse wegfallen (siehe Beispiel 5.7).

Beispiel 5.7 *Mit Kontextinformationen angereichertes Szenario am Beispiel des kontextsensitiven Scheibentönungssystems*

Bezeichner *USC01*
Kurzbezeichnung: *Die Scheiben des geparkten Fahrzeugs vollständig tönen*
Ziel: *G01 - Einblick von außen in das Fahrzeug beim geparkten Fahrzeug verhindern*
Kontext: *Die Aktionen finden im folgenden Kontext statt:*
1. *Der Fahrer bevorzugt es, die Scheiben seines geparkten Fahrzeugs bis zu einer Lichtdurchlässigkeit von 0% zu tönen.*
2. *Der Fahrer parkt das Fahrzeug ein und er verlässt es.*
3. *Das Fahrzeug befindet sich auf einem Stellplatz.*

Vorbedingung:
1. *Das Fahrzeug ist geparkt*
2. *Das Scheibentönungssystem ist aktiv*

Aktionen:
1. ***Optional:** Der Fahrer drückt auf den Knopf zur Auslösung der vollständigen Tönung der Scheiben*
2. *Das System tönt die Scheiben des Fahrzeugs bis zu einer Lichtdurchlässigkeit von 0%*

Nachbedingung:
1. *Die Scheiben des Fahrzeugs sind getönt*
2. *Die Lichtdurchlässigkeit bei den Scheiben des Fahrzeugs ist 0%*
3. *Das Scheibentönungssystem ist aktiv*

Anmerkung: *Vollständige Tönung der Scheiben := Tönung bis zu einer Lichtdurchlässigkeit von 0%.*

Wie in Abbildung 5.9 dargestellt, hängen wir den Kontext direkt an das Szenario und nicht etwa an die einzelnen Aktionen. Dies scheint auf den ersten Blick etwas problematisch zu sein. Denn ein Szenario besteht aus einer Folge von Aktionen und es sind die einzelnen Aktionen, die in einem Kontext stattfinden und nicht notwendigerweise das gesamte Szenario. Es ist durchaus möglich, dass zwei Aktionen eines Szenarios in

unterschiedlichen Kontexten stattfinden. In solche Fälle ist es empfehlenswert, entweder die Kontextinformationen allgemein genug zu halten oder das Szenario fein genug (d.h. mit wenigen aber dafür in einem gleichen Kontext stattfindenden Aktionen) zu modellieren. An dieser Stelle kommt es auf die Granularität der Szenarien an.

Bei der Ermittlung der Nutzungssituationen ist zu beachten, dass jede Nutzungssituation mit so wenig wie möglich und so viel wie nötig Kontextinformationen zu charakterisieren ist. Man soll sich also auf das Wesentliche beschränken. Die Anknüpfung des Konzeptes *Kontext* an das Konzept *Aktion* ist grundsätzlich nicht falsch. Allerdings skaliert dieser Ansatz nicht gut (für 10 Aktionen in einem Szenario würden im schlimmsten Fall 10-fache Angaben der selben Kontextinformationen erforderlich sein, was aber ineffizient ist).

An dem Beispiel 5.7 stellen wir fest, dass durch Hinzunahme der Kontextinformationen gewisse Aktionen wegfallen bzw. nicht mehr zwingend sind. In diesem Beispiel ist die Aktion *„Der Fahrer drückt auf den Knopf zur Auslösung der vollständigen Tönung der Scheiben“* aus dem ursprünglichen Szenario (siehe Beispiel 5.5) optional geworden. Diese Feststellung ist ein Zeichen dafür, dass indem das Szenario mit Kontextinformationen angereichert wird, d.h. indem die Kontextsensitivität der Anwendung explizit gemacht wird, die Menge der expliziten Aktionen (insbesondere der expliziten Nutzeraktionen) offensichtlich kleiner wird.

Schritt 7: Integration der Ergebnisse

Nach Durchführung des sechsten Schritts der RE-CAWAR Methodik enthält die Artefakten-Basis unter anderem:

- Eine Liste funktionaler Anforderungen, mit Angabe der Szenarien, aus den sie abgeleitet sind.
- Eine Liste von Anwendungsszenarien, mit Angabe von Stakeholder-Zielen, die sie konkretisieren. Die Anwendungsszenarien sind in diesem Statium der RE-CAWAR Methodik mit Kontextinformationen angereichert.
- Einzelne Teilmodelle des Kontextes: ein Nutzermodell, ein Aufgabenmodell, ein Umgebungsmodell und eventuell ein Plattformmodell, ein Interaktionsmodell sowie ein Präsentationsmodell

Die Kontextinformationen in den einzelnen Szenarien charakterisieren jeweils eine Nutzungssituation der Anwendung. Ziel des siebten Schritts der Methodik ist die Integration des bisherigen Artefakte, welche zu den Endergebnissen der RE-CAWAR Methodik führt. Die Endergebnisse sind das umfassende Kontextmodell der Anwendung und die Zusammensetzung der Nutzungssituationen und der funktionalen Anforderungen mit Angabe notwendiger Illustrationen. In diesem Schritt wird also zum einen das Kontextmodell der Anwendung als Integration der aufgeführten Teilmodelle erstellt. Zum anderen werden die Anforderungen und die Nutzungssituationen mit einander integriert, um die Contextual Requirements Chunks zu erstellen.

a- Erstellung des Kontextmodells der Anwendung

In diesem Stadium der Anwendung der Methodik sind hinreichende Annahmen über den Nutzer, seine Aktivitäten, die Umgebung, die Interaktionen, die Darstellungsmög-

lichkeiten und die Infrastruktur bereit in den Teilmodellen des Kontextes abgebildet.

Die Erstellung des Kontextmodells der Anwendung besteht darin, die existierenden Teilmodelle des Kontextes miteinander zu integrieren. Die Teilmodelle des Kontextes bilden spezifische Annahmen über die Teilaspekten des Kontextes ab. Sie beschreiben die Trägermengen $USR, TSK, ENV, ITX, PREundPLT$ der Informationen, die in den angereicherten Szenarien enthalten sind.

Für die Zwecke der Integration der Teilmodelle setzen wir die im Kapitel 4.8 aufgeführten Konzepte von Fakten in einer Situation ein. Dort haben wir bereits aufgeführt, dass jede Nutzungssituation durch eine Reihe von Fakten charakterisiert ist. In diesem Zusammenhang spiegeln die Fakten mögliche Wechselwirkungen zwischen den Teilmodellen des Kontextes wider.

Für die Erstellung des Kontextmodells haben wir ein konzeptuelles **Kontextmodellierungsschema** *(engl. Conceptual Context Modelling Schema – CCMS)* aufgesetzt, welches iterativ durchgelaufen wird. Das CCMS besteht aus den folgenden Schritten:

1. **Identifikation und Transformation**
   Geläufige Beispiele für Wechselwirkungen zwischen den Teilmodellen des Kontextes identifizieren, und die Beispiele als elementare Fakten ausdrücken.
2. **Darstellung und Validierung**
   Das Modells der Fakttypen (Faktenmodell) graphisch darstellen, und es mittels Beispiele von Fakten validieren.
3. **Optimierung**
   Das Faktenmodell optimieren: Zusammenführung, Zerlegung und Ableitung von Fakten.

Abbildung 5.10 stellt die Schritte des konzeptuellen Kontextmodellierungsschemas dar. Einzelheiten zu den CCMS-Schritten werden im Folgenden gegeben.

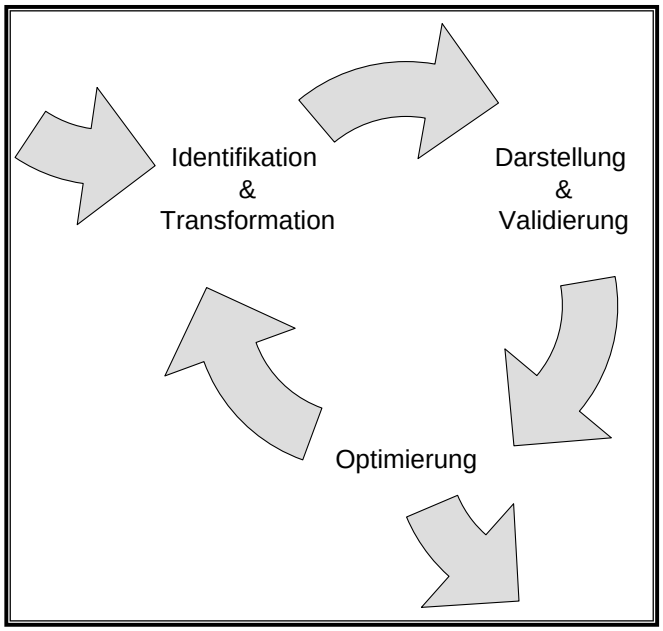

Abbildung 5.10: Konzeptuelles Kontextmodellierungsschema (CCMS)

Das vorgeschlagene konzeptuelle Kontextmodellierungsschema setzt voraus, dass der sechste Schritt der RE-CAWAR Methodik bereit durchgeführt wurde. Dies impliziert, dass die Teilmodelle des Kontextes bereit erstellt sind. Der Einsatz von Object-Role-Modeling als Modellierungstechnik wird ebenfalls als Voraussetzung gesehen.

1. **Identifikation und Transformation**

Die Wechselwirkungen zwischen den Teilmodellen bilden grundlegende Beziehungen zwischen ihnen. Ihre Identifikation erfordert gute Kenntnisse der Anwendungsdomäne. Es ist daher notwendig, dass Experte der Anwendungsdomäne (Personen mit guten Kenntnissen der Anwendungsdomäne) bei der Identifikation von Beispielen einbezogen werden. Die Experten sollen die Qualität der Beispiele prüfen. Es geht an dieser Stelle um die Identifikation konkreter Datensätze, ohne dabei an ihre Struktur und Form zu denken. Die Identifikation von Beispielen ist allerdings eng verbunden mit der Transformation dieser Beispiele in elementare Fakten.

Bei Transformation werden die identifizierten Beispiele für Wechselwirkungen in elementare Fakten ausgedrückt. Elementare Fakten sind der Art *„Ein Objekt hat eine Eigenschaft“* oder *„Ein oder mehrere Objekte sind in einer Beziehung beteiligt“*, wobei die Beziehung nicht als Konjunktion weiterer Fakten ausgedrückt werden kann. Eine Aussage wie „Der Fahrer drückt auf dem DST-Schaltknopf und steigt aus dem Fahrzeug“ stellt eine Konjunktion von zwei elementaren Fakten dar und muss entsprechend behandelt werden.

Als Richtlinie für die Durchführung des CCMS-Schritts der Identifikation und Transformation gilt, dass die Wechselwirkungen zwischen den Teilmodellen des Kontextes folgende Muster haben:

a) Der Nutzer u führt die Aufgabe t durch.

b) Die Durchführung der Aufgabe t erfolgt unter den Gegebenheiten (ex, ey, ez) der Einsatzumgebung.

c) Die Durchführung der Aufgabe t erfordert die Interaktion i.

d) Die Interaktion i findet zwischen dem Nutzer u und dem System statt.

e) Während der Interaktion i wird das Darstellungsobjekt po manipuliert.

f) Während der Interaktion i wird das Umgebungsobjekt eo manipuliert.

g) Die Interaktion i erfordert die Systeminfrastruktur p.

Dabei sind $u, t, ex, ey, ez, i, po, eo, p$ aus den aufgeführten Teilmodellen des Kontextes entnommen, und behalten demzufolge jeweils weiterhin ihre zuvor modellierten Eigenschaften.

Die Muster der Wechselwirkungen zwischen den Teilmodellen (und dementsprechend ihrer Zusammensetzung) sind wie folgt zu verstehen: Das zu entwickelnde System dient dazu, der Nutzer bei der Durchführung seiner Aufgaben in gegebenen Situationen zu unterstützen. Die Aufgaben werden nur unter bestimmten Umgebungsgegebenheiten durchgeführt. Die Durchführung einer Aufgabe kann dazu führen, dass Interaktionen zwischen dem Nutzer und dem System stattfinden. Dabei werden Objekte manipuliert. Diese Objekte können sowohl Teil der Umgebung, als auch Teil des unterstützenden Systems sein. Die Objekte, die Teil des Systems sind, werden für die

Darstellung (audio, visuell, haptisch) der Kommunikation zwischen dem Nutzer und dem System verwendet. Die Anwendung läuft auf einer Infrastruktur, welche eine Reihe von Eigenschaften hat, die ebenfalls für das adaptive Verhalten des Systems relevant sind.

Jede Zusammensetzung der Teilmodelle stellt einen Fakt dar, und jede Art der Zusammensetzung der Teilmodelle stellt einen Fakttyp dar. Die Typen der Fakten entnehmen wir daher der Zusammensetzung der Teilmodelle des Kontextes. Das Beispiel 5.8 illustriert den CCMS-Schritt der Identifikation und Transformation am Beispiel des Scheibentönungssystems.

Beispiel 5.8 *CCMS-Schritt der Identifikation und Transformation*

Identifikation

1. *Herr Weber parkt die Limousine FL112 auf den Stellplatz ein*
2. *Frau Müller parkt das Geländewagen FG201 in die Tiefgarage ein*
3. *Es befindet sich keine Person in der Limousine FL112*
4. *Es befindet sich 2 Person in dem Geländewagen FG201*
5. *Herr Weber fährt die Limousine FL112 durch den Tunnel*
6. *Herr Weber fährt die Limousine FL112 auf die offene Straße*

Transformation

1. ***Fahrer (Name)** parkt **Fahrzeug (Typ)** in **Garage (Typ)** ein.*
2. *Im **Fahrzeug (Typ)** befindet sich **Personen (Anzahl)**.*
3. ***Fahrer (Name)** fährt **Fahrzeug (Typ)** auf **Straße (Typ)**.*

Aus Gründen der Übersichtlichkeit und für die Erleichterung der Diskussion mit Experten der Anwendungsdomäne werden die Beziehungen (die logischen Prädikate) und die betroffenen Objekte/Entitäten (die Substantive) unterschiedlich hervorgehoben. In unserem Beispiel sind die Substantive **fett** markiert. Die Entitäten werden jeweils mit Schlüsseln versehen (jeweils in Klammer), welche sie in dem betrachteten Fakttyp eindeutig charakterisieren.

2. **Darstellung und Validierung**

Die aus der Transformation der identifizierten Beispiele in Fakttypen sind in dem CCMS-Schritt der Darstellung und Validierung zunächst graphisch abzubilden, und anschließend mit Hilfe von Beispielen zu validieren. Ein oder mehrere Fakten charakterisieren eine Nutzungssituation. Die Typen der Nutzungssituationen sind ungekürzt in dem Kontextmodell enthalten. Die identifizierten Fakttypen inklusive den daraus bestehenden Objekttypen, Prädikaten und Rollen bilden das umfassende Kontextmodell der Anwendung. Die graphische Darstellung des Modells erfolgt mittels dem Object-Role-Modeling. Eine kurze Einführung der für unsere Zwecke notwendige Notation ist im Abschnitt 4.8.2 gegeben. Die graphische Darstellung der Fakttypen (Faktenmodell/Kontextmodell der Anwendung) aus dem Beispiel 5.8 mit weiteren Fakttypen aus der Fallstudie der Scheibentönungssystems ist in der Abbildung 5.11 gegeben.

Für die Validierung des Faktenmodells werden Beispielsätze (Data Use Cases nach der ORM-Terminologie) angegeben. Dafür ist es wichtig, dass das Modell systematisch

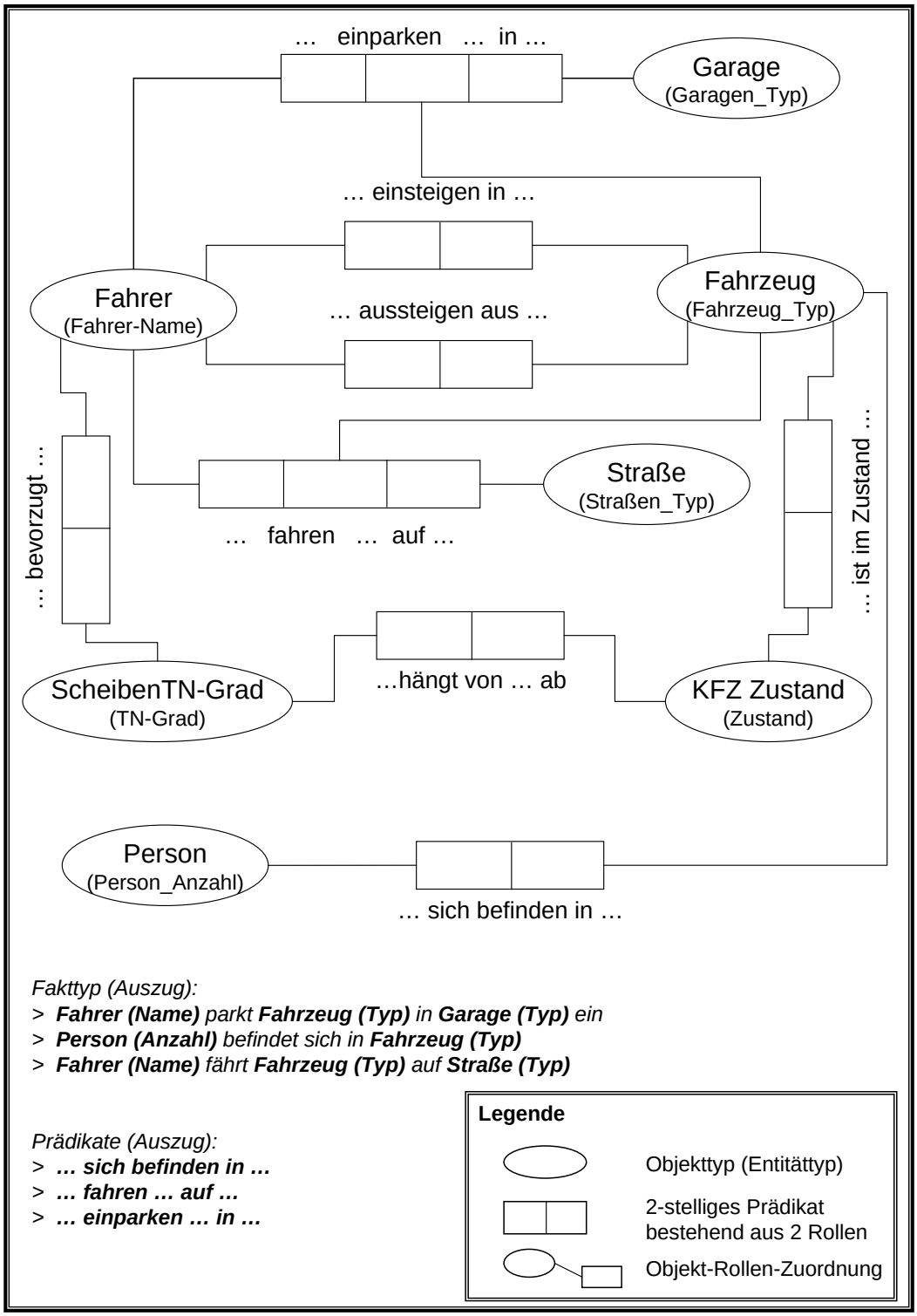

Abbildung 5.11: Graphische Darstellung des Faktenmodells

abgelesen wird. Man konzentriert sich dabei auf die Prädikate und gibt mindestens einen Beispielsatz an. Die Prädikate werden von links nach rechts und von oben nach unten abgelesen, es sei denn sie sind mit „«“ vorangestellt. In diesem Fall müssen

sie in der umgekehrten Richtung abgelesen werden. Die Angabe der Beispielsätze ist lediglich für die Validierung des Modells notwendig. Die Beispielsätze sind nicht Teil des eigentlichen Modells und dürfen nicht als solchen interpretiert werden. Abbildung 5.12 illustriert ansatzweise die Validierung des Faktenmodells aus Abbildung 5.11.

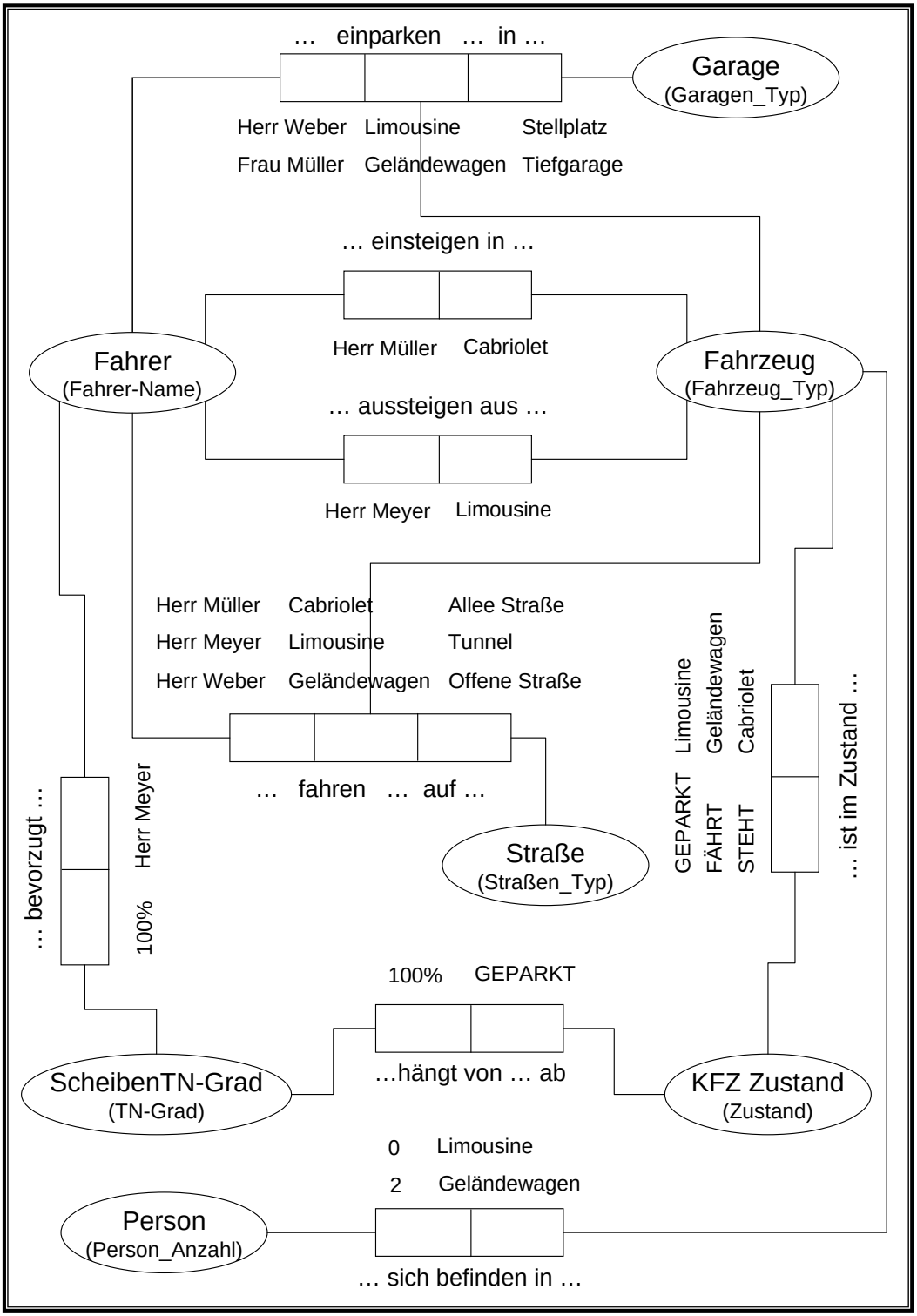

Abbildung 5.12: Validierung des Faktenmodells mittels Data Use Cases

3. **Optimierung**

Fakten, und auch Fakttypen, können zusammengesetzt werden. Die Fakttypen können beliebig lang werden[3]. Allerdings wird bei der Erstellung des Faktenmodells empfohlen, Fakttypen mit mehr als 4 Rollen zu vermeiden. Solche Prädikate sind im Zuge der Optimierung des Modells in kleinere Prädikate (d.h. mit einer Stelligkeit kleiner oder gleich 4) zu zerlegen. Im Zuge von Diskussionen mit den Experten der Anwendungsdomäne ist die angemessene Granularität der Fakten zu bestimmen. Dabei können existierende Fakttypen zusammengesetzt oder insbesondere voneinander getrennt werden. Dadurch werden dem Kontextmodell Fakttypen hinzugefügt bzw. entfernt.

Beispielsweise werden die Fakttypen

(1) ***Fahrer (Name)*** *befindet sich im* ***Fahrzeug (Typ)***

(2) ***Mitfahrer (Name)*** *befindet sich im* ***Fahrzeug (Typ)***

im Zuge der Optimierung in den Fakttyp

(3) ***Person (Anzahl)*** *befindet sich im* ***Fahrzeug (Typ)***

umgewandelt. Dabei wird angemerkt, dass es auf die Anzahl der Personen, die sich in dem Fahrzeug befinden, ankommt, und nicht etwa um die eigentlichen Personen. Eine weitere Optimierungsmöglichkeit ist die Feststellung von Fakttypen, die arithmetisch oder logisch aus anderen Fakttypen abgeleitet werden. In solche Fälle sind die Ableitungsregeln anzugeben. Beispielsweise, wenn die Fakttypen ***Mitfahrer (Name)*** *befindet sich im* ***Fahrzeug (Typ)*** und ***Fahrzeug (Typ)*** *ist im Zustand „FÄHRT“*[4] in unserem Modell enthalten sind, dann können wir daraus den Fakttyp ***Fahrer (Name)*** *befindet sich im* ***Fahrzeug (Typ)*** ableiten[5]. Dies kann auch für die arithmetische Berechnung der Anzahl der Personen gelten, die sich in dem Fahrzeug befinden.

Nach jedem Optimierungsschritt wird gefordert, dass das Modell angepasst und erneut mittels Beispiele von Fakten validiert wird. Dabei soll seine Konsistenz durch eine Validierung mittels Beispielsätze wie in CCMS-Schritt der Darstellung und Validierung sichergestellt sein.

b- Erstellung der Contextual Requirements Chunks

Das zweite Endergebnis der RE-CAWAR Methodik ist die Zusammenstellung der Contextual Requirements Chunks (CRCs). Die CRCs entstehen aus der Integration der funktionalen Anforderungen mit den in den erweiterten Szenarien enthaltenen charakteristischen Kontextinformationen der Nutzungssituation. Aufgrund der Tatsache, dass

- bei den funktionalen Anforderungen, die Szenarien angegeben sind, aus den sie abgeleitet sind, und
- die charakteristischen Kontextinformationen der Nutzungssituationen bei der Erweiterung der selben Szenarien entstehen und dort angegeben sind,

[3] Die Länge der Fakttypen hängt von der Anzahl der Entitätstypen ab, die an der Beziehung beteiligt sind. Sie ist gleich der Anzahl der Rollen, die in den Prädikaten vorkommen.

[4] Die zulässigen (mit den Experten der Anwendungsdomäne abgestimmt) Zustände des Fahrzeugs hinsichtlich des Scheibentönungssystems sind {GEPARKT, FÄHRT, STEHT}.

[5] Diese Ableitung gilt nur unter „normalen“ Bedingungen. Wir schließen an dieser Stelle aus, dass das Fahrzeug etwa ohne Fahrer in Bewegung sein darf.

erfolgt ihre Integration über die (Referenzen der) Szenarien. Die Anforderungen werden jeweils mit Angaben über eine Nutzungssituation und ein illustrierendes Szenario zusammengefasst. Das Ergebnis dieser Zusammenfassung ist eine Liste mit Contextual Requirements Chunks. Darin sind die Nutzungssituationen, die funktionalen Anforderungen und die illustrierenden Szenarien abgebildet. Die CRCs werden dementsprechend in einer Tabelle zusammengefasst, wobei jede Zeile der Tabelle einen CRC darstellt. Abbildung 5.13 illustriert den Aufbau eines CRCs, die Zusammensetzung aller CRCs, und die Rolle des Kontextmodells bei dieser Zusammensetzung.

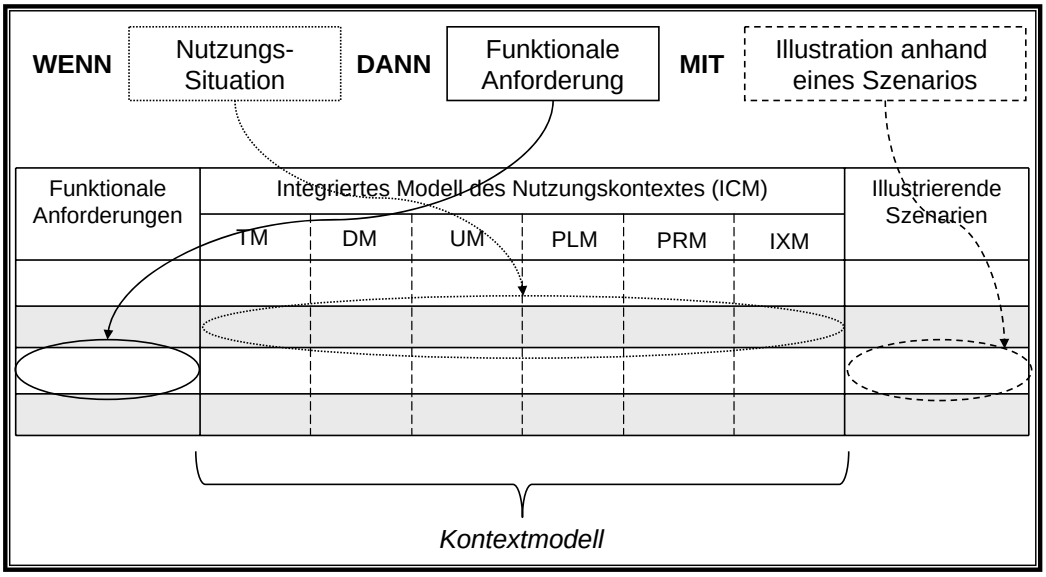

Abbildung 5.13: Contextual Requirements Chunks in RE-CAWAR

Das allgemeine Schema eines Contextual Requirements Chunks ist:

WENN $Nutzungssituation$ **DANN** $Funktionale\ Anforderung$
MIT $Illustration\ anhand\ eines\ Szenarios.$

Hinter den Nutzungssituationen steckt das integrierte Kontextmodell der Anwendung, welches die grundlegende Struktur der charakteristischen Informationen der Nutzung der Anwendung festlegt. Sie kommen gemeinsam mit den funktionalen Anforderungen im zugehörigen illustrierenden Szenario vor. Die Kontextinformationen aus den erweiterten Anwendungsszenarien sind konkrete Beispiele von Fakten. Jede Nutzungssituation ist mit Fakten über Aspekte des Kontextes der Anwendung eindeutig charakterisiert[6].

[6]Der Typ der Fakten in einer Nutzungssituation sollten in dem Kontextmodell enthalten sein. Andernfalls ist das Modell nicht ausreichend detailliert und muss um fehlende Objekttypen, Prädikate und/oder Rollen erweitert werden. Die Fakten werden auch Data-Use-Cases des Kontextmodells genannt. Bei der Konsolidierung der Ergebnisse kommen wir auf die Fragestellung der Abdeckung der Nutzungssituationen durch das Kontextmodell zurück.

Die Szenarien illustrieren beispielsweise anhand eines kurzen (strukturierten) Textes[7] die Nutzung der Systemfunktionen, welche die funktionalen Anforderungen realisieren, in den angegebenen Nutzungssituationen. Ein Szenario enthält die Basisinformation zur Nutzung des Systems, aus der Anforderungen und Nutzungssituationen abgeleitet werden.

Das Beispiel 5.9 zeigt ein konkretes Contextual Requirements Chunk aus der Fallstudie des kontextsensitiven Scheibentönungssystems.

Beispiel 5.9 *CRC-12: Scheiben bis 0% Lichtdurchlässigkeit tönen*

Funktionale Anforderung – Req. RQ12
Das System soll die Scheiben bis zu einer Lichtdurchlässigkeit von 0% tönen.

Nutzungssituation – Sit. US12

a) ***UM09:*** *Der Fahrer ist Herr Meyer;*
Herr Meyer bevorzugt einen Tönungsgrad von 100% (d.h. Lichtdurchlässigkeit ist 0%) für die Scheiben seines Fahrzeugs, wenn es geparkt ist;
b) ***TM09:*** *Herr Meyer parkt sein Fahrzeug ein;*
Herr Meyer steigt aus dem Fahrzeug aus;
c) ***EM10:*** *Das Fahrzeug ist im Zustand „GEPARKT“;*

Illustrierendes Szenario – ISce. ISC12
Herr Meyer parkt sein Fahrzeug auf einem Stellplatz ein und verlässt es. Aufgrund seiner Vorliebe die Scheiben seines Fahrzeugs im geparktem Zustand bis zu einer Lichtdurchlässigkeit von 0% zu tönen, tönt das Systems die Scheiben des Fahrzeugs bis zu einer Lichtdurchlässigkeit von 0%.

Die erarbeiteten Contextual Requirements Chunks und das Kontextmodell der Anwendung sind im nächsten Schritt der RE-CAWAR Methodik zu konsolidieren.

5.3.3 Konsolidierungsaktivitäten

In diesem Teil der Methodik erfolgt die Konsolidierung der bisherigen Ergebnisse. Ferner werden in diesem abschließenden Teil der RE-CAWAR Methodik die Abarbeitung jeglicher Feedback eingeleitet und notwendige Iterationen durchgeführt. Abbildung 5.14 stellt eine Übersicht dieses Methodikteils dar.

[7] Es ist an dieser Stelle möglich gängige Notationen von Szenarien einzusetzen, die häufig Einsatz im RE finden (siehe [Amyot und Eberlein, 2003] für eine Evaluation eine Reihe von Szenario-Notationen). Beispielsweise können *Structured Text [Somé et al., 1995], Message Sequence Charts [ITU-T, 1996] oder UML Sequenz Diagramm bzw. UML Use Case Diagramm [Jacobson, 2004]* eingesetzt werden. Mit ihrer Hilfe werden Abläufe des Systems bzw. Abläufe in der Umgebung oder auch Abläufe zwischen dem System und seiner Umgebung beschrieben.

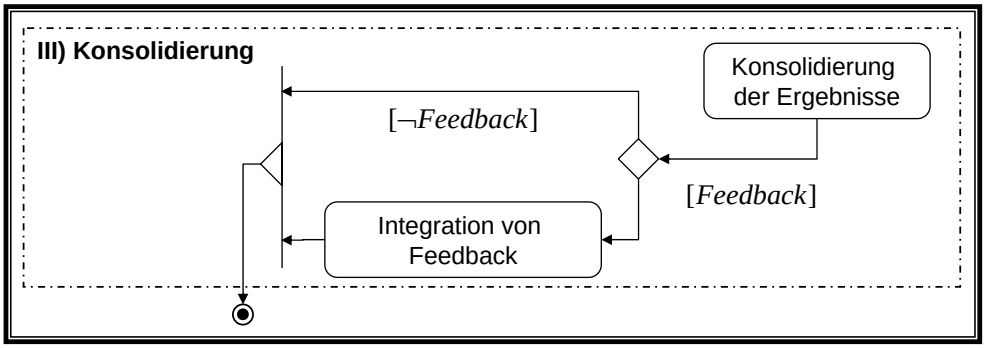

Abbildung 5.14: Integrations- und Konsolidierungsaktivitäten in RE-CAWAR

Schritt 8: Konsolidierung der Ergebnisse

Durch die Konsolidierung werden die Contextual Requiremens Chunks so „bereinigt“, dass sie lediglich notwendige Kontextinformationen zur Charakterisierung der Nutzungssituationen enthalten. „Überflüssige“ Angaben in den charakteristischen Informationen der Nutzungssituation sind auszufiltern.

Darüber hinaus wird im Zuge der Konsolidierung die CRCs auf Widerspruchsfreiheit, bezogen auf die Nutzungssituationen, geprüft. Festgestellte Widersprüche werden falls möglich durch Hinzufügen differenzierender Kontextinformationen beseitigt. Dabei wird das integrierte Kontextmodell der Anwendung auf Angemessenheit geprüft. Demzufolge muss jede Nutzungssituation von dem Kontextmodell abgedeckt sein. Andernfalls ist das Kontextmodell adäquat anzupassen, oder die Nutzung der Anwendung in der betroffenen nicht abgedeckten Nutzungssituation explizit auszuschließen.

Im Folgenden führen wir diese Konsolidierungsaktivitäten auf und illustrieren sie anhand geeigneter Beispiele.

a- Filterung der Contextual Requirements Chunks

Die Formulierung der CRCs, wie sie nach der Durchführung des siebten Schrittes der RE-CAWAR Methodik geschehen ist, ist dafür geeignet Abstimmungen mit den relevanten Stakeholdern, insbesondere mit den Experten der Anwendungsdomäne, durchzuführen. Die CRCs, wie sie in diesem Stadium formuliert sind, enthalten allerdings Informationen über die Nutzungssituationen, die für die weiteren Schritte der eigentlichen Systementwicklung überflüssig sind.

Das CRC-Beispiel 5.9 zeigt beispielsweise, dass die charakteristischen Informationen der Nutzungssituationen nicht kompakt formuliert sind. Eine kompakte Formulierung der Chunks, insbesondere der darin enthaltenen Nutzungssituationen, ist hinsichtlich ihrer weiteren Verarbeitung im Zuge der Systementwicklung erforderlich[8]. Die überflüssigen Angaben in den Situationen werden demzufolge in diesem Teil der Konsolidierung der Ergebnisse ausgefiltert. Danach werden Lediglich die tatsächlich notwendigen charakteristischen Kontextinformationen der jeweiligen Nutzungssituationen in

[8] Ein weiterer Schritt in der Entwicklung kontextsensitiver Systeme betrifft beispielsweise die Spezifikation des adaptiven Verhalten des Systems im Form von Systemkonfigurationen und dem situationsbedingten Übergang zwischen ihnen ausgehend von den Contextual Requirements Chunks.

den endgültigen CRCs aufzuführen. Wir reden an dieser Stelle von einer Filterung oder auch Bereinigung der Contextual Requirements Chunks.

Die Filterung des Contextual Requirements Chunks aus dem Beispiel 5.9 ergibt das im Beispiel 5.10 angegebene CRC. Die Erläuterung, dass ein Tönungsgrad von 100% die Tönung bis zu einer Lichtdurchlässigkeit von 0% entspricht wird an geeigneter Stelle, beispielsweise im Glossar festgehalten.

Die Filterung der CRCs zielt also auf die Elimination überflüssiger Angaben in den Kontextinformationen aus den Contextual Requirements Chunks. Dafür wird das Kontextmodell als Trichter eingesetzt. Jede tatsächliche Kontextinformation muss durch den Trichter gehen. Die Informationen, die dabei ausgefiltert werden, gelten als überflüssig und werden dementsprechend aussortiert. Demnach werden lediglich die notwendigen und tatsächlich im Kontextmodell abbildbaren Informationen in den endgültigen CRCs aufgeführt. Darüber hinaus wird durch den Einsatz des Kontextmodells vermieden, dass eine Kontextinformation mehrfach in der selben Instanz des Modells (d.h. der selben Nutzungssituation) aufgeführt wird.

Beispiel 5.10 *CRC-12 (gefiltert): Scheiben bis 0% Lichtdurchlässigkeit tönen*

Funktionale Anforderung – Req. RQ12
Das System soll die Scheiben bis zu einer Lichtdurchlässigkeit von 0% tönen.

Nutzungssituation – Sit. US12
a) ***UM09:*** *Fahrer-Name = Herr Meyer;*
Bevorzugter Tönungsgrad = 100%;
b) ***TM09:*** *Fahrzeug einparken;*
Aus dem Fahrzeug aussteigen;
c) ***EM10:*** *KFZ_Zustand = GEPARKT;*

Illustrierendes Szenario – ISce. ISC12
Herr Meyer parkt seine Limousine auf dem Stellplatz ein und verlässt das Fahrzeug. Aufgrund seiner Vorliebe die Scheiben seines Fahrzeugs im geparktem Zustand bis zu einer Lichtdurchlässigkeit von 0% zu tönen, tönt das Systems die Scheiben des Fahrzeugs automatisch bis zu einer Lichtdurchlässigkeit von 0%.

b- Beseitigung von Inkonsistenzen

Nach der Integration der Ergebnisse ist es möglich, dass „widersprüchliche“ Zeilen in der CRC-Tabelle stehen (siehe Tabelle 5.1). Die widersprüchlichen Zeilen bedürfen dann einer zusätzlichen Analyse.

Die Ursprünge der Inkonsistenzen in den Chunks sind zweier Natur: entweder werden irrtümlicherweise widersprüchliche Anforderungen in der selben Nutzungssituation gestellt oder die Nutzungssituationen sind fälschlicherweise nicht ausführlich genug charakterisiert. Beispielsweise sind die CRCs *CRC-12 = (RQ12, US12, ISC12)* und

CRC-Nr.	Anforderung	Situation	Szenario
...	...	...	...
...	...	...	...
CRC-12	RQ12	US12	ISC12
CRC-13	RQ13	US12	ISC13
...	...	...	...
...	...	...	...

Tabelle 5.1: Konsolidierung der Contextual Requirements Chunks

CRC-13 = (RQ13, US12, ISC13) widersprüchlich, aufgrund dessen, dass sie für die gleiche Nutzungssituation *US12* zwei Anforderungen *RQ12* und *RQ13* stellen, welche sich gegenseitig ausschließen. Genauere Angaben zu diesen CRCs sind in den Beispielen 5.10 und 5.11 enthalten.

Zur Beseitigung der Inkonsistenzen bietet sich zwei grundsätzliche Möglichkeiten, die einzeln oder kombiniert angewandt werden:

1. Die Anforderungen aus den betroffenen Contextual Requirements Chunks erneut mit den Stakeholder abstimmen.
2. Differenzierende Kontextinformationen in die betrachteten Nutzungssituationen einfügen, und bei Bedarf das Kontextmodell anpassen.

Die erste Möglichkeit impliziert eine Rückkehr zum fünften Schritt der Methodik. Dort wurden die Anforderungen ursprünglich identifiziert. Ihre Quelle sind zu prüfen und adäquate Maßnahmen sind zu ergreifen. Diesen Art von Inkonsistenzen ist nicht spezifisch für das Requirements Engineering kontextsensitiver Anwendungen. Dementsprechend wird diese erste Möglichkeit zur Beseitigung von Inkonsistenzen in den CRCs nicht weiter vertieft, wir verweisen lediglich auf bestehede Literatur zu diesem Thema (z.B. [Spanoudakis und Finkelstein, 1998, Robinson und Pawlowski, 1999] und [Kaiya et al., 2005]).

Die zweite Möglichkeit zur Beseitigung der Inkonsistenzen hingegen ist sehr spezifisch für das RE kontextsensitiver Anwendungen. Kontextinformationen werden angegeben, um die Nutzungssituationen zu charakterisieren. Falls Inkonsistenzen in den CRCs auftreten, die auf unzureichende Kontextinformationen zur Charakterisierung der Situationen zurück zu führen sind, müssen sie noch vor der Freigabe der CRCs beseitigt werden. Dafür sind differenzierende Kontextinformationen einzufügen. Die differenzierenden Kontextinformationen variieren von Kontextinformationen, welche bereit mit dem Kontextmodell abgedeckt werden, bis Kontextinformationen, die bislang nicht in Betracht genommen wurden. Unter Umständen wird die *Historie* oder die *Qualitätsattribute* (z.B. Zuverlässigkeit oder Wahrscheinlichkeit) der betrachteten Kontextinformationen mit berücksichtigt, um eine eindeutige Differenzierung zu erzielen. Dementsprechend wird das Kontextmodell bei Bedarf aktualisiert, d.h. um weitere Fakttypen erweitert oder um bestehende Fakttypen gekürzt.

Bei den Contextual Requirements Chunks aus den Beispielen 5.10 und 5.11 sollte spätestens während der Konsolidierung eine Inkonsistenz festgestellt werden: die angegebenen Kontextinformationen zur Charakterisierung der Nutzungssituationen sind identisch, die geforderten Anforderungen schließen sich jedoch gegenseitig aus (Wi-

derspruch). Die eine Anforderung fordert die Tönung der Scheiben bis zu einer Lichtdurchlässigkeit von 0% und die andere bis zu 50%. Diese Inkonsistenz muss an dieser Stelle beseitigt werden.

Nach Rücksprache mit den Stakeholdern wird zunächst vergewissert, dass diese Anforderungen tatsächlich gewünscht sind, allerdings sind die jeweiligen Nutzungssituationen nicht eindeutig charakterisiert. Demnach werden zusätzliche Kontextinformationen eingefügt, um die Nutzungssituationen von einander zu unterscheiden. Die Vorliebe des Nutzers bezüglich des Tönungsgrads der Scheiben ist nicht nur von dem Zustand (GEPARKT, FÄHRT, STEHT) des Fahrzeugs abhängig, sondern auch davon ab, ob es sich jemand im Fahrzeug befindet oder nicht (KFZ leer, KFZ nicht leer).

Beispiel 5.11 *CRC-13 (gefiltert): Scheiben bis 50% Lichtdurchlässigkeit tönen*

Funktionale Anforderung – Req. RQ13
Das System soll die Scheiben beim geparkten Fahrzeug bis 50% Lichtdurchlässigkeit tönen.

Nutzungssituation – Sit. US12
a) ***UM09:*** *Fahrer-Name = Herr Meyer;*
Bevorzugter Tönungsgrad = 100%;
b) ***TM09:*** *Fahrzeug einparken;*
Aus dem Fahrzeug aussteigen;
c) ***EM10:*** *KFZ_Zustand = GEPARKT;*

Illustrierendes Szenario – ISce. ISC13
Herr Meyer parkt sein Fahrzeug auf einem Stellplatz ein und verlässt das Fahrzeug. Daraufhin tönt das Systems die Scheiben des Fahrzeugs automatisch bis zu einer Lichtdurchlässigkeit von 50%.

In den Beispielen 5.10 und 5.11 von Contextual Requirements Chunks werden die Kontextinformation UM09 einmal durch UM09-1 und einmal durch UM09-2 ersetzen. Ferner wird die Kontextinformation EM10 jeweils durch die Kontextinformationen TM10-1 und TM10-2 ersetzt. Die illustrierenden Szenarien ISC12 und ISC13 werden entsprechend angepasst. Die Beispiele 5.12 und 5.13 zeigen die Ergebnisse der Konsolidierung der CRCs aus den Beispielen 5.10 und 5.11.

Das Hinzufügen zusätzlicher Kontextinformationen zur Differenzierung der Nutzungssituationen ist lediglich ein erster Schritt in der Beseitigung von Inkonsistenzen, welche auf unzureichende Kontextinformationen zurück zu führen sind. Es muss zusätzlich sichergestellt werden, dass die hinzugefügten Kontextinformationen ebenfalls von dem Kontextmodell abgedeckt werden. Das Kontextmodell ist gegebenenfalls zu erweitern.

Die Sicherstellung der Konsistenz der Nutzungssituationen (d.h. der charakteristischen Kontextinformationen) zu dem Kontextmodell erfolgt wie im CCMS-Schritt der Darstellung und Validierung des Kontextmodells bei der Erstellung des Kontextmodells im siebten Schritt der RE-CAWAR Methodik. Es wird geprüft, ob das Kontextmodell angepasst werden soll (Operationen *„Fakttypen hinzufügen“ und „Fakttypen ent-*

fernen“ der Kontextmodellierung). Praktisch wird geprüft, ob die charakteristischen Kontextinformationen der angepassten Nutzungssituationen auf das Kontextmodell abgebildet werden können. Dafür werden zunächst die Typen der in den angepassten Nutzungssituationen enthaltenen Fakten identifiziert. Anschließend wird geprüft, ob diese Fakttypen bereit in dem Modell enthalten sind. Im positiven Fall wird keine Erweiterung bzw. Änderung des Kontextmodells vorgenommen. Die Ergebnisse der Konsolidierung werden dann freigegeben. Im negativen Fall wird das Kontextmodell entsprechend angepasst. Neue Fakttypen werden bei Bedarf eingefügt. Bestehende Fakttypen können im Zuge der Konsolidierung gegebenenfalls entfernt werden. Die Erweiterung des Kontextmodells ist ebenfalls nach dem RE-CAWAR CCMS Schema zur Erstellung eines Kontextmodells zu erfolgen (Identifikation & Transformation, Darstellung & Validierung, Optimierung).

Die Beseitigung der Inkonsistenzen aus den Contextual Requirements Chunks CRC-12 und CRC-13 aus den Beispielen 5.10 und 5.11 erfordert die Anpassung des ursprünglichen Kontextmodells aus Abbildung 5.11. Abbildung 5.15 stellt das angepasste Kontextmodell dar. Dabei sind die von dieser Anpassung betroffenen Teile des Modells aufgrund der Übersichtlichkeit farblich markiert.

Beispiel 5.12 *CRC-12 (konsolidiert): Scheiben bis 0% Lichtdurchlässigkeit tönen*

Funktionale Anforderung – Req. RQ12
Das System soll die Scheiben bis zu einer Lichtdurchlässigkeit von 0% tönen.

Nutzungssituation – Sit. US12

a) ***UM09-1:*** *Fahrer-Name = Herr Meyer;*
   Bevorzugter Tönungsgrad beim geparkten leeren Fahrzeug = 100%;
b) ***TM09:*** *Fahrzeug einparken;*
   Aus dem Fahrzeug aussteigen;
c) ***EM10-1:*** *KFZ_Zustand = GEPARKT;*
   # Personen im Fahrzeug = 0;

Illustrierendes Szenario – ISce. ISC12
Herr Meyer parkt sein Fahrzeug auf einem Stellplatz ein und verlässt das Fahrzeug. Es befindet sich keine Person in dem Fahrzeug. Aufgrund seiner Vorliebe die Scheiben seines Fahrzeugs im geparktem Zustand bis zu einer Lichtdurchlässigkeit von 0% zu tönen falls sich keine Person in dem Fahrzeug befindet, tönt das Systems die Scheiben des Fahrzeugs automatisch bis zu einer Lichtdurchlässigkeit von 0%.

Beispiel 5.13 *CRC-13 (konsolidiert): Scheiben bis 50% Lichtdurchlässigkeit tönen*

Funktionale Anforderung – Req. RQ13
Das System soll die Scheiben beim geparkten Fahrzeug bis 50% Lichtdurchlässigkeit tönen.

Nutzungssituation – Sit. US13

a) ***UM09-2:*** *Fahrer-Name = Herr Meyer;*
Bevorzugter Tönungsgrad beim geparkten nicht leeren Fahrzeug = 50%;
b) ***TM09:*** *Fahrzeug einparken;*
Aus dem Fahrzeug aussteigen;
c) ***EM10-2:*** *KFZ_Zustand = GEPARKT;*
 # Personen im Fahrzeug > 0;

Illustrierendes Szenario – ISce. ISC13
Herr Meyer parkt sein Fahrzeug auf einem Stellplatz ein und verlässt das Fahrzeug. Es befindet sich allerdings eine Person in dem Fahrzeug. Aufgrund seiner Vorliebe die Scheiben seines Fahrzeugs im geparktem Zustand bis zu einer Lichtdurchlässigkeit von 50% zu tönen falls sich mindestens eine Person in dem Fahrzeug befindet, tönt das Systems die Scheiben des Fahrzeugs automatisch bis zu einer Lichtdurchlässigkeit von 50%.

Die Contextual Requirements Chunks sollten nach der Konsolidierung der Ergebnisse keine „horizontale“ Inkonsistenz mehr enthalten[9]. Dies bedeutet dass, wenn ein CRC $c1$ eine funktionale Anforderung $r1$ in der Situation $s1$ erfordert und ein weiteres CRC $c2$ eine funktionale Anforderung $r2$ in der selben Situation $s1$ fordert, dann sollen sich $r1$ und $r2$ nicht gegenseitig ausschließen, d.h. dass, das System $r1$ und $r2$ gleichzeitig erfüllen können soll. Abgesehen von den horizontalen Inkonsistenzen ist möglich, dass es „vertikale“ Inkonsistenzen in der CRC-Tabelle existiert. Eine systematische Identifikation dieser Art von Inkonsistenzen erfordert eine Formalisierung der Anforderungen sowie eine Aufstellung eines Situationskalküls, mit Hilfe dessen die Nutzungssituationen miteinander in Beziehung gesetzt werden und Aussagen über den Wechsel zwischen den Situationen gemacht werden können. Solche Aussagen werden bei der Spezifikation des Adaptionsverhaltens gemacht und geprüft. Die Beseitigung von vertikalen Inkonsistenzen (inkl. des Kalküls über die Nutzungssituationen der Anwendung) bleibt ein offener Punkt, der aufgrund des begrenzten Umfangs dieser Arbeit bewusst ausgespart wurde.

Die RE-CAWAR Methodik erfordert eine intensive Kommunikation zwischen den Stakeholder. Eintreffende Feedback sind konsequent abzuarbeiten. Auch Feedback aus den späteren Aktivitäten im Entwicklungsprozess (beispielsweise aus der Systemspezifikation, dem Entwurf und der Implementierung) sind bis zum RE-CAWAR Schritt der Konsolidierung der Ergebnisse zu berücksichtigen und einzuarbeiten. Falls kein Feedback mehr einzuarbeiten ist, sind die Aktivitäten der RE-CAWAR Methodik an

[9] Horizontal und vertikal beziehen sich auf die CRC-Tabelle. Horizontale Inkonsistenzen betreffen die Zeilen, und vertikale Inkonsistenzen betreffen die Spalten der Tabelle.

dieser Stelle abgeschlossen. Die freizugebenden Endergebnisse der RE-CAWAR Methodik sind das umfassende Kontextmodell der Anwendung und die Contextual Requirements Chunks.

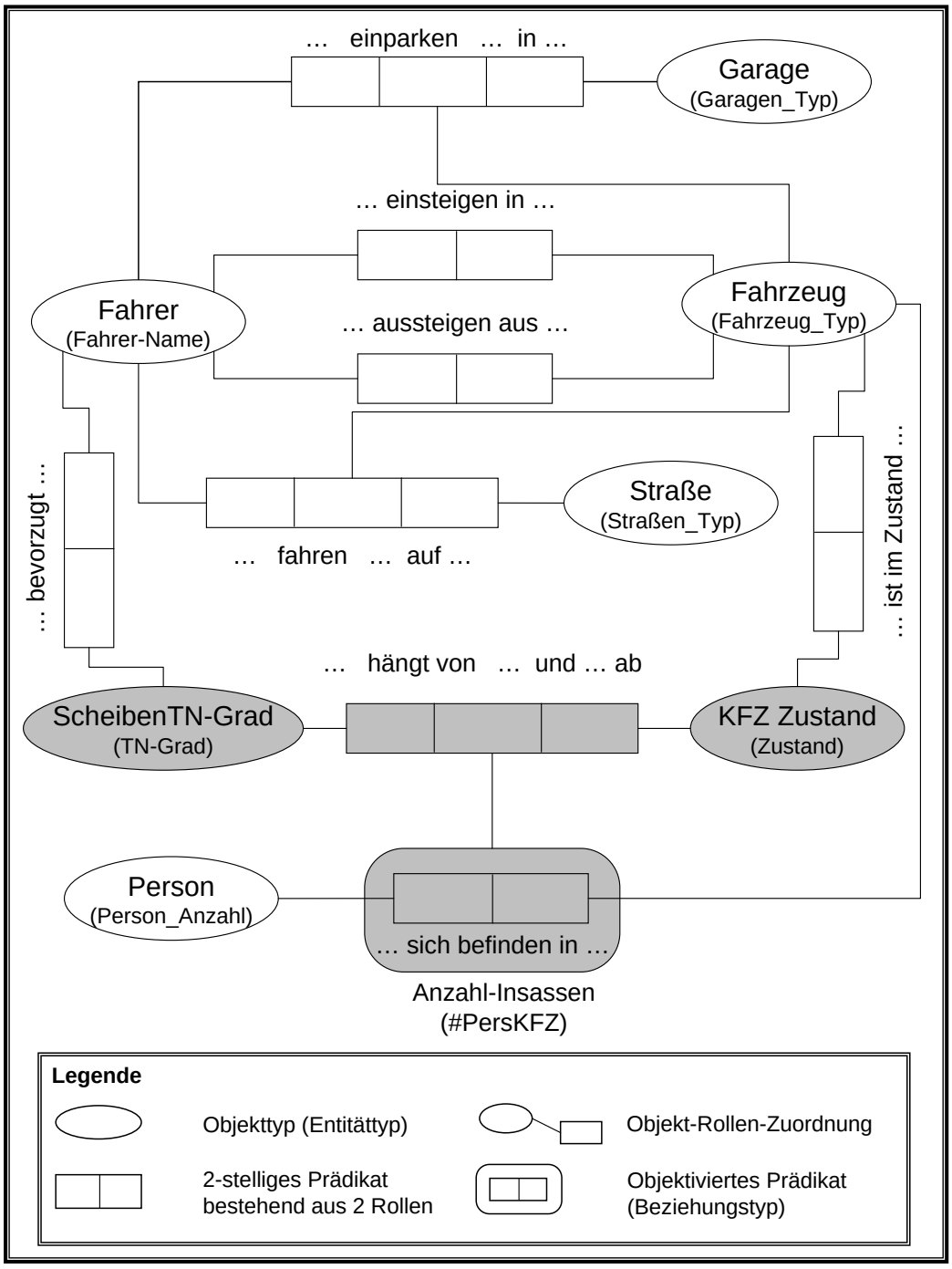

Abbildung 5.15: Erweiterung des Kontextmodells im Zuge der Konsolidierung

Schritt 9: Integration von Feedback

Mit der Durchführung von Feedback-Iterationen mit Beteiligung relevanter Stakeholder (insbesondere Nutzer, Requirements Ingenieur, Architekt sowie Produkt-Managements) werden die Artefakte der Methodik überprüft. Dabei ist die Rolle der Experten der Anwendungsdomäne nicht zu vernachlässigen. Anfallende Feedback werden abgestimmt und eingearbeitet. Bei der Integration von Feedback unterscheiden wir zwischen Scoping-bezogenem und Modellierungsbezogenem Feedback.

1. Scoping-bezogenes Feedback

Diese Art von Feedback betrifft die Artefakte der Systemvision und der Stakeholder-Ziele. Zwar werden die Aktivitäten der Scoping-Phase von einer Integration von Feedback und Abstimmung mit relevanten Stakeholder begleitet, allerdings ist nicht ausgeschlossen, dass Feedback bzgl. der Erreichung der Ziele nach der Konsolidierung auftauchen. Die Einarbeitung derartigen Feedback erfordert eine erneute Untersuchung der im Schritt 4 modellierten Anwendungsszenarien. Eventuell sind zusätzliche Anwendungsszenarien zu modellieren.

2. Modellierungsbezogenes Feedback

Diese Feedback betrifft die funktionalen Anforderung, die Nutzungssituationen und das Kontextmodell der Anwendung (Konflikte in den Contextual Requirements Chunks oder Unzulänglichkeiten in dem Kontextmodell). Unregelmäßigkeiten hierzu können beispielsweise bei dem Übergang zur Systemspezifikation oder im Systementwurf festgestellt werden. Sie werden als Feedback an die Anforderungsspezifikation geliefert und müssen verarbeitet werden. Demnach sind die gelieferten Ergebnisse, das Kontextmodell und die Contextual Requirements Chunks, erneut zu analysieren. Mögliche Änderungen sind dementsprechend vorzunehmen. Die Änderungen können eine Erweiterung oder einen Wegfall eines Contextual Requirements Chunks bedeuten. Es ist aber auch möglich, dass dem Kontextmodell Kontextattribute hinzugefügt oder entfernt werden.

Um sicher zu gehen, dass durch die Integration von Feedback keine Unstimmigkeiten in den Ergebnissen entstehen, wird anschließend über die Methodik iteriert. Der Einstiegspunkt jeder Iteration hängt von der Art des Feedback ab. Bei einem Scoping-bezogenen Feedback ist Schritt 2 der ideale Einstiegspunkt der Iteration. Bei einem Modellierungsbezogen Feedback beginnt die Iteration ab Schritt 4, 5 oder 6 der RE-CAWAR Methodik je nachdem, ob es um die Nutzungsszenarien, die funktionalen Anforderungen oder das Kontextmodell bzw. die Nutzungssituationen geht.

5.4 Zusammenfassung

Wie zu Beginn dieses Kapitels aufgeführt ist, sind wir im RE kontextsensitiver Anwendungen daran interessiert, die zwei grundlegenden Fragestellungen zu adressieren:

1. Was sind die charakteristischen Kontextinformationen einer Nutzungssituation?
2. Welche Systemreaktion/Funktion wird in welcher Nutzungssituation erwartet?

Um die erste Fragestellung zu adressieren, sind Methoden zur Identifikation der relevanten Kontextinformationen in den gewünschten Nutzungssituationen der Anwendung erforderlich. Um die zweite Fragestellung zu adressieren, werden Methoden benötigt, mit Hilfe derer die Anforderungen an die Anwendung unter Berücksichtigung der Nutzer-Ziele und Wünsche ermittelt, und in Beziehung mit den Nutzungssituationen analysiert und spezifiziert werden können. In diesem Kapitel wurden diese Methoden integriert in einer Methodik – dem Requirements Engineering kontextsensitiver Anwendungen (RE-CAWAR) – vorgestellt.

Die RE-CAWAR Methodik ist ein modellbasierter RE-Ansatz, im Rahmen derer die zu erarbeiteten Artefakte im RE kontextsensitiver Anwendungen aufgeführt sind. Kernstück der Methodik ist eine integrierte Kontext- und Szenario-Analyse, die ausgehend von Nutzer-Zielen die funktionalen Anforderungen und den Nutzungskontext der Anwendung schrittweise ermittelt, analysiert und modelliert. Wichtig ist die Tatsache, dass die Anforderungen mit den Nutzungssituationen, in denen erstere auftreten, gekoppelt sind. Jede Nutzungssituation wird mit Angabe konkreter Kontextinformationen charakterisiert. Um die Kopplung zwischen den Anforderungen und den Nutzungssituationen zu verdeutlichen, werden illustrierende Szenarien angegeben, in denen sowohl die funktionalen Anforderungen (genauer die Funktionalitäten, mit denen sie umgesetzt werden) als auch die charakterisierenden Kontextinformationen der jeweiligen Nutzungssituationen vorkommen. Das resultierende Tupel aus funktionaler Anforderung, Nutzungssituation und illustrierendem Szenario wird in der RE-CAWAR Methodik *Contextual Requirements Chunk* genannt. Jedes Chunk hat dabei folgende Gestalt:

WENN $Nutzungssituation$ **DANN** $Funktionale\ Anforderung$
MIT $Illustration\ anhand\ eines\ Szenarios.$

Dadurch, dass die Chunks für jede Nutzungssituation die geforderten funktionalen Anforderung enthalten, beschreiben sie in der Sprache der Nutzer (d.h. auf Ebene der Problemstellung) die Logik der Adaption in Form von *WENN-DANN-Regeln*. Aus der Menge der charakteristischen Kontextinformationen aller Nutzungssituationen der Anwendung wird das Kontextmodell der Anwendung abgeleitet. Genauer gesagt, werden die Nutzungssituationen als Menge von Fakten beschrieben, wobei die Typen dieser Fakten (d.h. die Fakttypen) das Kontextmodell der Anwendung bilden. Bei der Modellierung des Kontextes der Nutzung der Anwendung und der Zusammenstellung der Contextual Requirements Chunks wird gefordert, dass jede aufgeführte Nutzungssituation von dem Kontextmodell abgedeckt wird. Dies bedeutet, dass die charakteristischen Kontextinformationen einer Nutzungssituation auf das Kontextmodell abgebildet werden sollen. Diese Forderung wird dadurch erfüllt, dass das Kontextmodell aus Fakttypen besteht, welche die Typen der in den Kontextinformationen enthaltenen Fakten sind. Für die Modellierung der Fakten bzw. der Fakttypen wurde ein konzeptuelles Schema *(Conceptual Context Modelling Schema – CCMS)* vorgeschlagen. Die vorgestellten Konzepte wurden anhand von Beispielen aus den begleitenden Fallstudien des kontextsensitiven Terminplaners und des kontextsensitiven Scheibentönungssystems illustriert.

Kapitel 6

Resümee

If ease of use was the only valid criterion,
people would stick to tricycles and never try bicycles.

Douglas Engelbart,
Der Erfinder der Maus

Inhaltsangabe

6.1 Einleitung

Mit dem Begriff *Context-Awareness* führte Bill Schilit ein Konzept ein, bei dem der Kontext der Nutzung einer Anwendung eine essenzielle Rolle bei der Entscheidung spielt, wie die Anwendung auf eine Nutzereingabe hin zu reagieren hat. Bei diesem Konzept reicht es nicht mehr aus, den Zweck einer Anwendung auf ihren funktionalen Anforderungen zu reduzieren. Es muss zusätzlich analysiert werden, in welchen Situationen die Anwendung genutzt wird. Die Spezifikation der Nutzungssituationen der Anwendung erfolgt, genau wie die Spezifikation der Anforderungen, im Requirements Engineering.

In der vorliegenden Arbeit wurde eine RE-Methodik (RE-CAWAR – Requirements Engineering for Context-Aware Applications) konzipiert, anhand derer die Anforderun-

gen an eine kontextsensitive Anwendung gekoppelt mit den Nutzungssituationen modelliert werden. Darin werden die Nutzungssituationen der Anwendung durch Kontextinformationen charakterisiert und in einem Kontextmodell abgebildet. Um die Modellierung des Kontextes im Rahmen der konzipierten Methodik zu erleichtern, haben wir ebenfalls ein Framework definiert, mit dessen Hilfe der Kontext der Anwendung systematisch modelliert werden kann. Die Konzepte, die in der vorliegenden Arbeit aufgeführt sind, wurden in ihren frühen Stadien zum Teil in [Leuxner, 2006] anhand des kontextsensitiven Terminplaners und [Uwineza Gatabazi, 2008] anhand des kontextsensitiven Scheibentönungssystems erprobt. Die dabei gewonnenen Erkenntnisse sind in diese Arbeit mit eingeflossen. Eine aktuelle Anwendung – und zugleich Erprobung für die Zwecke der Weiterentwicklung – der RE-CAWAR Methodik erfolgt im Rahmen des von der Deutschen Forschungsgemeinschaft (DFG) finanzierten interdisziplinären CawarFlow-Projektes (siehe [C2MCA, 2008] - CawarFlow).

In diesem Kapitel werden die wesentlichen Ergebnisse der Arbeit zusammengefasst. Anschließend werden die offenen Punkte und Fragestellungen, sowie mögliche Weiterentwicklungen und Ansätze angesprochen.

6.2 Zusammenfassung

In der vorliegenden Dissertation wurde eine Methodik für das Requirements Engineering kontextsensitiver Anwendungen (RE-CAWAR – *Requirements Engineering for Context-Aware Applications*) entwickelt. Der Ablauf der Methodik, einschließlich der Artefakte, d.h. der Ergebnisse, die bei der Durchführung der methodischen Schritte erzielt werden, wird im Folgenden zusammengefasst.

Die RE-CAWAR Methodik beginnt mit einer *Scoping-Aktivität*, welche in drei iterativen Schritten durchgeführt wird: die Festlegung der Anwendungsdomäne, die Identifikation der Stakeholder und ihrer Ziele sowie die Definition der Vision des zu entwickelnden Systems. Die daraus resultierenden Artefakte sind zum Einen die *Liste der Stakeholder-Ziele* und zum Anderen die *Systemvision*.

Nach der Scoping-Aktivität erfolgt eine integrierte Kontext- und Szenario-Analyse im Rahmen der *Modellierung*. Dabei werden die Schritte der Modellierung der Anwendungsszenarien, der funktionalen Anforderungen an das System und des Kontextes der Nutzung der Anwendung durchgeführt. Die Artefakte, die bei der Modellierungsaktivität erstellt werden, sind die *Anwendungsszenarien*, die *funktionalen Anforderungen* und das *Kontextmodell* der Anwendung. Die Anwendungsszenarien enthalten zunächst nicht notwendigerweise die für die Adaption der Anwendung relevanten Kontextinformationen. Sie werden explizit bei der Kontextmodellierung mit letzteren angereichert. Die Modellierung des Kontextes im Rahmen der RE-CAWAR Methodik erfolgt auf Basis eines grundlegenden Konzeptes. Die Grundlagen der Kontextmodellierung werden in einem eigenständigen Framework behandelt.

Nach der Modellierungsaktivität erfolgt anschließend die Konsolidierung der erzielten Ergebnisse (*Konsolidierungsaktivitäten*). Die Artefakte der Konsolidierungsaktivität – die Endergebnisse der RE-CAWAR Methodik – bestehen einerseits aus dem *Kontextmodell* der Anwendung und andererseits aus den funktionalen Anforderungen an das System (*User Requirements Specification*). Die Anforderungen sind jeweils mit einer

Menge von Kontextinformationen verknüpft, welche die einzelnen Nutzungssituationen der Anwendung charakterisieren. Zugleich ist für jedes Paar, bestehend aus einer funktionalen Anforderung und den charakteristischen Informationen einer Nutzungssituation ein illustrierendes Szenario angegeben. Das Szenario beschreibt in der angegebenen Situation die Nutzung der Anwendungsfunktion, welche der betroffenen funktionalen Anforderung genügt. Die so beschriebenen Tupel bestehend aus funktionalen Anforderungen, charakteristischen Informationen der Nutzungssituationen sowie illustrierenden Szenarien werden in der RE-CAWAR Methodik *Contextual Requirements Chunks* (CRC) genannt.

Die Artefakte des Requirements Engineering kontextsensitiver Anwendungen sind in der Abbildung 6.1 zusammengefasst.

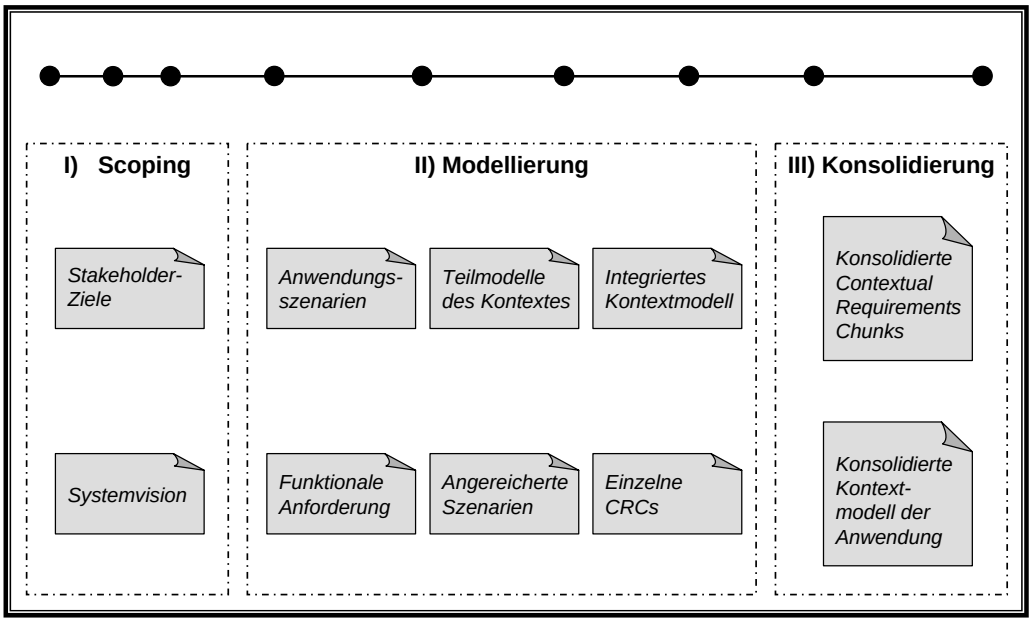

Abbildung 6.1: Artefakte der RE-CAWAR Methodik

Als wichtigste Ergebnisse der Arbeit sind hier zu nennen:

6.2.1 Herausforderungen kontextsensitiver Anwendungen

Um die RE-CAWAR Methodik, einschließlich des Frameworks zur Kontextmodellierung, zielgerichtet zu konzipieren, war es notwendig, die Herausforderungen kontextsensitiver Anwendungen sorgfältig zu analysieren. Die Ergebnisse dieser Analyse fassen wir im Folgenden zusammen.

Das beobachtbare Verhalten einer kontextsensitiven Anwendung hängt nicht nur von den Eingaben des Nutzers ab, sondern auch von ihrem Nutzungskontext. Dies kann dazu führen, dass die Anwendung auf die gleiche Nutzereingabe unterschiedlich reagiert. Demnach weisen kontextsensitive Anwendungen aus Sicht des betrachteten Nutzers ein nicht-deterministisches Verhalten auf.

In einer übergreifenden Taxonomie des Kontextes ist der Nutzer der Anwendung explizit zu betrachten. Dabei ist zu bedenken, dass Nutzer grundsätzlich unterschiedlich

sind. Wir unterscheiden zwischen *gewöhnlichen Nutzern*, *vorsichtigen Nutzern* und *erfahrenen Nutzern*.

Der Nutzungskontext ist der am wenigsten verstandene und am wenigsten untersuchte Aspekt eines Systems [Cheng und Atlee, 2007]. In vielen Fällen ist ein Versagen des Systems, dessen Ursache vermeintlich in fehlerhaften oder unvollständigen Anforderungen liegt, tatsächlich auf eine unzureichende Analyse des Kontextes zurückzuführen (siehe [Loucopoulos und Karakostas, 1995]). Ein System kann sich kaum erwartungsgemäß verhalten, wenn der Kontext, in dem es genutzt wird, im Zuge des Requirements Engineering fehlerhaft bzw. unzureichend berücksichtigt wurde.

Eine Herausforderung des RE kontextsensitiver Anwendungen ist das Vermeiden unerwünschten Systemverhaltens, welches aufgrund unzureichender Kontextmodelle, unterschiedlicher Nutzererwartungen oder falscher Regeln für die Adaptionsentscheidung auftreten.

In der Literatur zum Thema Requirements Engineering werden kontextsensitive Anwendungen bislang nicht explizit und ausreichend behandelt. Geeignete Entwicklungsmethoden müssen erarbeitet werden, welche sowohl den Nutzer als auch die Nutzungsgegebenheiten in den Mittelpunkt stellen. Die Untersuchung der Nutzerwünsche, seiner Bedürfnisse und Ziele sowie des beabsichtigten Nutzungskontexts der Anwendung ist wesentlicher Bestandteil des Requirements Engineering kontextsensitiver Anwendungen.

6.2.2 Framework zur Kontextmodellierung

Zentral in einer kontextsensitiven Anwendung ist das Kontextmodell. Es wird dazu verwendet, Nutzungssituationen der Anwendung festzustellen, und darauf hin zu entscheiden, welche Systemreaktion und dementsprechend welche Adaption in einer identifizierten Nutzungssituation angebracht ist. Die Komplexität der Erstellung eines Kontextmodells wächst mit der Anzahl der Faktoren, die bei den Adaptionsentscheidungen berücksichtigt werden. Im Allgemeinen gilt: je mehr Kontextinformationen betrachtet werden, desto genauer werden die Adaptionsentscheidungen, und dementsprechend situationsgerechter können die Anwendungen reagieren.

In einem Anwendungsfall wie „sobald eine Nachricht eingetroffen ist, soll der Nutzer benachrichtigt werden“, spielen eine Reihe von Kontextinformationen eine wichtige Rolle. Beispiele für die Kontextinformationen sind die Nutzerpräferenzen bzgl. der Benachrichtigungsart, seine Beziehung zu anderen Personen in seiner unmittelbaren Nähe, sowie die für die Zwecke der Benachrichtigung zur Verfügung stehenden Geräte. Mit etwas mehr Sorgfalt können weitere relevante Kontextinformationen für die Benachrichtigungsanwendung festgestellt werden.

In der vorliegenden Arbeit haben wir ein Framework konzipiert, welches die sorgfältige Modellierung des Nutzungskontextes einer Anwendung unterstützt. Die möglichen Informationen, die zum Kontext einer Anwendung gehören können, wurden klassifiziert, damit ihre Modellierung zielgerichtet und systematisch erfolgen kann. Kern des Frameworks ist eine grundlegende Taxonomie des Kontextes, bei dem seine zentralen Aspekte *Nutzer, Aktivität und Einsatzumgebung sowie ihre Wechselwirkungen unter einander und ihre Veränderungen über die Zeit* hervorgehoben werden.

Um die Aspekte des Kontextes adäquat abzudecken, wurden sechs Modelle eingeführt, die allesamt Teil eines umfassenden Kontextmodells sein können:

1. ein Nutzermodell,
2. ein Aufgabenmodell,
3. ein Umgebungsmodell,
4. ein Plattformmodell,
5. ein Interaktionsmodell und
6. ein Präsentationsmodell.

Dabei sind die ersten drei Modelle zentraler Natur und dürfen nicht allesamt gleichzeitig fehlen. Die letzten drei Modelle hingegen sind als unterstützende Modelle zu betrachten.

Ein Ansatz zur Integration der einzelnen Modelle in ein umfassendes Kontextmodell wurde ebenfalls entwickelt. Dieser Ansatz basiert auf die Tatsache, dass Nutzungssituationen mit Kontextinformationen charakterisiert werden, welche als Fakten ausgedrückt werden. Die Typen dieser Fakten (Fakttypen) bilden ein Faktenmodell, welches als Kontextmodell der Anwendung fungiert. Demzufolge stellt jede Instanz des Kontextmodells eine Nutzungssituation der Anwendung dars.

6.2.3 Modellbasierte RE-Methodik (RE-CAWAR)

Mit der RE-CAWAR Methodik bieten wir einen modellbasierten Ansatz an, im Rahmen dessen die zu erarbeiteten Artefakte im RE kontextsensitiver Anwendungen aufgeführt sind. Grundlegend an der RE-CAWAR Methodik ist die integrierte Ziel-, Szenario- und Kontext-Analyse. Ausgehend von Nutzer-Zielen werden die funktionalen Anforderungen und der Nutzungskontext der Anwendung schrittweise ermittelt, analysiert und modelliert.

Eine Besonderheit der RE-CAWAR Methodik ist die Tatsache, dass die Anforderungen mit Kontextinformationen gekoppelt sind, welche die Nutzungssituationen der umsetzenden Systemfunktionen charakterisiert. Mit dieser Kopplung wird vermieden, dass das System Reaktionen in Situationen zeigt, in denen sie nicht zu erwarten sind. Die Kopplung zwischen den Anforderungen und den Nutzungssituationen wird dadurch verdeutlicht, dass illustrierende Szenarien angegeben werden. In den Szenarien kommen sowohl die Anforderungen (genauer die Funktionalitäten, mit denen sie umgesetzt werden) als auch die charakterisierenden Kontextinformationen der jeweiligen Nutzungssituationen vor. Das resultierende Tupel aus funktionaler Anforderung, Nutzungssituation und illustrierendem Szenario wird in der RE-CAWAR Methodik *Contextual Requirements Chunk* genannt.

Die Menge der charakteristischen Kontextinformationen aller Nutzungssituationen der Anwendung ergibt das Kontextmodell der Anwendung. Die Nutzunssituationen werden als Menge von Fakten beschrieben, wobei die Typen dieser Fakten (d.h. die Fakttypen) das Kontextmodell der Anwendung bilden. Die Modellierung des Kontextes erfolgt mit Hilfe des in dieser Arbeit konzipierten Frameworks.

Konsolidierte Chunks dienen insbesondere als Grundlage für die Spezifikation der

Systemfunktionen bei dem Übergang zwischen Anforderungsanalyse und Systementwurf. Auch die Bildung von Konfigurationen in kontextsensitiven Anwendungen für die Zwecke der Realisierung von Adaptionen, wird von den Chunks geleitet. Dadurch, dass die Chunks für jede Nutzungssituation die geforderten funktionalen Anforderung enthalten, beschreiben sie auf Ebene der Problemstellung die Logik der Adaption in Form von *WENN-DANN-Regeln*. Aufgrund der expliziten Behandlung der Nutzer-Ziele in der Methodik können Abweichungen zwischen Nutzererwartung und Systemverhalten seltener vorkommen.

Die vorgestellten Konzepte wurden anhand von Beispielen aus den begleitenden Fallstudien des kontextsensitiven Terminplaners und des kontextsensitiven Scheibentönungssystems illustriert und validiert.

6.3 Ausblick

Die Entwicklung kontextsensitiver Anwendungen stellt eine Reihe interessanter Fragestellungen. In dieser Arbeit haben wir uns auf das Requirements Engineering konzentriert und einen Ansatz entwickelt, mit Hilfe dessen die Anforderungen an das System sowie der Nutzungskontext der Anwendung ermittelt, analysiert und modelliert werden.

Eine übergreifende und durchgängige Methodik zur Entwicklung kontextsensitiver Anwendung erfordert die Einbindung weiterer Abschnitte aus dem Gebiet des Software Engineering aufgegriffen werden. Diese weiteren Abschnitte müssen gegebenenfalls für die Bedürfnisse kontextsensitiver Anwendungen so angepasst werden, wie es hier für den Abschnitt des Requirements Engineering geschehen ist.

Aus den Ergebnissen der Arbeit ergeben sich eine Vielzahl unmittelbarer Anknüpfungspunkte für weitere Themen. Diese Themen bauen auf den in dieser Arbeit erzielten Ergebnissen auf bzw. ergänzen sie. Wir gehen im Folgenden kurz darauf ein.

6.3.1 Entscheidung für eine kontextsensitive Lösung

Die Entscheidung, ob eine kontextsensitive Anwendung anstelle einer nichtkontextsensitiven als Lösung eines Problems erwogen werden sollte, ist bislang nicht ausreichend untersucht. Es gibt eine Reihe von Vermutungen, die darauf hindeuten, dass diese Fragestellung hochgradig interdisziplinär ist. Es gibt allerdings keine Systematik, mit der entschieden werden kann, wann Kontextsensitivität einer herkömmlichen Lösung vorzuziehen ist. Eine erste Hypothese hierzu ist in [Welsh und Sawyer, 2008] formuliert, welche allerdings noch zu prüfen ist.

6.3.2 Validierung der Contextual Requirements Chunks

Ein folgerichtiger weiterführender Schritt der in dieser Arbeit vorgestellten RECAWAR Methodik besteht in der Validierung der Contextual Requirements Chunks. Mit einer solchen Validierung wird sichergestellt, dass die richtige Anwendung entwickelt wird, d.h. die richtigen Paare von Anforderungen und Nutzungssituationen

erhoben und modelliert werden. Aufgrund der Konstruktion der Contextual Requirements Chunks wird sichergestellt, dass für die spezifizierten Nutzungssituationen, die Anwendung hinsichtlich unerwünschten Verhaltens (UB – siehe Abschnitt 3.4) robust ist. Es bleibt jedoch offen, welchen Anforderungen die Anwendung in nicht-spezifizierten Nutzungssituationen genügen soll. Es ist offensichtlich, dass alle Nutzungssituationen prinzipiel nicht abgedeckt werden können, allerdings kann eine Annäherung davon angestrebt werden. Ein Mittel hierzu ist der Einsatz von „Anti-Szenarien“, wie es beispielsweise im Bereich der Safety Analyse vorgenommen wird (siehe z.B. [Potts, 1999, Bontemps et al., 2005]). Mit dem Einsatz von Anti-Szenarien werden Gegenbeispiele der Nutzung der Anwendung ausgedrückt, und anschließend überprüft, ob die spezifizierten CRCs weiterhin ihre Gültigkeit behalten. Auf diese Art und Weise können weitere potentielle Nutzungssituationen identifiziert werden, in denen falls unbehandelt unerwünschte Systemverhalten bei der Nutzung der Anwendung auftreten würden, und entsprechende Maßnahmen ergriffen werden.

6.3.3 Formalisierung von Situationen

Für die Zwecke der Verifikation – *(d.h. wird die Anwendung richtig entwickelt?)* – der Contextual Requirements Chunks sind formale Methoden gefragt. Durch diese können beispielsweise Widersprüche zwischen Anforderungen festgestellt werden. Die Anforderungen kontextsensitiver Anwendung werden allerdings gemeinsam mit den Situationen, in denen die umsetzenden Systemfunktionen gelten, spezifiziert. Dies macht zudem eine Formalisierung der entsprechenden Situationen erforderlich.

Ein Rahmenwerk zur Formalisierung von Situationen würde es ermöglichen, zum einen eine gewisse Art von Inkonsistenzen in Contextual Requirements Chunks zu entdecken, und zum anderen Verfeinerungskonzepte, Abstraktionen und Sichtenbildung bzgl. Nutzungssituationen zu definieren. Ein solches Rahmenwerk würde also eine Art „Situationskalkül“ ermöglichen. Beispielsweise können Teilmengen- oder Vorgänger/Nachfolger-Beziehungen zwischen Situationen ausgedrückt werden. Aussagen der Art *eine Situation s_m ist Teil einer Situation s_n* oder *eine Situation s_i folgt aus einer Situation s_j* wären ohne Bedenken möglich. Solche Aussagen werden beispielsweise bei der Bildung von Systemkonfigurationen zur Realisierung der Adaption der Anwendung gemacht und können somit geprüft werden.

6.3.4 Bildung von Konfigurationen

Eine mögliche Herangehensweise für den Entwurf kontextsensitiver Anwendungen besteht darin, Systemfunktionen zu kombinieren, um daraus Konfigurationen zu bilden. Um die Frage der geeigneten Konfigurationsbildung zu adressieren, wäre eine eigene Methodik erforderlich. Diese sollte erörtern, wie Systemfunktionen auf Konfiguration abgebildet werden.

Mit der gekoppelten Aufstellung der Anforderungen und der Nutzungssituationen einer kontextsensitiven Anwendung ist ein Schritt in diese Richtung bereits in den frühen Phasen der Entwicklung kontextsensitiver Anwendungen gemacht worden. Die von Bill Schilit geförderte flexible Strukturierung kontextsensitiver Anwendungen (siehe [Schilit, 1995]) kann mit der Bildung von Konfigurationen adressiert werden.

Anhang

Anhang A

Fallstudie: ein kontextsensitives Scheibentönungssystem

Inhaltsangabe

A.1 Systemvision

Die Sichtbehinderung durch die blendende Sonne stellt eine Gefährdung im Straßenverkehr dar und führt häufig zu Unfällen. In den Wintermonaten wird der Fahrer durch die tief stehende Sonne geblendet, im Sommer ist die Sicht des Fahrers aufgrund der hohen Lichtintensität stark eingeschränkt.

Sonnenbrillen sind ein weit verbreitetes Mittel um der Sichtbehinderung durch die Sonne entgegenzuwirken und auch die Sonnenblenden innerhalb des Fahrzeugs verbessern die Lichtsituation. Dennoch sind beide Möglichkeiten lediglich suboptimal, da ihr Einsatz die Konzentration des Fahrers (zumindest kurzzeitig) beeinflusst:

1. Der Fahrer muss die Sonnenbrille während der Fahrt aufsetzen; bei einem Fahrer mit Sehschwäche damit auch ein Brillenwechsel verbunden, d.h. seine Sehkraft ist kurzzeitig nicht ausreichend
2. Die Sonnenblende muss ebenfalls während der Fahrt herunter geklappt werden.

Besonders deutlich wird dieser Aufwand an kognitiver Aufmerksamkeit bei häufig wechselnden Lichtverhältnissen, d.h. Wechsel zwischen hell und dunkel in zeitlich

kurzen Abständen. Der Fahrer kann sich dann nicht mehr ausreichend auf die Verkehrssituation und das eigentliche Fahren konzentrieren und somit steigt das Unfallrisiko.

Das System der dynamischen Scheibentönung soll den Fahrer vor einer Blendung durch die Sonne schützen, indem es die Scheiben des Fahrzeugs entsprechend der Lichtverhältnisse (Lichtintensität, Lichteinfallswinkel) tönt.

Darüber hinaus kann die Scheibentönung beim geparkten Fahrzeug als Diebstahlschutz fungieren, indem die Scheiben bis zu einer Lichtdurchlässigkeit von 0% getönt werden und somit der Einblick von außen in das Fahrzeug verhindert wird.

A.2 Stakeholder-Ziele

Stakeholder-Ziel G01

Bezeichner: G01

Kurzbezeichnung: Verringerung des Unfallrisikos im Straßenverkehr

Beschreibung: Mit dem Einsatz des Systems soll das Risiko von Unfällen im Straßenverkehr verringert werden.

Übergeordnetes Ziel: -

Stakeholder: Fahrer, Hersteller, Gesetzgeber, Mitfahrer

Anmerkung: -

Stakeholder-Ziel G02

Bezeichner: G02

Kurzbezeichnung: Schutz des Fahrers vor einer Blendung durch die Sonne

Beschreibung: Mit dem Einsatz des Systems soll der Fahrer vor einer Blendung durch die Sonne geschützt werden

Übergeordnetes Ziel: G01

Stakeholder: Fahrer, Hersteller, Gesetzgeber, Mitfahrer

Anmerkung: Mit diesem Ziel wird die Vermeidung der Blendung des Fahrers durch die Sonne angestrebt.

Stakeholder-Ziel G03

Bezeichner: G03

Kurzbezeichnung: Freie Sicht im Tunnelbereich

Beschreibung: Mit dem Einsatz des Systems soll eine freie Sicht des Fahrers während der Fahrt im Tunnelbereich gewährleistet werden.

Übergeordnetes Ziel: G01

Stakeholder: Fahrer, Hersteller, Gesetzgeber, Mitfahrer

Anmerkung: Mit diesem Ziel wird die Vermeidung von Sichtbehinderung

durch getönte Scheiben im Tunnelbereich angestrebt.

Stakeholder-Ziel G04

Bezeichner: G04

Kurzbezeichnung: Wärmereduktion des geparkten Fahrzeugs

Beschreibung: Mit dem Einsatz des Systems soll beim geparkten Fahrzeug die Temperatur im Fahrzeuginnenraum reduziert werden.

Übergeordnetes Ziel: -

Stakeholder: Fahrer, Mitfahrer

Anmerkung: Mit diesem Ziel wird die Tönung der Scheiben des geparkten Fahrzeugs angestrebt, um die Temperatur im Fahrzeuginnenraum zu reduzieren.

Stakeholder-Ziel G05

Bezeichner: G05

Kurzbezeichnung: Diebstahlschutz des geparkten Fahrzeugs

Beschreibung: Mit dem Einsatz des Systems soll beim geparkten Fahrzeug der Einblick von außen verhindert werden

Übergeordnetes Ziel: -

Stakeholder: Fahrer, Mitfahrer

Anmerkung: Mit diesem Ziel wird die Tönung der Scheiben des geparkten Fahrzeugs angestrebt, um den Fahrzeuginnenraum vor neugierigen Blicken zu schützen

Stakeholder-Ziel G06

Bezeichner: G06

Kurzbezeichnung: Schutz des Fahrers vor einer Blendung durch variierende Lichtverhältnisse

Beschreibung: Mit dem Einsatz des Systems soll der Fahrer vor einer Blendung durch die Sonne geschützt werden, welche aufgrund variierender Lichtverhältnisse und wiederholter Abfolge von Scheiben-abdunkeln und Scheiben-aufhellen.

Übergeordnetes Ziel: G01

Stakeholder: Fahrer, Hersteller, Gesetzgeber, Mitfahrer

Anmerkung: Mit diesem Ziel wird die Vermeidung einer hin und her Tönung der Scheiben angestrebt.

Stakeholder-Ziel G07

Bezeichner: G07

Kurzbezeichnung: Vollständige Abhängigkeit des Fahrers vermeiden

Beschreibung: Mit dem Einsatz des Systems soll der Fahrer vor einer vollständi-

gen Abhängigkeit von dem System geschützt werden. Dem Fahrer soll trotz allem die Möglichkeit zum Eingreifen erhalten bleiben.

Übergeordnetes Ziel: G01

Stakeholder: Fahrer, Gesetzgeber

Anmerkung: Mit diesem Ziel wird die Vermeidung einer Verlust der Kontrolle des Fahrers über das Scheibentönungsverhalten des Systems angestrebt.

A.3 Anwendungsszenarien

Anwendungsszenario USC01

Kurzbezeichnung: Scheiben des geparkten Fahrzeugs vollständig tönen

Ziel: G04 und G05 - Wärmereduktion bzw. Diebstahlschutz beim geparkten Fahrzeug

Vorbedingung:

1. Das Fahrzeug ist geparkt
2. Das Scheibentönungssystem ist aktiv

Aktionen:

1. Der Fahrer drückt auf den Knopf zur Auslösung der vollständigen Tönung der Scheiben.
2. Das System tönt die Scheiben des Fahrzeugs bis zu einer Lichtdurchlässigkeit von 0%

Nachbedingung:

1. Die Scheiben des Fahrzeugs sind getönt
2. Die Lichtdurchlässigkeit bei den Scheiben des Fahrzeugs ist 0%
3. Das Scheibentönungssystem ist aktiv

Anmerkung: Vollständige Tönung der Scheiben := Tönung bis zu einer Lichtdurchlässigkeit von 0%.

Anwendungsszenario USC02

Kurzbezeichnung: Scheiben des Fahrzeugs abhängig von den Lichtverhältnissen tönen

Ziel: G02 - Schutz des Fahrers vor einer Blendung durch die Sonne

Vorbedingung:

1. Das Fahrzeug fährt
2. Das Scheibentönungssystem ist aktiv

Aktionen:

1. Das System stellt eine Änderung der Lichtverhältnisse (Intensität und Einfallswinkel) fest

2. Das System berechnet den angemessenen Tönungsgrad N
3. Das System tönt die Scheiben des Fahrzeugs bis zu dem berechneten Tönungsgrad N

Nachbedingung:

1. Die Scheiben des Fahrzeugs sind bis zum Tönungsgrad N getönt
2. Das Scheibentönungssystem ist aktiv

Anmerkung: Der angemessene Tönungsgrad N lässt sich aus dem Lichteinfallswinkel und der äußeren Lichtintensität berechnen.

Anwendungsszenario USC03

Kurzbezeichnung: Scheiben des Fahrzeugs abhängig von den Lichtverhältnissen im Tunnelbereich tönen

Ziel: G03 - Freie Sicht im Tunnelbereich

Vorbedingung:

1. Das Fahrzeug fährt
2. Das Scheibentönungssystem ist aktiv
3. Das Fahrzeug kommt in einen Tunnelbereich

Aktionen:

1. Das System stellt den Eintritt in den Tunnelbereich fest
2. Das System stellt den aktuellen Tönungsgrad fest
3. Das System berechnet den angemessenen Tönungsgrad N
4. Das System tönt die Scheiben des Fahrzeugs bis zu dem berechneten Tönungsgrad N

Nachbedingung:

1. Die Scheiben des Fahrzeugs sind bis zum Tönungsgrad N getönt
2. Das Scheibentönungssystem ist aktiv

Anmerkung: Der angemessene Tönungsgrad N lässt sich aus dem Lichteinfallswinkel und der äußeren Lichtintensität in dem Tunnelbereich berechnen. Dabei soll die Lichtdurchlässigkeit stets zwischen 75% und 100% liegen.

Anwendungsszenario USC04

Kurzbezeichnung: Scheiben des Fahrzeugs bei wechselnden Lichtverhältnissen im Allee-Bereich angemessen tönen

Ziel: G06 - Schutz des Fahrers vor einer Blendung durch variierende Lichtverhältnisse

Vorbedingung:

1. Das Fahrzeug fährt
2. Das Scheibentönungssystem ist aktiv

3. Das Fahrzeug kommt in einen Allee-Bereich

Aktionen:

1. Das System stellt den Eintritt in den Allee-Bereich fest
2. Das System stellt den aktuellen Tönungsgrad fest
3. Das System berechnet den angemessenen Tönungsgrad N
4. Das System tönt die Scheiben des Fahrzeugs bis zu dem berechneten Tönungsgrad N

Nachbedingung:

1. Die Scheiben des Fahrzeugs sind bis zum Tönungsgrad N getönt
2. Das Scheibentönungssystem ist aktiv

Anmerkung: Der angemessene Tönungsgrad N lässt sich aus dem Lichteinfallswinkel und der äußeren Lichtintensität in dem Allee-Bereich berechnen.

Anwendungsszenario USC05

Kurzbezeichnung: Deaktivierung des Scheibentönungssystem

Ziel: G07 - Vollständige Abhängigkeit des Fahrers vermeiden

Vorbedingung:

1. Das Fahrzeug fährt
2. Das Scheibentönungssystem ist aktiv

Aktionen:

1. Der Nutzer entscheidet für eine Deaktivierung des Systems
2. Der Nutzer drückt auf dem Knopf zur Deaktivierung des Systems
3. Das System enttönt die Scheiben vollständig

Nachbedingung:

1. Die Scheiben des Fahrzeugs sind enttönt
2. Das Scheibentönungssystem ist inaktiv

Anmerkung: Alternativ zur vollständigen Enttönung ist die Beibehaltung der aktuellen Tönung. In den Diskussionen hat sich allerdings ergeben, dass diese Variante nicht als Standard gelten soll.

Anwendungsszenario USC06

Kurzbezeichnung: Aktivierung des Scheibentönungssystem

Ziel: G07 - Vollständige Abhängigkeit des Fahrers vermeiden

Vorbedingung:

1. Das Scheibentönungssystem ist inaktiv

Aktionen:

1. Der Nutzer entscheidet für eine Aktivierung des Systems

2. Der Nutzer drückt auf dem Knopf zur Aktivierung des Systems
3. Das System tönt die Scheiben bis zu einer angemessenen Grad

Nachbedingung:

1. Das Scheibentönungssystem ist aktiv

Anmerkung: Der angemessene Grad der Tönung nach einer Aktivierung des Systems wird von dem System ermittelt.

Anwendungsszenario USC07

Kurzbezeichnung: Manuelle Einstellung des Scheibentönungsgrads

Ziel: G07 - Vollständige Abhängigkeit des Fahrers vermeiden

Vorbedingung:

1. Das Scheibentönungssystem ist aktiv

Aktionen:

1. Der Nutzer stellt manuell den Grad der Tönung ein
2. Das System tönt die Scheiben bis zu dem eingestellten Grad

Nachbedingung:

1. Das Scheibentönungssystem ist aktiv

Anmerkung: Bei der manuellen Einstellung des Tönungsgrad werden lediglich tatsächlich einstellbare Werte angeboten.

A.4 Funktionalen Anforderungen

Funktionale Anforderung RQ01

Beschreibung: Das System soll einen Bedienknopf zum Anstoß der vollständigen Tönung der Scheiben des Fahrzeugs haben.

Abgeleitet aus Szenarien: USC01 (Scheiben des geparkten Fahrzeugs vollständig tönen).

Funktionale Anforderung RQ02

Beschreibung: Der Nutzer soll per Knopf-Druck die vollständige Tönung der Scheiben des Fahrzeugs anstoßen können.

Abgeleitet aus Szenarien: USC01 (Scheiben des geparkten Fahrzeugs vollständig tönen).

Funktionale Anforderung RQ03

Beschreibung: Das System soll die Lichtdurchlässigkeit der Scheiben des Fahrzeugs messen.

Abgeleitet aus Szenarien: USC01 (Scheiben des geparkten Fahrzeugs vollständig tönen).

Funktionale Anforderung RQ04

Beschreibung: Das System soll die äußere Lichtintensität messen.

Abgeleitet aus Szenarien: USC02 und USC03 (Scheiben des Fahrzeugs abhängig von den Lichtverhältnissen tönen).

Funktionale Anforderung RQ05

Beschreibung: Das System soll den Lichteinfallswinkel messen.

Abgeleitet aus Szenarien: USC02 und USC03 (Scheiben des Fahrzeugs abhängig von den Lichtverhältnissen tönen).

Funktionale Anforderung RQ06

Beschreibung: Der Nutzer soll das System deaktivieren können.

Abgeleitet aus Szenarien: USC05 (Deaktivierung des Scheibentönungssystem).

Funktionale Anforderung RQ07

Beschreibung: Der Nutzer soll das System aktivieren können.

Abgeleitet aus Szenarien: USC06 (Aktivierung des Scheibentönungssystem).

Funktionale Anforderung RQ08

Beschreibung: Der Nutzer soll den Tönungsgrad manuell einstellen können.

Abgeleitet aus Szenarien: USC07 (Manuelle Einstellung des Scheibentönungsgrads).

Funktionale Anforderung RQ09

Beschreibung: Das System soll die Scheiben bis zu einem manuell eingestellten Tönungsgrad tönen können.

Abgeleitet aus Szenarien: USC07 (Manuelle Einstellung des Scheibentönungsgrads).

Funktionale Anforderung RQ10

Beschreibung: Das System soll die Scheiben des Fahrzeugs bei wechselnden Lichtverhältnissen bis zum relativen Mittelwert μ tönen.

Abgeleitet aus Szenarien: USC02, USC03 und USC04 (Tönung der Scheiben des Fahrzeugs im Allee-Bereich).

Anmerkung: Das System soll die Scheiben des Fahrzeugs bei schnell variierenden Lichtverhältnissen (Intensität und Einfallswinkel) nicht hin und her tönen und enttönen (abdunkeln und aufhellen). Der Tönungsgrad wird auf einen kontinuierlich relativen Mittelwert μ festgelegt und alle drei Sekunden $(3s)$ aktualisiert.

Funktionale Anforderung RQ11

Beschreibung: Das System soll die Scheiben des Fahrzeugs abhängig von der äußeren Lichtintensität und dem Lichteinfallswinkel tönen.

Abgeleitet aus Szenarien: USC02, USC03 und USC04 (Scheiben des Fahrzeugs abhängig von den Lichtverhältnissen tönen).

Funktionale Anforderung RQ12

Beschreibung: Das System soll die Scheiben des Fahrzeugs bis zu einer Lichtdurchlässigkeit von 0% tönen.

Abgeleitet aus Szenarien: USC01 (Scheiben des geparkten Fahrzeugs vollständig tönen – Leeres Fahrzeug).

Anmerkung: Mit einer Scheibentönung bis 0% Lichtdurchlässigkeit wird die vollständige Tönung der Scheiben erreicht.

Funktionale Anforderung RQ13

Beschreibung: Das System soll die Scheiben des Fahrzeugs bis zu einer Lichtdurchlässigkeit von 50% tönen.

Abgeleitet aus Szenarien: USC02 (Scheiben des geparkten Fahrzeugs vollständig tönen – Nicht leeres Fahrzeug).

Funktionale Anforderung RQ14

Beschreibung: Das System soll die Scheiben des Fahrzeugs bis zu einer Lichtdurchlässigkeit von $n\% \in [75, 100]$ tönen.

Abgeleitet aus Szenarien: USC03 (Scheiben des Fahrzeugs abhängig von den Lichtverhältnissen im Tunnelbereich tönen).

Funktionale Anforderung RQ15

Beschreibung: Das System soll die Scheiben des Fahrzeugs abhängig von der äußeren Lichtintensität bis zu einem Tönungsgrad von $n\%$ tönen. $n \in \{0, 25, 50, 75\}$

Abgeleitet aus Szenarien:
USC03 (Tönung abhängig von der äußeren LI – sonnig)
USC04 (Tönung abhängig von der äußeren LI – schattig)
USC05 (Tönung abhängig von der äußeren LI – halbdunkel)
USC06 (Tönung abhängig von der äußeren LI – dunkel)

Anmerkung: LI steht für Lichtintensität
sonnig, d.h. $LI \geq 30000\ Lux \rightarrow n = 75$
schattig, d.h. $LI \geq 15000\ Lux \rightarrow n = 50$
halbdunkel, d.h. $LI \geq 5000\ Lux \rightarrow n = 25$
dunkel, d.h. $LI \leq 3500\ Lux \rightarrow n = 0$

A.5 Teilmodelle des Nutzungskontextes

Abbildungen A.1, A.2 und A.3 stellen jeweils das Benutzermodell, das Umgebungsmodell und das Aufgabenmodell des Scheibentönungssystems dar. Die Erstellung dieser Modelle erfolgt iterativ während der Anreicherung der Szenarien mit Kontextinformationen. Es ist wichtig zu merken, dass diese Modelle dazu dienen das eigentliche Kon-

textmodell der Anwendung zu erstellen, und keinerlei als Endergebnisse zu betrachten sind. Die Teilmodelle des Kontextes bieten eine Möglichkeit die Kontextinformationen strukturiert und zielgerichtet zu erfassen.

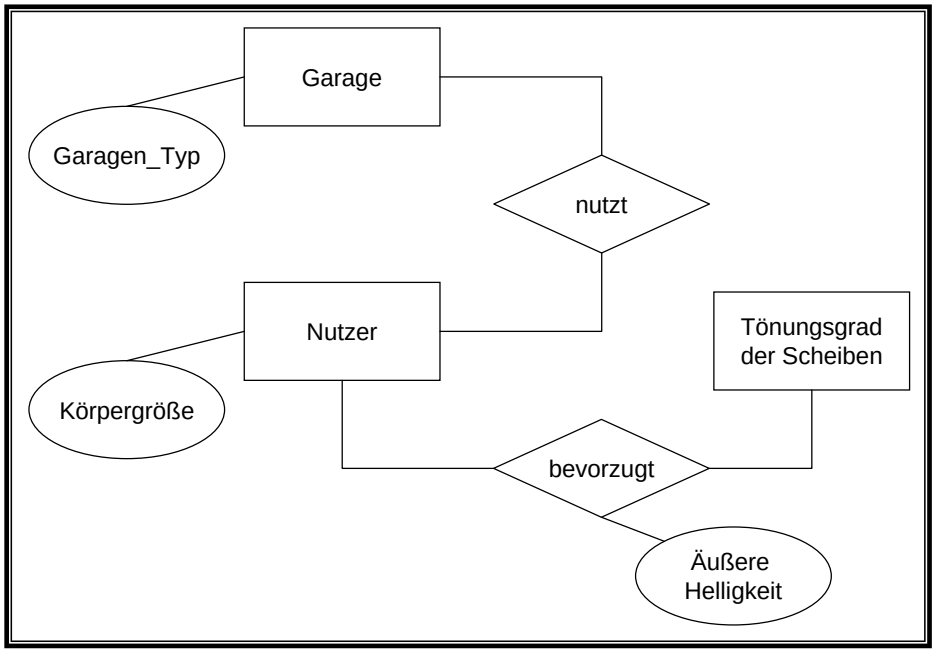

Abbildung A.1: Benutzermodell des Scheibentönungssystems

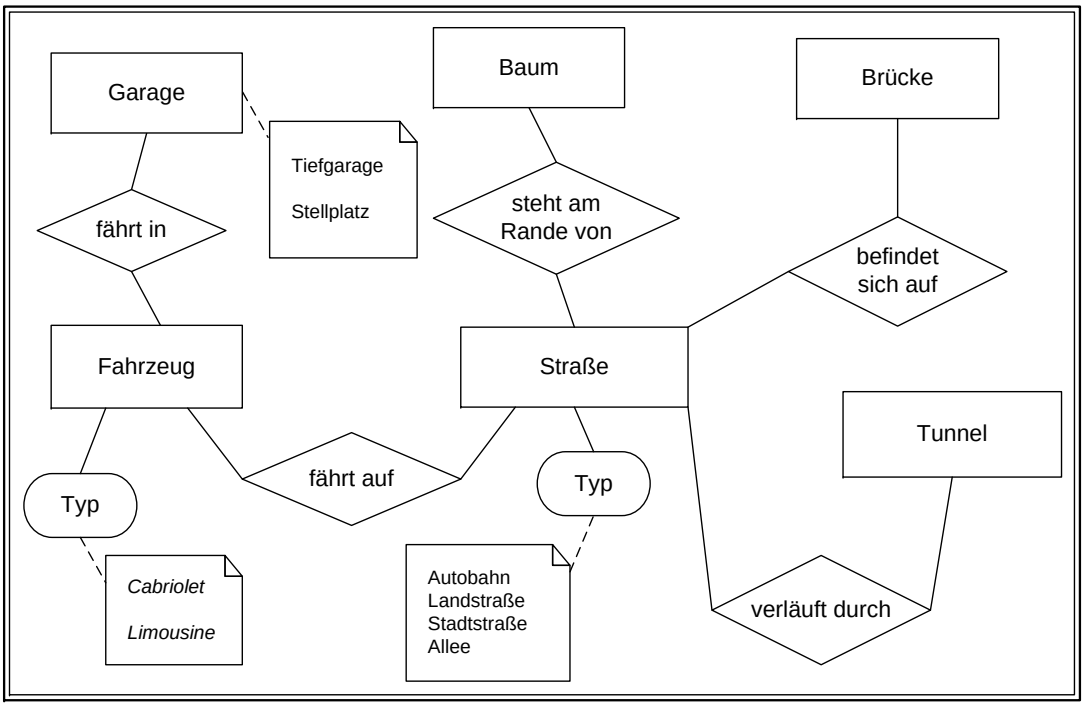

Abbildung A.2: Umgebungsmodell des Scheibentönungssystems

Bei dem Aufgabenmodell kommen zu den aufgeführten Aufgaben noch die Nutzer-Aufgaben hinzu:

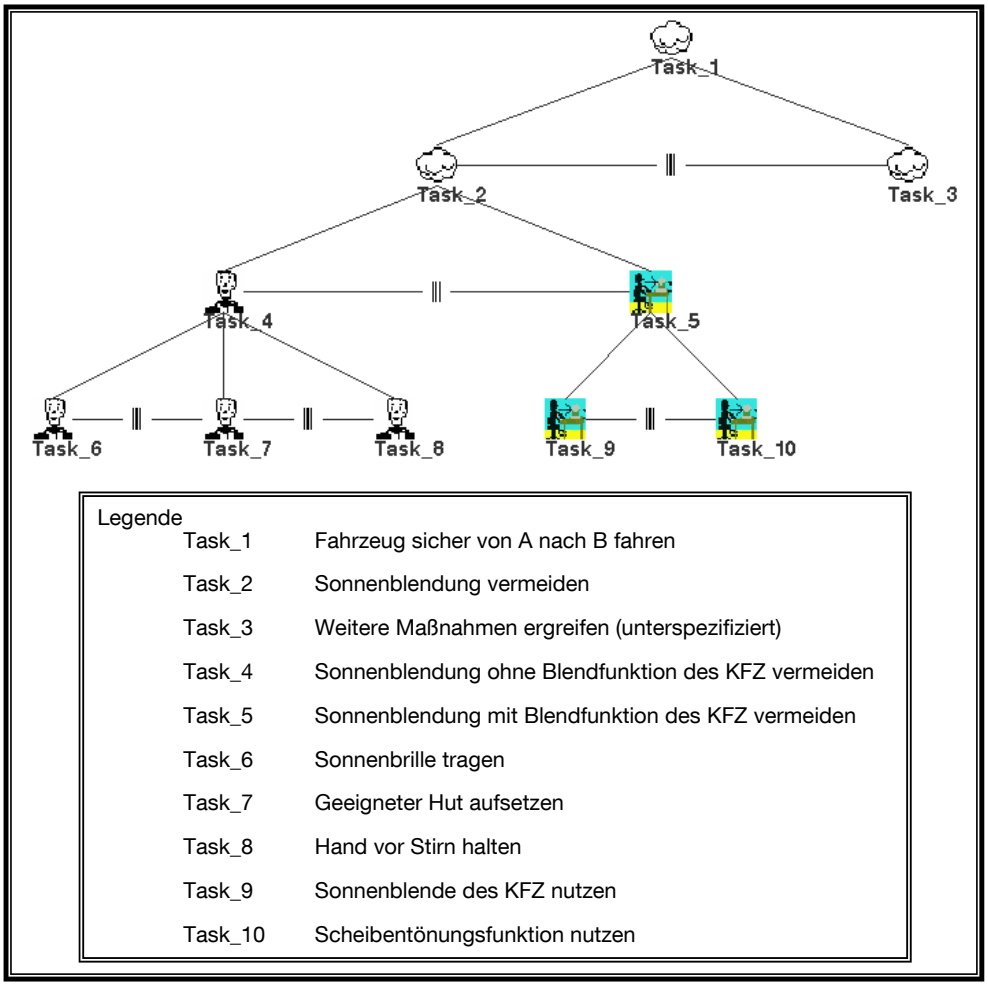

Abbildung A.3: Aufgabenmodell des Scheibentönungssystems

- $Task\_11$ *(In das Fahrzeug einsteigen)*
- $Task\_12$ *(Fahrzeug einparken)*
- $Task\_13$ *(Aus dem Fahrzeug aussteigen)*

Diese Aufgaben sind aufgrund der Übersicht nicht in der Abbildung A.3 dargestellt.

A.6 Angereicherte Szenarien

Angereichertes Szenario USC01

Kurzbezeichnung: Die Scheiben des geparkten Fahrzeugs vollständig tönen

Ziel: G04 und G05 - Wärmereduktion bzw. Diebstahlschutz beim geparkten Fahrzeug

Kontext: Die Aktionen finden im folgenden Kontext statt:

1. Der Fahrer bevorzugt es, die Scheiben seines geparkten Fahrzeugs bis

zu einer Lichtdurchlässigkeit von 0% zu tönen.

2. Der Fahrer parkt das Fahrzeug ein und er verlässt es.
3. Das Fahrzeug befindet sich auf einem Stellplatz.

Vorbedingung:

1. Das Fahrzeug ist geparkt
2. Das Scheibentönungssystem ist aktiv

Aktionen:

1. **Optional:** Der Fahrer drückt auf den Knopf zur Auslösung der vollständigen Tönung der Scheiben
2. Das System tönt die Scheiben des Fahrzeugs bis zu einer Lichtdurchlässigkeit von 0%

Nachbedingung:

1. Die Scheiben des Fahrzeugs sind getönt
2. Die Lichtdurchlässigkeit bei den Scheiben des Fahrzeugs ist 0%
3. Das Scheibentönungssystem ist aktiv

Anmerkung: Vollständige Tönung der Scheiben := Tönung bis zu einer Lichtdurchlässigkeit von 0%.

Angereichertes Szenario USC02

Kurzbezeichnung: Scheiben des Fahrzeugs abhängig von den Lichtverhältnissen tönen

Ziel: G02 - Schutz des Fahrers vor einer Blendung durch die Sonne

Kontext: Die Aktionen finden im folgenden Kontext statt:

1. Der Fahrer bevorzugt es, die Scheiben seines Fahrzeugs beim Fahren bis zu einer maximalen bzw. minimalen Lichtdurchlässigkeit von MAX% bzw. MIN% zu tönen.
2. Der Fahrer fährt das Fahrzeug.
3. Die Kombination aus gemessenem Lichteinfallswinkel und und gemessener Lichtintensität ändert sich. Das Fahrzeug befindet sich auf einer Straße (Autobahn, Landstraße oder Stadtstraße).

Vorbedingung:

1. Das Fahrzeug fährt
2. Das Scheibentönungssystem ist aktiv

Aktionen:

1. Das System berechnet den angemessenen Tönungsgrad N
2. Das System tönt die Scheiben des Fahrzeugs bis zu dem berechneten Tönungsgrad N

Nachbedingung:

1. Die Scheiben des Fahrzeugs sind bis zum Tönungsgrad N getönt
2. Das Scheibentönungssystem ist aktiv

Anmerkung: Der angemessene Tönungsgrad N lässt sich aus dem Lichteinfallswinkel und der äußeren Lichtintensität berechnen. MAX und MIN variieren jeweils nach dem Fahrer. Sie liegen allerdings stets zwischen 50 und 100, wobei MAX in jedem Tupel (MIN, MAX) immer größer als MIN ist. Die Werte hängen beispielsweise von der Sehstärke des Fahrers ab.

Angereichertes Szenario USC03

Kurzbezeichnung: Scheiben des Fahrzeugs abhängig von den Lichtverhältnissen im Tunnelbereich tönen

Ziel: G03 - Freie Sicht im Tunnelbereich

Kontext: Die Aktionen finden im folgenden Kontext statt:

1. Der Fahrer bevorzugt es, die Scheiben seines Fahrzeugs beim Fahren bis zu einer maximalen bzw. minimalen Lichtdurchlässigkeit von MAX% bzw. MIN% zu tönen.
2. Der Fahrer fährt das Fahrzeug.
3. Die Kombination aus gemessenem Lichteinfallswinkel und und gemessener Lichtintensität ändert sich. Das Fahrzeug befindet sich in einem Tunnelbereich.

Vorbedingung:

1. Das Fahrzeug fährt
2. Das Scheibentönungssystem ist aktiv

Aktionen:

1. Das System stellt den aktuellen Tönungsgrad fest
2. Das System berechnet den angemessenen Tönungsgrad N
3. Das System tönt die Scheiben des Fahrzeugs bis zu dem berechneten Tönungsgrad N

Nachbedingung:

1. Die Scheiben des Fahrzeugs sind bis zum Tönungsgrad N getönt
2. Das Scheibentönungssystem ist aktiv

Anmerkung: Der angemessene Tönungsgrad N lässt sich aus dem Lichteinfallswinkel und der äußeren Lichtintensität in dem Tunnelbereich berechnen. MAX und MIN variieren jeweils nach dem Fahrer. Sie liegen zwischen 75 und 100, wobei MAX in jedem Tupel (MIN, MAX) immer größer als MIN ist. Die Werte hängen beispielsweise von der Sehstärke des Fahrers ab.

Angereichertes Szenario USC04

Kurzbezeichnung: Scheiben des Fahrzeugs bei wechselnden Lichtverhältnissen im Allee-Bereich angemessen tönen

Ziel: G06 - Schutz des Fahrers vor einer Blendung durch variierende Lichtverhältnisse

Kontext: Die Aktionen finden im folgenden Kontext statt:

1. Der Fahrer bevorzugt es, die Scheiben seines Fahrzeugs beim Fahren bis zu einer maximalen bzw. minimalen Lichtdurchlässigkeit von MAX% bzw. MIN% zu tönen.
2. Der Fahrer fährt das Fahrzeug.
3. Die Kombination aus gemessenem Lichteinfallswinkel und und gemessener Lichtintensität ändert sich. Das Fahrzeug befindet sich in einem Allee-Bereich.

Vorbedingung:

1. Das Fahrzeug fährt
2. Das Scheibentönungssystem ist aktiv

Aktionen:

1. Das System stellt den aktuellen Tönungsgrad fest
2. Das System berechnet den angemessenen Tönungsgrad N
3. Das System tönt die Scheiben des Fahrzeugs bis zu dem berechneten Tönungsgrad N

Nachbedingung:

1. Die Scheiben des Fahrzeugs sind bis zum Tönungsgrad N getönt
2. Das Scheibentönungssystem ist aktiv

Anmerkung: Der angemessene Tönungsgrad N lässt sich aus dem Lichteinfallswinkel und der äußeren Lichtintensität in dem Allee-Bereich berechnen. MAX und MIN variieren jeweils nach dem Fahrer. Sie liegen allerdings stets zwischen 50 und 100, wobei MAX in jedem Tupel (MIN, MAX) immer größer als MIN ist. Die Werte hängen beispielsweise von der Sehstärke des Fahrers ab.

Angereichertes Szenario USC05

Kurzbezeichnung: Deaktivierung des Scheibentönungssystem

Ziel: G07 - Vollständige Abhängigkeit des Fahrers vermeiden

Kontext: Die Aktionen finden im folgenden Kontext statt:

–

Vorbedingung:

1. Das Fahrzeug fährt
2. Das Scheibentönungssystem ist aktiv

Aktionen:

1. Der Nutzer entscheidet für eine Deaktivierung des Systems

2. Der Nutzer drückt auf dem Knopf zur Deaktivierung des Systems
3. Das System enttönt die Scheiben vollständig

Nachbedingung:

1. Die Scheiben des Fahrzeugs sind enttönt
2. Das Scheibentönungssystem ist inaktiv

Anmerkung: Alternativ zur vollständigen Enttönung ist die Beibehaltung der aktuellen Tönung. In den Diskussionen hat sich allerdings ergeben, dass diese Variante nicht als Standard gelten soll. Dieses Szenario kann in jeder Nutzungssituation stattfinden (Keine Einschränkung des Nutzungskontextes).

Angereichertes Szenario USC06

Kurzbezeichnung: Aktivierung des Scheibentönungssystem

Ziel: G07 - Vollständige Abhängigkeit des Fahrers vermeiden

Kontext: Die Aktionen finden im folgenden Kontext statt:

–

Vorbedingung:

1. Das Scheibentönungssystem ist inaktiv

Aktionen:

1. Der Nutzer entscheidet für eine Aktivierung des Systems
2. Der Nutzer drückt auf dem Knopf zur Aktivierung des Systems
3. Das System tönt die Scheiben bis zu einer angemessenen Grad

Nachbedingung:

1. Das Scheibentönungssystem ist aktiv

Anmerkung: Der angemessene Grad der Tönung nach einer Aktivierung des Systems wird von dem System ermittelt. Dieses Szenario kann in jeder Nutzungssituation stattfinden (Keine Einschränkung des Nutzungskontextes).

Angereichertes Szenario USC07

Kurzbezeichnung: Manuelle Einstellung des Scheibentönungsgrads

Ziel: G07 - Vollständige Abhängigkeit des Fahrers vermeiden

Kontext: Die Aktionen finden im folgenden Kontext statt:

–

Vorbedingung:

1. Das Scheibentönungssystem ist aktiv

Aktionen:

1. Der Nutzer stellt manuell den Grad der Tönung ein
2. Das System tönt die Scheiben bis zu dem eingestellten Grad

Nachbedingung:

1. Das Scheibentönungssystem ist aktiv

Anmerkung: Bei der manuellen Einstellung des Tönungsgrad werden lediglich tatsächlich einstellbare Werte angeboten. Dieses Szenario kann in jeder Nutzungssituation stattfinden (Keine Einschränkung des Nutzungskontextes).

A.7 Integriertes Kontextmodell

Abbildung A.4 stellt das Kontextmodell des Scheibentönungssystem dar.

Das Kontextmodell enthält folgende Fakttypen:

(1) **Fahrer (Name)** parkt **Fahrzeug (Typ)** in **Garage (Typ)** ein

(2) **Fahrer (Name)** steigt in **Fahrzeug (Typ)** ein

(3) **Fahrer (Name)** steigt aus **Fahrzeug (Typ)** aus

(4) **Fahrer (Name)** fährt **Fahrzeug (Typ)** auf **Straße (Typ)**

(5) **Fahrer (Name)** bevorzugt **Bevorzugter Scheibentönungsgrad (Grad)**

(6) **Fahrzeug (Typ)** ist im Zustand **KFZ Zustand (Zustand)**

(7) **Bevorzugter Scheibentönungsgrad (Grad)** hängt von **Anzahl-Insassen (#PersKFZ)** und **KFZ Zustand (Zustand)** ab

(8) **Person (Anzahl)** befindet sich in **Fahrzeug (Typ)**

(9) **Berechneter Scheibentönungsgrad (Grad)** beruht auf **Lichteinfallswinkel (Winkel)** und **Äußere Lichtintensität (Intensität)**

Folgende Prädikate sind in den Fakttypen enthalten:

(1) **einparken in**

(2) **einsteigen in**

(3) **aussteigen aus**

(4) **fahren auf**

(5) **bevorzugen**

(6) **im Zustand** z **sein**

(7) **abhängen von** x **und** y

(8) **sich befinden in**

(9) **auf** u **und** v **beruhen**

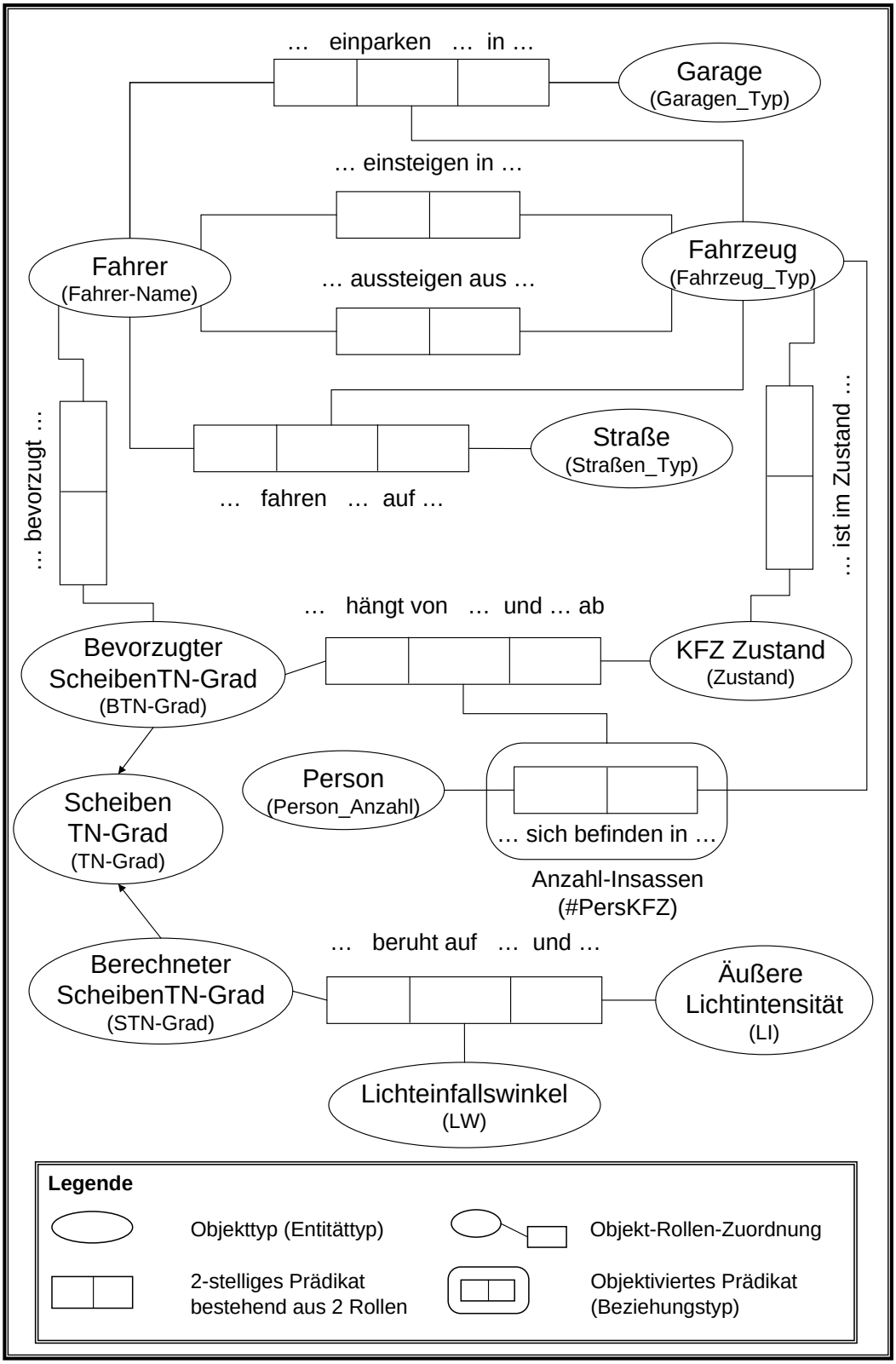

Abbildung A.4: Kontextmodell des Scheibentönungssystems

A.8 Contextual Requirements Chunks

Contextual Requirements Chunks CRC-12

Kurzbezeichnung: Scheiben bis 0% Lichtdurchlässigkeit tönen

Funktionale Anforderung: Req. RQ12
Das System soll die Scheiben bis zu einer Lichtdurchlässigkeit von 0% tönen.

Nutzungssituation: Sit. US12

a) **UM09-1:** Fahrer-Name = Herr Meyer;
Bevorzugter Tönungsgrad beim geparkten leeren Fahrzeug = 100%;

b) **TM09:** Fahrzeug einparken;
Aus dem Fahrzeug aussteigen;

c) **EM10-1:** KFZ_Zustand = GEPARKT;
# Personen im Fahrzeug = 0;

Illustrierendes Szenario: ISce. ISC12
Herr Meyer parkt sein Fahrzeug auf einem Stellplatz ein und verlässt das Fahrzeug. Es befindet sich keine Person in dem Fahrzeug. Aufgrund seiner Vorliebe die Scheiben seines Fahrzeugs im geparktem Zustand bis zu einer Lichtdurchlässigkeit von 0% zu tönen falls sich keine Person in dem Fahrzeug befindet, tönt das Systems die Scheiben des Fahrzeugs automatisch bis zu einer Lichtdurchlässigkeit von 0%.

Contextual Requirements Chunks CRC-13

Kurzbezeichnung: Scheiben bis 50% Lichtdurchlässigkeit tönen

Funktionale Anforderung: Req. RQ13
Das System soll die Scheiben beim geparkten Fahrzeug bis 50% Lichtdurchlässigkeit tönen.

Nutzungssituation: Sit. US13

a) **UM09-2:** Fahrer-Name = Herr Meyer;
Bevorzugter Tönungsgrad beim geparkten nicht leeren Fahrzeug = 50%;

b) **TM09:** Fahrzeug einparken;
Aus dem Fahrzeug aussteigen;

c) **EM10-2:** KFZ_Zustand = GEPARKT;
# Personen im Fahrzeug > 0;

Illustrierendes Szenario: ISce. ISC13
Herr Meyer parkt sein Fahrzeug auf einem Stellplatz ein und verlässt das Fahrzeug. Es befindet sich allerdings eine Person in dem Fahrzeug. Aufgrund seiner Vorliebe die Scheiben seines Fahrzeugs im geparktem Zustand bis zu einer Lichtdurchlässigkeit von 50% zu tönen falls sich mindestens eine Person in dem Fahrzeug befindet, tönt das Systems die Scheiben des Fahrzeugs automatisch bis zu einer Lichtdurchlässigkeit von 50%.

Contextual Requirements Chunks CRC-10

Kurzbezeichnung: Tönungsgrad im Allee-Bereich anpassen

Funktionale Anforderung: Req. RQ10
Das System soll die Scheiben des Fahrzeugs bei wechselnden Lichtverhältnissen bis zum relativen Mittelwert μ tönen.

Nutzungssituation: Sit. US10

a) **UM02:** Fahrer-Name = Herr Meyer; Bevorzugte Intervall des Tönungsgrads beim fahrenden Fahrzeug = $[60\%LD, 100\%LD]$

b) **TM02:** Fahrzeug fahren;

c) **EM04:** Straßen_Typ = Allee;

Illustrierendes Szenario: ISce. ISC10
Herr Meyer fährt sein Fahrzeug und kommt in einen Allee-Bereich. Das System ermittelt den angemessenen Tönungsgrad abhängig von der Lichtintensität und dem Lichteinfallswinkel. Der ermittelte Tönungsgrad ist so, dass nach der Tönung der Scheiben die Lichtdurchlässigkeit zwischen 60% und 100% liegt. Als Tönungsgrad wird der relative Mittelwert des letzten fünf Sekunden-Fahrt verwendet (Erläuterungen hierzu sind im Entwicklerhandbuch DevGuide23). Das System tönt die Scheiben bis zu dem ermittelten Tönungsgrad.

Contextual Requirements Chunks CRC-11

Kurzbezeichnung: Tönungsgrad bei normalen Lichtverhältnissen anpassen

Funktionale Anforderung: Req. RQ11
Das System soll die Scheiben des Fahrzeugs abhängig von der äußeren Lichtintensität und dem Lichteinfallswinkel tönen.

Nutzungssituation: Sit. US11

a) **UM02:** Fahrer-Name = Herr Meyer; Bevorzugte Intervall des Tönungsgrads beim fahrenden Fahrzeug = $[60\%LD, 100\%LD]$

b) **TM02:** Fahrzeug fahren;

c) **EM02:** Straßen_Typ = Autobahn $\vee$ Landstraße $\vee$ Stadtstraße;

Illustrierendes Szenario: ISce. ISC11
Herr Meyer fährt sein Fahrzeug auf die Autobahn (oder die Landstraße bzw. die Stadtstraße). Das System ermittelt den angemessenen Tönungsgrad abhängig von der Lichtintensität und dem Lichteinfallswinkel. Der ermittelte Tönungsgrad ist so, dass nach der Tönung der Scheiben die Lichtdurchlässigkeit zwischen 60% und 100% liegt. Das System tönt die Scheiben bis zu dem ermittelten Tönungsgrad.

Contextual Requirements Chunks CRC-14

Kurzbezeichnung: Tönungsgrad im Tunnelbereich anpassen

Funktionale Anforderung: Req. RQ14
Das System soll die Scheiben des Fahrzeugs bis zu einer Lichtdurchlässig-

keit von $n\% \in [75, 100]$ tönen.

Nutzungssituation: Sit. US14

a) **UM:** -

b) **TM02:** Fahrzeug fahren;

c) **EM03:** Straßen_Typ = Tunnel;

Illustrierendes Szenario: ISce. ISC14

Herr Meyer fährt sein Fahrzeug und kommt in einen Tunnelbereich. Das System ermittelt den angemessenen Tönungsgrad abhängig von der Lichtintensität und dem Lichteinfallswinkel. Der ermittelte Tönungsgrad ist so, dass nach der Tönung der Scheiben die Lichtdurchlässigkeit zwischen 75% und 100% liegt. Das System tönt die Scheiben bis zu dem neu ermittelten Tönungsgrad.

Anhang B

Gliederung eines Anforderungsdokuments für kontextsensitive Anwendungen

Inhaltsangabe

B.0 Vorbemerkung

1. *Die hier vorgeschlagene Gliederung eines Anforderungsdokuments eignet sich für kontextsensitive Anwendung. Eine Untersuchung der Gliederung auf deren Eignung für nicht-kontextsensitive Anwendungen ist im Rahmen dieser Arbeit nicht durchgeführt worden.*
2. *Die Grundidee dieses Gliederungsvorschlags besteht darin, sowohl Anforderungen als auch deren zugehörige Nutzungssituationen gemeinsam (integriert) zu entwickeln. Kapitel B.7 enthält eine Auflistung sowohl der Anforderungen in Form von Funktionen als auch der Situationen, in welchen erstere auftreten.*
3. *Im Gegensatz zu gängigen Gliederungsvorschlägen für Anforderungsdokumente – wie etwa dem Volere-Template [Robertson und Robertson, 2007] oder dem IEEE-Standard 830 [IEEE, 1998] – konzentriert sich der an dieser Stelle vorgestellte Vorschlag auf die spezifischen Fragestellungen kontextsensitiver Anwendungen. Themen wie Qualitätsanforderungen oder allgemein nicht-funktionale Anforderungen werden in dieser Gliederung nicht behandelt.*

B.1 Vorwort

Dieses Kapitel hat einen rein einleitenden Charakter, und enthält nützliche Informationen und Anmerkungen zum Zweck des Anforderungsdokuments; beispielsweise die Eigenschaft, dass das Dokument Funktionen, Situationen und illustrierende Szenarien beschreibt, welche das zu entwickelnde System realisieren muss.

Das Vorwort darf keine Anforderungen, Situationsbeschreibungen, Bilder oder Tabellen enthalten.

B.2 Einleitung

Kurze einführende Worte angeben.

B.2.1 Zielsetzung

In diesem Kapitel werden die übergeordneten Ziele der Entwicklung des Systems definiert (z.B. Unterstützung im OP-Saal, Unterstützung des täglichen Ablaufs eines Nutzers oder auch Minimierung des Unfallrisikos im Autoverkehr). Es geht darum festzulegen, was die Entwicklung des Systems bezwecken soll.

- *Beschreibung der abstrakten Ziele als Vision des Systems*
- *Festlegung der Systemgrenze*
- *Auflistung der Hauptfunktionalitäten des Systems*
- *Rechtfertigung für das zu erstellende System (wirtschaftlicher etc.)*
- *Systemgrenzen festlegen (enden vermutlich bei verwendeter Sensorik)*

B.2.2 Einsatzumgebung des Systems

Kurze Beschreibung der Einsatzumgebung des Systems

- *Wie sieht die Einsatzumgebung des Systems aus?*
- *Angabe einer grob granulare Beschreibung der umgebenden Entitäten*
- *In diesem Kapitel kann ein Kontextdiagramm (verfügbare Sensoren, Kommunikations- bzw- Interaktionspartner etc.)*

B.2.3 Relevante Nutzer

Kurze Beschreibung der zukünftigen Nutzer des Systems

B.3 Verfolgte Ziele

Hier geht es um die Auflistung der mit der Entwicklung des Systems verfolgten Ziele.

B.3.1 Primäre Ziele

Beschreibung aller primären Ziele des Systems

B.3.2 Sekundäre Ziele

Beschreibung aller sekundären Ziele des Systems. Diese führen zu den so genannten „nice-to-have“ Funktionalitäten des Systems

B.4 Aktuelle Abläufe

In diesem Kapitel werden die aktuelle Abläufe (d.h Abläufe ohne Einsatz des geplanten Systems) beschrieben. Diese Abläufe charakterisieren die Ist-Situation in der Umgebung, in die das zu entwickelnde kontextsensitiven System eingesetzt werden. Es bietet sich Szenarien aufzuschreiben, anhand dessen die Innovation hinter dem kontextsensitiven System motivieren. Meistens

geht es darum, die aktuelle Abläufe zu optimieren, den Alltag einer Person komfortabler zu gestalten (z.B. Interaktion mit einem vorhandenen System auf ein Minimum reduzieren, die Nutzung eines Systems in möglichst vielen Situationen zu ermöglichen).

- *Falls vorhanden Anwendungsfälle des aktuellen Systems beschreiben*
- *Auf jeden Fall sind Szenarien nötig, welche die aktuelle Abläufe ohne Einsatz des geplanten Systems beschreiben. Bei dieser Art von Szenarien sind keine Angaben zum Kontext erforderlich.*

B.4.1 Anwendungsfälle des aktuellen Systems

Beschreibung der Anwendungsfälle des aktuellen Systems, falls vorhanden.

B.4.2 Aktueller Ablauf 1 bis m

Beschreibung des ersten aktuellen Ablaufs bzw. Nutzungsszenarien zu dem ersten Anwendungsfall des Systems.

B.5 Zukünftige Abläufe

Bei zukünftigen Abläufen (d.h. Abläufe mit dem System im Einsatz) geht es um visionäre Szenarien. Die Visionäre Szenarien haben die Eigenheit, dass sie auch Informationen über die zukünftigen Nutzungssituationen enthalten. Es ist für die spätere Ermittlung des Nutzungskontexts des Systems wichtig, bereit bei der Ermittlung der Szenarien Informationen über benötigte Kontexte zu berücksichtigen.

B.5.1 Um Kontextinformation erweiterte Anwendungsfälle

Beschreibung angedachte Anwendungsfälle des Systems.

B.5.2 Erweiterte Szenarien zum Anwendungsfall 1 bis n

Beschreibe mögliche um den Kontext erweiterte Szenarien zu dem zweiten Anwendungsfall des Systems.

B.6 Ermittlung des Kontextes

Kurze Einführung des erarbeiteten Kontextmodells.

B.6.1 Benutzermodell

Das erarbeitete Benutzermodell (User Model) einfügen und seine einzelnen Elemente beschreiben.

B.6.2 Aufgabenmodell

Das erarbeitete Aufgabenmodell (Task Model) einfügen und seine einzelnen Elemente beschreiben.

B.6.3 Umgebungsmodell

Das erarbeitete Umgebungsmodell (Environment Model) einfügen und seine einzelnen Elemente beschreiben.

B.6.4 Plattformmodell

Das erarbeitete Plattformmodell (Platform Model) einfügen und seine einzelnen Elemente beschreiben. Falls kein Plattformmodel erforderlich ist, dann eine adäquate Begründung angeben.

B.6.5 Interaktionsmodell

Das erarbeitete Interaktionsmodell (Interaction Model) einfügen und seine einzelnen Elemente beschreiben. Falls kein Interaktionsmodell erforderlich ist, dann eine adäquate Begründung angeben.

B.6.6 Darstellungsmodell

Das erarbeitete Darstellungsmodell (Presentation Model) einfügen und seine einzelnen Elemente beschreiben. Falls kein Darstellungsmodell erforderlich ist, dann eine adäquate Begründung angeben.

B.6.7 Integriertes Kontextmodell

Die einzelnen Modelle über Fakten integrieren. Dafür werden Fakttypen (Situationstypen) über die Modelle definiert. In einem Situationstyp werden Situationen identischer Typ zusammengefasst. Situationen sind Aussagen über Belegungen der einzelnen Modelle und ihre Beziehungen untereinander. Sie charakterisieren die Nutzung des Systems. Eine Situation kann nicht von zwei unterschiedlichen Situationstypen sein. Jede Situation ist eindeutig bestimmt durch dem Zeitstempel und die Belegung der Modelle.

B.7 Contextual Requirements Chunks

In diesem Kapitel werden die Ziele des Systems, wie sie im Kapitel B.3 definiert sind, verfeinert. Die resultierende Anforderungen an das zu entwickelnde System werden mit den Situationen in Beziehung gesetzt. Das ganze wird nach Situationstypen gruppiert. Nach Auflistung der Situationstypen samt Beschreibung der einzelnen Situationen und der darin zu genügenden Anforderungen werden die entstehenden Contextual Requirements Chunks konsolidiert.

B.7.1 Funktionale Anforderungen

Die im Kapitel B.3 aufgeführten Ziele werden verfeinert und auf Anforderungen/Funktionen des Systems abgebildet. Es entsteht eine baumartige Struktur. Die Knoten repräsentieren entweder Ziele oder Funktionen des Systems. Die Kanten werden mit Situationstypen (auch Situationsklassen) bzw. Situationen annotiert. Eine weitere Möglichkeit zur Identifikation der funktionalen Anforderungen besteht in der Konkretisierung der Stakeholder-Ziele mittels Szenarien gefolgt von einer Ableitung der Anforderungen aus den Szenarien.

B.7.2 Situationsklasse 1 bis r

Zu dem ersten Situationstyp gehörenden Anforderungen bzw. Funktionen des Systems (d.h. Anforderungen bzw. Funktionen, die auf Knoten unterhalb einer mit diesem Situationstyp beschrifteten Kanten liegen) resultierend aus der Verfeinerung der Ziele beschreiben.

- Auflistung der Anforderungen bzw. Funktionen, die zu diesem Situationstyp gehören.
- Die Anforderungen mit konkreten Situationen verbinden, in welchen sie gelten. Ebenfalls auf entsprechende Nutzungsszenarien verweisen.
- Auflistung der übergreifenden Anforderungen. Also Anforderungen, die für den gesamten Situationstyp gelten, unabhängig von den konkreten Situationen.

B.7.3 Konsolidierte Contextual Requirements Chunks

- *Konsolidiere die Contextual Requirements Chunks*
- *Priorisiere die Anforderungen (also die Paare aus Situationen und Funktionen)*
- *Dokumentiere Konflikte und deren Auflösung*

B.8 Zurückgestellte Anforderungen

Ziel ist eine Dokumentation der zwar analysierte aber nicht abgedeckte Nutzungssituationen und ihre Anforderungen zu gewährleisten.

- *Alle Anforderungen auflisten, die ursprünglich analysiert wurden, aber im Laufe der Konsolidierung der Contextual Requirements Chunks bewusst zurückgelegt wurden.*

- *Die Entscheidung, bestimmte Anforderungen zurückzulegen, ausführlich begründen.*

B.9 Mitgeltende Unterlagen

Die mitgeltenden Unterlagen (Gesetze, Normen, auftraggeberspezifische Regelwerke, etc.) auf die im Text des Lastenheftes verwiesen wird, sollten nach Dokumentenarten differenziert aufgeführt werden. Es gelten die am Ausgabedatum des LH gültigen mitgeltenden Unterlagen. Darüber hinaus ist eine Bezugsquelle (alternativ: Ansprechpartner) der Informationen für den Auftragnehmer zu benennen. Wird auf Vorschriften (Gesetze, Verordnungen und dergleichen) verwiesen, muss der Bezug derart sein, dass die Beschaffung möglich ist.

Als mitgeltende Unterlagen werden in diesem Zusammenhang ausschließlich die im Lastenheft zitierten Dokumente verstanden. Die Zugangsadresse und -modalitäten für die Zulieferer sollen an dieser Stelle ebenfalls genannt werden.

Glossar

A

Adaption – Adaptivität Im alltäglichen Sprachgebrauch versteht man unter dem Begriff Adaption ein Anpassungsvermögen an die Umwelt. Anpassungsvermögen steht für die Fähigkeit, sich den gegebenen Verhältnissen anzupassen, sich nach jeweiligen Umständen zu richten. Die allgemeine Fähigkeit, sich in einer bestimmten Situation nach den durch die Umwelt vorgegebenen Verhältnissen zu richten nennt man Adaptivität. Der Anpassungsprozess wird Adaption genannt. Sie kann als ein Spezialfall der Parametrisierung betrachtet werden, da bei der Adaption genau wie bei der Parametrisierung das Systemverhalten über definierte Parameter beeinflusst wird. Der Unterschied besteht vor allem darin, dass im Falle von Adaption das System selbst die Parameterwahl durchführt oder zumindest den Benutzer dabei unterstützt. Darüber hinaus findet Adaption im Gegensatz zu Parametrisierung vorrangig zur Laufzeit statt.

Adaptives System Ein adaptives System (genauer ein kontextadaptives oder auch kontextsensitives System) ist ein Computer-basiertes System mit der Fähigkeit (1) *Änderungen in der Domäne*, in der sie genutzt wird, zu erkennen, und (2) sein *beobachtbares Verhalten*, selbst ohne direkte *Nutzerinteraktion*, so zu ändern, dass dieses der veränderten Domäne *angepasst* ist. Bei der Charakterisierung eines adaptiven Systems ist die Betrachtung des Nutzers sowie des *Kontextes* von Bedeutung. Adaptive Systeme unterscheiden sich nicht grundlegend von anderen reaktiven Systemen. Allerdings muss dem Aspekt der Konfigurationsbildung, der Modellierung der Umgebung bzw. des Kontextes und der Automatisierung der Konfigurationswahl bei der Systementwicklung Rechnung getragen werden. Die Konfigurationsbildung bietet eine Erleichterung bei der Handhabung der Komplexität in multifunktionalen Systemen (siehe auch *Adaption*, *Kontext* und *System*).

Anforderung Eine Anforderung ist (1) eine Bedingung, Fähigkeit oder Eigenschaft, die ein Stakeholder für ein Produkt oder einen Prozess fordert, um ein Problem zu lösen oder ein Ziel zu erreichen; (2) eine Bedingung/Fähigkeit/Eigenschaft, die ein System erfüllen/haben muss, um einen Vertrag, einen Standard, eine Spezifikation oder andere formal vorgegebene Dokumente zu erfüllen; (3) eine dokumentierte Repräsentation

einer Bedingung, Fähigkeit oder Eigenschaft wie in (1) oder (2) definiert [IEEE, 1990].

Anwendungssystem Ein Anwendungssystem ist ein System, das Software-Komponenten enthält. Im weiteren Sinne umfasst es eine Menge von inhaltlich zusammengehörigen Aufgaben, die dafür verantwortlichen Menschen als Aufgabenträger und die zu ihrer Erfüllung eingesetzte technische Ausstattung. Im engeren Sinn wird darunter oft ein Anwendungsprogramm (d.h. das reine SW-System) verstanden, das eine spezifische Aufgabe unterstützt. Als Synonym dafür wird der Begriff von *Anwendung* verwendet. Ein Beispiel für Anwendung ist ein Standardprogramm für die Finanzbuchhaltung [Gesellschaft für Informatik, 1998].

Aufgabenmodell Ein Aufgabenmodell (auch Task Modell) in einer kontextsensitiven Anwendung ist ein Modell, das Annahmen über alle Aktivitäten des Nutzers bei der Nutzung der Anwendung enthält, die für die Adaption der Anwendung relevant sind. Es ist ein potentieller Bestandteil des Kontextmodells.

B

Benutzermodell Ein Benutzermodell (auch Nutzermodell) in einer kontextsensitiven Anwendung ist ein Modell, das Annahmen über alle Benutzeraspekte enthält, die für die Adaption der Anwendung relevant sind. Zu den Annahmen gehören Annahmen über Nutzerdaten (demographische Daten und Präferenzen des Nutzers) und Nutzungsdaten (Benutzungsverhalten). Es ist ein potentieller Bestandteil des Kontextmodells.

C

Context-Awareness Context-Awareness means that a system is able to observe its environment using some sort of sensor and that it is aware of this environment and is able to react to changes in it [Schilit et al., 1994].

Contextual Requirements Chunk (CRC) Ein Contextual Requirements Chunk ist ein Tripel bestehend aus den charakteristischen Informationen über eine Nutzungssituation, der funktionalen Anforderungen sowie dem illustrierenden Szenario. Es gilt, dass jedes Szenario in einem CRC sowohl von der in dem CRC angegebenen Anforderung als auch von der Situation abhängig ist. Es gilt auch, dass viele Anforderungen in einer gleichen Situation vorkommen können. Es gilt allerdings die Einschränkung, dass widersprechende Anforderungen nicht in einer gleichen Situation vorkommen dürfen. Jedes Contextual Requirements Chunk x ist der Form $x \in REQ \times SIT \times SCE$. Dabei sind die Bestandteile von x wie folgt mit einander verbunden:

WENN $Nutzungssituation$ **DANN** $Funktionale\ Anforderung$
MIT $Illustration\ anhand\ eines\ Anwendungsszenarios$

.

K

Kalibrierung Die Kalibrierung eines kontextadaptiven Systems ist die nachträgliche manuelle Anpassung der zugrunde liegenden Logik der Adaption des Systems. Sie ist also eine Adaption der Adaption, und ihre Umsetzung erfordert eine entsprechende Expertise über das Adaptionsverhalten des Systems. Sie ist die vom Nutzer zeitversetzt und direkt beeinflusste Anpassung des Adaptionsteilsytems eines kontextadaptiven Systems. Sie ist eine zusätzliche Adaption der Adaption. Die Nutzerinteraktion muss nicht notwendigerweise bewusst sein, aber direkt. Direkt heißt, dass die Handlungsweise nicht interpretiert wird. Ein Beispiel dafür wäre der *one-button* bei mobilen Endgeräten. Der Benutzer weiß dass, wenn er den Knopf drückt, seine Unzufriedenheit dem System signalisiert wird. Er ist sich aber nicht bewusst, wie sich das auf das System auswirkt.

Kontext Kontext ist die hinreichend genaue Charakterisierung der Situationen eines Systems anhand von für die automatische Anpassung des Verhaltens des Systems relevanten und von dem System wahrnehmbaren Informationen.

Kontextadaptation Kontextadaption bezeichnet die automatisierte Anpassung des beobachtbaren Verhaltens eines Systems an seinen Kontext. Das Verhalten des Systems ist bestimmt durch die an der Schnittstelle beobachtbaren Ein- und Ausgaben. Die Ausgaben des Systems hängen dabei nicht nur von den Nutzereingaben ab, sondern auch von den über Sensorik gesammelten Kontextinformationen.

M

Methode Eine Methode bedeutet allgemein eine Grundlage für ein planmäßiges, folgerichtiges Verfahren, Vorgehen, Forschen oder Handeln. In der Softwaretechnik bezeichnet eine Methode eine Vorgehensweise bei der Erstellung von Software. Eine Gesamtheit von Methoden wird Methodik genannt (Siehe auch Glossareintrag *Methodik*).

Methodik Eine Methodik ist eine Gesamtheit von *Methoden*, als Teildisziplin einer Wissenschaft auch die Lehre von den in dieser Wissenschaft angewandten Methoden. Methodik ist zu unterscheiden von Methodologie, der theoretischen Reflexion über Methoden eines Fachgebiets.

Modell Ein Modell ist eine idealisierte, vereinfachte, in gewisser Hinsicht ähnliche Darstellung eines Gegenstands, Systems oder sonstigen Weltausschnitts mit dem Ziel, daran bestimmte Eigenschaften des Vorbilds besser studieren zu können [Gesellschaft für Informatik, 1998]. In der Modelltheorie wird mit Modell eine vereinfachende Abbildung der Wirklichkeit bezeichnet. Wichtigste Merkmale eines Modell sind: das Abbildungsmerkmal (jedes Modell ist Abbild oder Vorbild), das Verkürzungsmerkmal (jedes Modell abstrahiert) und das pragmatische Merkmal (jedes Modell wird im Hinblick auf einen Verwendungszweck geschaffen).

P

Parametrierung – Parametrisierung Die Parametrierung eines Systems ist die konkrete Einstellung aller Parameter und Stammdaten in der Datenhaltung dieses Systems zur Realisierung individueller Funktionalitäten. Sie ist also das Einrichten der Parameter und Stammdaten im Rahmen der Systemeinführung zur Herstellung der für den konkreten Betrieb notwendigen Individualität. Im Zusammenhang mit Parametrierung taucht der Begriff Parametrisierung auf, welcher als Realisierungstechnik dient. Durch Parametrisierung werden Systemeigenschaften realisiert, die sich häufig ändern oder z.B. benutzerspezifisch sind, und eine Veränderung des Systems ohne Neuübersetzung ermöglicht. Ob eine erneute Verteilung bei Parameteränderungen nötig ist, hängt vom Parametrisierungsmechanismus ab, d.h. je nach Mechanismus kann eine Anpassung der Parameter zur Entwicklungs- oder Laufzeit erfolgen.

R

Requirements Engineering (RE) Requirements Engineering ist innerhalb des Systems und Software Engineering eine iterative systematische Vorgehensweise mit dem Ziel, eine explizite, mit den Stakeholdern abgestimmte Anforderungs- und Systemspezifikation zu erstellen, die als Basis für die weitere Entwicklung dient. Kernaufgaben des Requirements Engineering sind das gemeinsame (1) Erfassen und Herausarbeiten von Anforderungen (Elicitation), (2) Analysieren und Modellieren von Anforderungen, (3) Abstimmen und Validieren von Anforderungen, (4) Spezifizieren und Dokumentieren der Abstimmungsergebnisse, (5) und das Verwalten der Artefakte des RE (Requirements Management) (angelehnt an [IEEE, 1990]).

S

Situation Eine Situation (auch Nutzungssituation) ist die augenblickliche Lage, die Verhältnisse, die Umstände oder der allgemeine (objektive) Zustand, in dem sich jemand (oder etwas) befindet. In einer kontextsensitiven Anwendung wird der Begriff von Nutzungssituation verwendet. Dabei ist sie je nach Anwendungsfall durch Informationen über (1) den Nutzer, (2) die Aufgaben, die der Nutzer durchführt, (3) die Umgebung, in der der Nutzer die Anwendung nutzt, (4) die Plattform, auf der die Anwendung läuft, (5) die Interaktionen, die den Nutzer benötigt, um die Aufgaben durchzuführen, und (6) die Art/Modalität der Präsentation bzw. Darstellung der Elemente der Interaktionen zwischen dem Nutzer und dem System charakterisiert. Sie bildet eine Instanz des Kontextmodells zu einem gegebenen Zeitpunkt. Situationen liefern die notwendigen Informationen zur Feststellung der Zielen, der Anforderungen und der Randbedingungen der Systementwicklung sowie des angemessenen Verhaltens.

Spezifikation Eine Spezifikation bezeichnet die präzise Darstellung einer Menge von Aussagen, die die Funktionalität bzw. die Leistungen oder die Eigenschaf-

ten eines bestimmten Problems oder Produktes beschreibt. Es ist im Idealfall eine deklarative Beschreibung. Idealerweise ist die Spezifikation in einer formalen Sprache festgehalten, um missverständliche Interpretationen der Aussagen vermeiden und ihre Vollständigkeit überprüfen zu können [IEEE, 1990].

Stakeholder Ein Stakeholder (auch Interessenvertreter) bezeichnet eine Einzelperson, eine Gruppe von Personen oder eine Organisation, die Anteil oder Interesse an dem System, und dementsprechend an den Anforderungen, haben [Mitchell et al., 1997, Sharp et al., 1999, Nuseibeh und Easterbrook, 2000]. Stakeholder werden von der Entwicklung, vom Einsatz und vom Betrieb des zu entwickelnden Systems betroffen und haben Ziele, die sie mit der Systementwicklung verfolgen. Eine Spezifikation (bzw. Modellierung) der Anforderungen aus Sicht der unterschiedlichen Stakeholder ist ein essentieller Teil der Softwareentwicklung.

System Ein System ist ein abstrakter, d.h. aus der Sicht eines Betrachters oder einer Gruppe von Betrachtern bestimmter und explizit von seiner Umgebung abgegrenzter Gegenstand [Gesellschaft für Informatik, 1998]. Systeme sind aus Teilen (Systemkomponenten oder Subsystemen) zusammengesetzt, die untereinander in verschiedenen Beziehungen stehen können. Systemteile, die nicht weiter zerlegbar sind oder zerlegt werden sollen, werden als Systemelemente bezeichnet. Ein System bezeichnet also ein Gebilde, dessen wesentliche Elemente so aufeinander bezogen sind und in einer Weise wechselwirken, dass sie (aus einer übergeordneten Sicht heraus) als aufgaben-, sinn- bzw. zweckgebundene Einheit angesehen werden (können) und sich in dieser Hinsicht gegenüber der sie umgebenden Umwelt auch abgrenzen.

Systemspezifikation Eine Systemspezifikation spezifiziert ein Lösungskonzept für eine erfasste Problemstellung, idealerweise in einer Anforderungsspezifikation festgehalten, in Form eines Systemkonzeptes. Feste Teile der Systemspezifikation sind (1) eine Abgrenzung zur Umgebung des Systems, (2) eine Beschreibung der Funktionalität/Verhalten und der Eigenschaften des Systems, und (3) eine Beschreibung der Schnittstellen [Geisberger, 2005].

Systemverhalten Von Verhalten eines Systems spricht man dann, wenn eine Veränderung des Zustandes bzw. der Zustandsgrößen des Systems auf der Makroebene beobachtet werden kann. Als Ereignis wird der Übergang von einem Zustand in einen anderen bezeichnet. Die Veränderung kann selbständig, ohne Einflüsse von außen erfolgen oder mit einem Einfluss von außen zusammenhängen. Beobachtbar ist auch, dass ein System trotz eines Einflusses von außen keine Änderung zeigt. Das Verhalten eines Systems kann direkt oder indirekt (zum Beispiel über einen Kreislauf) auf seinen eigenen Zustand wieder zurückwirken.

T

Technik Eine Technik ist eine Prozedur, eventuell mit einer vorgeschriebenen Notation, zur Durchführung einer Entwicklungsaktivität. Am häufigsten wird der Begriff dazu verwendet, um allein Notationen zu bezeichnen. Beispiele für Techniken sind Datenmodellierung mit *ER-Diagrammen* oder Durchführung von Interviews mittels *strukturierten Interviews-Schablonen*. Techniken können beispielsweise danach klassifiziert werden, wie formal die zugrundeliegende Notation ist (z.B. natürlch sprachlich, strukturierte Graphiken) oder welche Entwicklungsaktivität sie unterstützen (z.B. Datenmodellierung, Prozessmodellierung, Interaktionsdesign).

U

Ubiquitous Computing (UbiComp) Ubiquitous Computing ist die direkte oder indirekte Nutzung computer-basierter Anwendungssysteme in möglichst vielen Situationen (z.B. Orte, Zeiten, Tätigkeiten) eines Benutzers. (Angelehnt an [Kephart und Chess, 2003, Fahrmair, 2005]).

Umgebungsmodell Ein Umgebungsmodell (auch Modell der Einsatzumgebung) in einer kontextsensitiven Anwendung ist ein Modell, das Annahmen über alle Aspekte der Umgebung enthält, in der die Anwendung eingesetzt bzw. genutzt wird (Einsatzumgebung), welche für die Adaption der Anwendung relevant sind. Es ist ein potentieller Bestandteil des Kontextmodells.

Usability Der englischen Begriff Usability bezeichnet eine Qualitätseigenschaft von Software und entspricht dem deutschen Gebrauchstauglichkeit oder Benutzerfreundlichkeit. Die Modelle, Methoden, Werkzeuge und Prozesse zur Konstruktion und Evaluierung von Usability sind Gegenstand des Usability Engineering [Wikimedia Foundation, 2007]. Die Gebrauchstauglichkeit bezeichnet den Eignungsgrad einer Sache oder eines Systems in Bezug auf seinen Verwendungszweck in einem bestimmten Benutzungskontext. Die Gebrauchstauglichkeit beruht unter anderem auf Gebrauchseigenschaften und den Bedürfnissen des Nutzers. Somit gibt es neben einer objektiven Beurteilung auch eine subjektive Beurteilung, die von Individuum zu Individuum sehr unterschiedlich ausfallen kann. Die Gebrauchstauglichkeit ist in der Normenreihe DIN EN ISO 9241 in Teil 11 definiert als das Produkt aus Effektivität, Effizienz und Zufriedenheit.

V

Validierung Die Validierung ist die Feststellung der Übereinstimmung eines Produktes mit den Anforderungen und Bedürfnissen des Kunden bzw. des Nutzers. Sie dient dazu die Frage “Baue ich das richtige Produkt?” zu beantworten. Die Validierung ist eine zentrale Aufgabe des Requirements Engineering. Durch die Erstellung der Anforderungsspezifikation und einer entsprechenden Systemspezifikation ist es möglich, frühzeitig auf

der Basis der darin enthaltenen Modelle die Übereinstimmung des "fertigen" Produktes mit den Kunden/Nutzerbedürfnissen zu überprüfen, zu verbessern und damit die Problemadäquatheit und die Qualität des zu entwickelnden Produktes sicher zustellen (siehe auch Verifikation) [IEEE, 1990].

Verifikation Die Verifikation ist der Nachweis, dass ein im Rahmen einer Software bzw. Systementwicklungsaktivität entwickeltes Produkt die Anforderungen erfüllt, die zuvor für dieses Produkt definiert worden sind. Sie dient dazu die Frage "Baue ich das Produkt richtig?" zu beantworten. Im Requirements Engineering findet neben der Vaildierung auch die Verifikation statt. Erarbeitete (alternative) Anforderungs- und Systemkonzeptmodelle (Spezifikationen) müssen bei der Verifikation auf die Erfüllung von definierten Anforderungen untersucht werden [IEEE, 1990].

Z

Ziel Ein Ziel ist ein bewusst angestrebter zukünftiger Zustand. Ein solcher Zustand muss nach Inhalt, Zeit und Ausmaß bestimmt werden, um erreicht und überprüft werden zu können. Ziele sind Zustände in der Zukunft, die durch menschliches Handeln herbeigeführt werden sollen. Sie haben eine handlungsweisende Funktion.

Abbildungsverzeichnis

Literaturverzeichnis

[Abidi, 1992] Abidi, M. A. (1992). *Data Fusion in Robotics and Machine Intelligence*, chapter Fusion of Multi-Dimensional Data Using Regularization, pages 415–455. Academic Press, Boston, MA.

[Abowd, 1999] Abowd, G. D. (1999). Software Engineering Issues for Ubiquitous Computing. In *Proceedings of the 1999 International Conference on Software Engineering*, pages 75–84.

[Abowd und Mynatt, 2000] Abowd, G. D. und Mynatt, E. D. (2000). Charting Past, Present, and Future Research in Ubiquitous Computing. *ACM Transactions on Computer-Human Interaction*, 7(1):29–58.

[Amann et al., 2004] Amann, K., Reichgruber, T., und Roming, M. (2004). Personalisierung kontextadaptiver Dienste am Beispiel des Cawar@Campus Portals. Systementwicklungsproject, Technische Universität München.

[Amyot und Eberlein, 2003] Amyot, D. und Eberlein, A. (2003). An Evaluation of Scenario Notations and Construction Approaches for Telecommunication Systems Development. *Telecommunication Systems*, 24(1):61–94.

[Antón et al., 2001] Antón, A., Carter, R., Dagnino, A., Dempster, J., und Siege, D. (2001). Deriving Goals from a Use-Case Based Requirements Specification. *Requirements Engineering*, 6(1):63–73.

[Antón, 1996] Antón, A. I. (1996). Goal-based Requirements Analysis. In *Proceedings of the Second International Conference on Requirements Engineering*, pages 136–144.

[Antón, 1997] Antón, A. I. (1997). *Goal Identification and Refinement in the Specification of Software-Based Information Systems*. PhD thesis, Georgia Institute of Technology.

[Antón und Potts, 1998] Antón, A. I. und Potts, C. (1998). The use of goals to surface requirements for evolving systems. In *Proceedings of the 20th international conference on Software engineering*, pages 157–166. IEEE Computer Society Washington, DC, USA.

[Armour und Miller, 2001] Armour, F. und Miller, G. (2001). *Advanced Use Case Modeling – Software Systems*. Addison-Wesley.

[Baldauf et al., 2007] Baldauf, M., Dustdar, S., und Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*,

2(4):263.

[Baldes et al., 2005] Baldes, S., Kröner, A., und Bauer, M. (2005). Configuration and Introspection of Situated User Support. In *13th Workshop on Adaptivity and User Modeling in Interactive Systems – LWA/ABIS 2005*.

[Balzert, 1996] Balzert, H. (1996). *Lehrbuch der Software-Technik: Software-Entwicklung*. Spektrum Akademischer Verlag.

[Bayer et al., 1999] Bayer, J., Flege, O., Knauber, P., Laqua, R., Muthig, D., Schmid, K., Widen, T., und DeBaud, J.-M. (1999). PuLSE: a Methodology to Develop Software Product Lines. In *Proceedings of the 1999 Symposium on Software Reusability, SSR'99*, pages 122–131, New York, NY, USA. ACM.

[Bittner und Spence, 2002] Bittner, K. und Spence, I. (2002). *Use Case Modeling*. Addison-Wesley.

[Bleistein et al., 2006] Bleistein, S., Cox, K., Verner, J., und Phalp, K. (2006). B-SCP: a requirements analysis framework for validating strategic alignment of organisational IT based on strategy, context and process. *Information and Software Technology*, 48(9):846–868.

[Boehm, 1986] Boehm, B. (1986). A Spiral Model of Software Development and Enhancement. *ACM SIGSOFT Software Engineering Notes*, 11(4):22–42.

[Bontemps et al., 2005] Bontemps, Y., Heymans, P., und Schobbens, P.-Y. (2005). *Scenarios: Models, Transformations and Tools*, chapter Lightweight Formal Methods for Scenario-Based Software Engineering, pages 174–192. Springer Berlin / Heidelberg.

[Booch, 1993] Booch, G. (1993). *Object-oriented analysis and design with applications*. Benjamin-Cummings Publishing Co., Inc. Redwood City, CA, USA.

[Booch et al., 1999] Booch, G., Rumbaugh, J., und Jacobson, I. (1999). *The Unified Modeling Language User Guide*. Addison-Wesley Reading Mass.

[Breu, 2001] Breu, R. (2001). *Objektorientierter Softwareentwurf: Integration Mit UML*. Springer.

[Brinkkemper, 1996] Brinkkemper, S. (1996). Method engineering: engineering of information systems development methods and tools. *Information and Software Technology*, 38(4):275–280.

[Brown, 1996] Brown, P. (1996). The stick-e document: a framework for creating context-aware applications. *Electronic Publishing@article*, 96:259–272.

[Brown et al., 1997] Brown, P., Bovey, J., und Chen, X. (1997). Context-aware applications: from the laboratory to the marketplace. *IEEE Personal Communications*, 4(5):58–64.

[Broy, 2005] Broy, M. (2005). *Engineering Theories of Software Intensive Systems*, chapter Service-oriented Systems Engineering: Specification and Design of Services and Layered Architectures - The JANUS Approach, pages 47 – 81. Springer Verlag.

[Broy, 2006a] Broy, M. (2006a). Challenges in Automotive Software Engineering. In *Proceedings of the 28th International Conference on Software Engineering, ICSE'06*, pages 33–42, New York, NY, USA. ACM.

[Broy, 2006b] Broy, M. (2006b). Requirements Engineering as a Key to Holistic Softwa-

re Quality. In *Proceedings of the 21th International Symposium on Computer and Information Sciences – ISCIS 2006*.

[Broy et al., 2007] Broy, M., Geisberger, E., Kazmeier, J., Rudorfer, A., und Beetz, K. (2007). Ein Requirements-Engineering-Referenzmodell. *Informatik-Spektrum*, 30(3):127–142.

[Broy et al., 2009] Broy, M., Leuxner, C., Sitou, W., Spanfelner, B., und Winter, S. (2009). Formalizing the Notion of Adaptive System Behavior. In *24th Annual ACM Symposium on Applied Computing*, Honolulu, Hawaii, U.S.A. ACM.

[Broy und Stølen, 2001] Broy, M. und Stølen, K. (2001). *Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement*. Springer.

[C2MCA, 2008] C2MCA (2008). Software & Systems Engineering – Competence Center for Mobility & Context Awareness. Available Online at http://www.cawar.de/.

[Chen und Kotz, 2000] Chen, G. und Kotz, D. (2000). A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Dartmouth Computer Science.

[Chen, 1976] Chen, P. P.-S. (1976). The Entity-Relationship Model – Toward a Unified View of Data. *ACM Transactions on Database Systems (TODS)*, 1(1):9–36.

[Chen et al., 2002] Chen, Z., Lin, F., Liu, H., Liu, Y., Ma, W.-Y., und Wenyin, L. (2002). User Intention Modeling in Web Applications Using Data Mining. *World Wide Web*, 5(3):181–191.

[Cheng und Atlee, 2007] Cheng, B. H. C. und Atlee, J. M. (2007). Research Directions in Requirements Engineering. In *FOSE '07: 2007 Future of Software Engineering*, pages 285–303, Washington, DC, USA. IEEE Computer Society.

[Cheverst et al., 2000] Cheverst, K., Davies, N., Mitchell, K., Friday, A., und Efstratiou, C. (2000). Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 17–24. ACM Press.

[Chung et al., 1999] Chung, L., Nixon, B., Yu, E., und Mylopoulos, J. (1999). *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers.

[Coad und Yourdon, 1991a] Coad, P. und Yourdon, E. (1991a). *Object-Oriented Analysis*. Yourdon Press Englewood Cliffs, NJ, USA.

[Coad und Yourdon, 1991b] Coad, P. und Yourdon, E. (1991b). *Object-Oriented Design*. Yourdon Press Englewood Cliffs, NJ, USA.

[Cockburn, 2001] Cockburn, A. (2001). *Writing effective use cases*. Addison-Wesley.

[Coleman et al., 1994] Coleman, D., Arnold, P., Bodoff, S., Dollin, C., Gilchrist, H., Hayes, F., und Jeremaes, P. (1994). *Object-Oriented Development: the Fusion Method*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA.

[Dano et al., 1997] Dano, B., Briand, H., und Barbier, F. (1997). An approach based on the concept of use case to produce dynamicobject-oriented specifications. In *Proceedings of the Third IEEE International Symposium on Requirements Engineering*, pages 54–64.

[Dardenne et al., 1993] Dardenne, A., van Lamsweerde, A., und Fickas, S. (1993). Goal-

directed Requirements Acquisition. *Science of Computer Programming*, 20:3–50.

[Darimont et al., 1997] Darimont, R., Delor, E., Massonet, P., und van Lamsweerde, A. (1997). GRAIL/KAOS: An Environment for Goal-Driven Requirements Engineering. In *19th International Conference on Software Engineering*.

[Darimont und van Lamsweerde, 1996] Darimont, R. und van Lamsweerde, A. (1996). Formal Refinement Patterns for Goal-driven Requirements Elaboration. In *Proceedings of the 4th ACM Symposium on the Foundations of Software Engineering*, pages 179–190. ACM.

[Davies et al., 2005] Davies, N., Landay, J., Hudson, S., und Schmidt, A. (2005). Rapid Prototyping for Ubiquitous Computing. *Pervasive Computing*, 4(4).

[De Angeli et al., 2004] De Angeli, A., Athavankar, U., Joshi, A., Coventry, L., und Johnson, G. (2004). Introducing ATMs in India: a Contextual Inquiry. *Interacting with Computers*, 16(1):29–44.

[DeMarco, 1978] DeMarco, T. (1978). *Structured Analysis and System Specification*. Yourdon Inc., New York.

[Dennett, 1984] Dennett, D. C. (1984). Cognitive Wheels: The Frame Problem of AI. *Minds, Machines, and Evolution*, pages 128–151. Cambridge University Press.

[Dey, 1998] Dey, A. (1998). Context-Aware Computing: The CyberDesk Project. In *Proceedings of AAAI '98 Spring Symposium on Intelligent Environments*, pages 51–54.

[Dey und Abowd, 2000] Dey, A. und Abowd, G. (2000). Towards a Better Understanding of Context and Context-Awareness. In *CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness*.

[Dey et al., 1998] Dey, A., Abowd, G., und Wood, A. (1998). CyberDesk: a framework for providing self-integrating context-aware services. *Knowledge-Based Systems*, 11(1):3–13.

[Dey, 2000] Dey, A. K. (2000). *Providing Architectural Support for Building Context-Aware Applications*. PhD thesis, College of Computing, Georgia Institute of Technology.

[Dey et al., 2001] Dey, A. K., Abowd, G. D., und Salber, D. (2001). A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction*, 16(2, 3 & 4):97–166.

[Dillinger et al., 2004] Dillinger, M., Mohyeldin, E., Luo, J., Fahrmair, M., Dornbusch, P., und Schulz, E. (2004). Cross Layer and End to End Reconfiguration Management. In *WWRF11- Services and Applications Roadmaps- Invigorating the Visions*.

[Dix et al., 2000] Dix, A., Rodden, T., Davies, N., Trevor, J., Friday, A., und Palfreyman, K. (2000). Exploiting space and location as a design framework for interactive mobile systems. *ACM Transactions on Computer-Human Interaction*, 7(3):285–321.

[D'Souza und Wills, 1999] D'Souza, D. F. und Wills, A. C. (1999). *Objects, Components, and Frameworks with UML: the Catalysis Approach*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[Duden, 2001] Duden (2001). *Duden Band 5 - Das Fremdwörterbuch*. Dudenverlag, Mannheim, 7 edition.

[Esprit Project 26900, 1998] Esprit Project 26900 (1998). Technology for Enabling Awa-

reness (TEA). Available Online at http://www.teco.edu/tea/.

[Essing, 2001] Essing, N. (2001). Anwendungen der Benutzermodelle aus der Künstlichen Intelligenz: Repräsentationsmöglichkeiten für Benutzerprofile. Technical report, Institut für Wirtschaftsinformatik, Westfälische Wilhelms-Universität Münster.

[Fahrmair et al., 2008] Fahrmair, M., Leuxner, C., Sitou, W., und Spanfelner, B. (2008). Adaptation Design in Ubiquitous Computing. TR TUM-I0801, Technische Universität München. available at http://www.in.tum.de/forschung/pub/reports/2008/TUM-I0801.pdf.gz.

[Fahrmair et al., 2005] Fahrmair, M., Sitou, W., und Spanfelner, B. (2005). Security and Privacy Rights Management for Mobile and Ubiquitous Computing. In *Privacy in Context – UbiComp Workshop*.

[Fahrmair et al., 2006a] Fahrmair, M., Sitou, W., und Spanfelner, B. (2006a). An Engineering Approach to Adaptation and Calibration. In Roth-Berghofer, T. R., Schulz, S., und Leake, D. B., editors, *Modeling and Retrieval of Context*, volume 3946 of *Springer LNCS*, pages 134 – 147. Springer Berlin / Heidelberg.

[Fahrmair et al., 2006b] Fahrmair, M., Sitou, W., und Spanfelner, B. (2006b). Unwanted Behavior and its Impact on Adaptive Systems in Ubiquitous Computing. In *14th Workshop on Adaptivity and User Modeling in Interactive Systems – LWA/ABIS 2006*.

[Fahrmair et al., 2007] Fahrmair, M., Sitou, W., und Spanfelner, B. (2007). Privacy Management for Context Transponders. In *2007 International Symposium on Applications and the Internet – SAINT SPMS Workshop*, Hiroshima, Japan. IEEE Computer Society.

[Fahrmair, 2005] Fahrmair, M. R. (2005). *Kalibrierbare Kontextadaption für Ubiquitous Computing*. PhD thesis, Department of Informatics, Technische Universität München, Germany.

[Falkenberg, 1976] Falkenberg, E. D. (1976). Concepts for Modelling Information. In *IFIP Working Conference on Modelling in Data Base Management Systems*, pages 95–109, Freudenstadt, Germany. North-Holland.

[Fickas, 2005] Fickas, S. (2005). Clinical Requirements Engineering. In *27th International Conference on Software Engineering*, St. Louis.

[Fickas et al., 2005] Fickas, S., Robinson, W., und Sohlberg, M. (2005). The Role of Deferred Requirements in a Longitudinal Study of Email. In *13th IEEE International Conference on Requirements Engineering*.

[Fischer, 2001] Fischer, G. (2001). User Modeling in Human-Computer Interaction. *User Modeling and User-Adapted Interaction*, 11(1-2):65–86.

[FOLDOC – Compting Dictionary, 2007] FOLDOC – Compting Dictionary (2007). The Free Online Dictionary of Computing. Online. http://foldoc.org.

[Gause und Weinberg, 1989] Gause, D. und Weinberg, G. (1989). *Exploring Requirements: Quality Before Design*. Dorset House Publishing Co., Inc. New York, NY, USA.

[Geihs, 2008] Geihs, K. (2008). Selbst-adaptive Software. *Informatik-Spektrum*, 31(2):133–145.

[Geisberger, 2005] Geisberger, E. (2005). *Requirements Engineering eingebetteter Systeme - ein interdisziplinärer Modellierungsansatz*. PhD thesis, Department of Informatics,

Technische Universität München.

[Geisberger et al., 2006] Geisberger, E., Berenbach, B., Broy, M., Kazmeier, J., Paulish, D., und Rudorfer, A. (2006). Requirements Engineering Reference Model (REM). Technical Report TUM-I0618, Technische Universität München.

[Geisberger und Schätz, 2007] Geisberger, E. und Schätz, B. (2007). Modellbasierte Anforderungsanalyse mit AutoRAID. *Informatik Forschung und Entwicklung*, 21:231–242.

[Gesellschaft für Informatik, 1998] Gesellschaft für Informatik (1998). Arbeitskreis Terminologie der Softwaretechnik der Fachgruppe 2.1.1: Begriffe. Online.

[Goldsby und Cheng, 2006] Goldsby, H. und Cheng, B. H. (2006). Goal-Oriented Modeling of Requirements Engineering for Dynamically Adaptive Systems. In *IEEE International Conference on Requirements Engineering (RE'06)*.

[Gorschek und Wohlin, 2006] Gorschek, T. und Wohlin, C. (2006). Requirements Abstraction Model. *Requirements Engineering*, 11(1):79–101.

[Halpin, 1996] Halpin, T. (1996). Business Rules and Object Role Modeling. *Database Programming & Design*, 9(10):66 – 72.

[Halpin, 1998] Halpin, T. (1998). Object Role Modeling: an Overview. White Paper (Available Online at www.orm.net).

[Halpin, 2001] Halpin, T. (2001). *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*. Morgan Kaufmann.

[Hammond et al., 2001] Hammond, J., Rawlings, R., und Hall, A. (2001). Will It Work? In *Fifth IEEE International Symposium on Requirements Engineering*, pages 102 – 109. IEEE Computer Society.

[Hartson et al., 1990] Hartson, H. R., Siochi, A. C., und Hix, D. (1990). The UAN: A User-Oriented Representation for Direct Manipulation Interface Designs. *ACM Transactions on Information Systems*, 8(3):181–203.

[Henricksen, 2003] Henricksen, K. (2003). *A Framework for Context-Aware Pervasive Computing Applications*. PhD thesis, The University of Queensland – The School of Information Technology and Electrical Engineering.

[Henricksen et al., 2005] Henricksen, K., Indulska, J., und McFadden, T. (2005). Modelling Context Information with ORM. In *OTM Federated Conferences Workshop on Object-Role Modeling (ORM)*, volume 3762 of *LNCS*, pages 626–635. Springer.

[Henricksen et al., 2002] Henricksen, K., Indulska, J., und Rakotonirainy, A. (2002). Modeling Context Information in Pervasive Computing Systems. In *Proceedings of the First International Conference on Pervasive Computing*, pages 167–180. Springer.

[Henricksen et al., 2004] Henricksen, K., Livingstone, S., und Indulska, J. (2004). Towards a hybrid approach to context modelling, reasoning and interoperation. In *Proceedings of the First International Workshop on Advanced Context Modelling, Reasoning And Management, UbiComp'2004*.

[Heumesser und Houdek, 2003] Heumesser, N. und Houdek, F. (2003). Towards Systematic Recycling of Systems Requirements. In *Proceedings of 25th International Conference on Software Engineering – ICSE*, pages 512–519.

[Hong et al., 2005] Hong, D., Chiu, D. K. W., und Shen, V. Y. (2005). Requirements Elicitation for the Design of Context-aware Applications in a Ubiquitous Environment. In *Proceedings of the 7th International Conference on Electronic Commerce*, pages 590–596, New York, NY, USA. ACM.

[Horn und Schubert, 1993] Horn, E. und Schubert, W. (1993). *Objektorientierte Software-Konstruktion*. Hanser.

[Horvitz et al., 1998] Horvitz, E., Breese, J., Heckerman, D., Hovel, D., und Rommelsey, K. (1998). The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In Kaufmann, M., editor, *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 256–265, San Francisco.

[Hourizi und Johnson, 2001] Hourizi, R. und Johnson, P. (2001). Beyond Mode Error: Supporting Strategic Knowledge Structures to Enhance Cockpit Safety. In *Joint Proc. HCI 2001 and ICM 2001*.

[Houssos et al., 2003] Houssos, N., Alonistioti, A., Merakos, L., Mohyeldin, E., illinger, M., Fahrmair, M., und Schoenmakers, M. (2003). Advanced Adaptability and Profile Management Framework for the Support of Flexible Mobile Service Provision. *IEEE Wireless Communications*, 10(4):52–61.

[IEEE, 1990] IEEE (1990). IEEE Standard Glossary of Software Engineering Terminology – IEEE Std 610.12-1990. Technical report, Institute of Electrical and Electronics Engineers, Inc.

[IEEE, 1998] IEEE (1998). IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 830-1998). Technical report, Institute of Electrical and Electronics Engineers, Inc.

[ITU-T, 1996] ITU-T (1996). Recommendation Z.120 – Message Sequence Chart (MSC). http://www.itu.int/ITU-T/studygroups/com10/languages/. (Study Group 10 – 11/99) Letzter Zugriff am 14.10.2007.

[Jackson, 1995] Jackson, M. (1995). The World and the Machine. In *Proceedings of the 17th International Conference on Software Engineering*, pages 283–292, New York, NY, USA. ACM.

[Jackson und Zave, 1993] Jackson, M. und Zave, P. (1993). Domain Descriptions. In *Proceedings of IEEE International Symposium on Requirements Engineering*, pages 56–64.

[Jacobson, 2004] Jacobson, I. (2004). *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison Wesley Longman Publishing Co., Inc. Redwood City, CA, USA.

[Jacobson et al., 1999] Jacobson, I., Booch, G., und Rumbaugh, J. (1999). *The Unified Software Development Process*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

[Jacobson et al., 1992] Jacobson, I., Christenson, M., Jonsson, P., und Övergard, G. (1992). *Object-Oriented Software Engineering – A Use Case Driven Approach*. Addison-Wesley.

[Jameson, 2001] Jameson, A. (2001). Modelling both the Context and the User. *Personal and Ubiquitous Computing*, 5(1):29–33.

[Jameson, 2005] Jameson, A. (2005). User Modeling Meets Usability Goals. In Ardissono, L., Brna, P., und Mitrovic, A., editors, *UM2005, User Modeling: Proceedings of the Tenth International Conference*, pages 1–3. Springer, Berlin. Abstract of an Invited Talk.

[Kaindl, 2000] Kaindl, H. (2000). A Design Process Based on a Model Combining Scenarios with Goals and Functions. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 30(5):537–551.

[Kaiya et al., 2002] Kaiya, H., Horai, H., und Saeki, M. (2002). AGORA: Attributed Goal-Oriented Requirements Analysis Method. In *10th Anniversary IEEE Joint International Conference on Requirements Engineering*, pages 13 – 22.

[Kaiya et al., 2005] Kaiya, H., Shinbara, D., Kawano, J., und Saeki, M. (2005). Improving the detection of requirements discordances among stakeholders. *Requirements Engineering*, 10(4):289–303.

[Kavakli, 2002] Kavakli, E. (2002). Goal-Oriented Requirements Engineering: A Unifying Framework. *Requirements Engineering*, 6(4):237–251.

[Kay, 1994] Kay, J. (1994). The um toolkit for cooperative user modelling. *User Modeling and User-Adapted Interaction*, 4(3):149–196.

[Kemper und Eickler, 1999] Kemper, A. und Eickler, A. (1999). *Datenbanksysteme: Eine Einführung*. R. Oldenbourg Verlag München Wien, 3., korrigierte auflage edition.

[Kephart und Chess, 2003] Kephart, J. und Chess, D. (2003). The Vision of Autonomic Computing. *Computer*, 36(1):41–50.

[Kobsa, 1993] Kobsa, A. (1993). *Adaptive User Interfaces: Principles and Practice*, chapter User Modeling: Recent Work, Prospects and Hazards.

[Kobsa, 1994] Kobsa, A. (1994). User modeling and user-adapted interaction. In *CHI '94: Conference companion on Human factors in computing systems*, pages 415–416, New York, NY, USA. ACM.

[Kolos-Mazuryk et al., 2006] Kolos-Mazuryk, L., van Eck, P. A. T., und Wieringa, R. J. (2006). A Survey of Requirements Engineering Methods for Pervasive Services. Deliverable TI/RS/2006/018, Freeband A-MUSE, Enschede.

[Kotonya und Sommerville, 1998] Kotonya, G. und Sommerville, I. (1998). *Requirements Engineering: Processes and Techniques*. Wiley, John & Sons.

[Krause, 2006] Krause, M. (2006). *Kontextbereitstellung in offenen, ubiquitären Systemen*. PhD thesis, Ludwig-Maximilians-Universität München, Fakultät für Mathematik, Informatik und Statistik.

[Kruchten, 2003] Kruchten, P. (2003). *The Rational Unified Process: An Introduction*. Addison-Wesley Professional.

[Kulak und Guiney, 2000] Kulak, D. und Guiney, E. (2000). *Use Cases – Requirements in Context*. Addison-Wesley.

[Kules, 2000] Kules, B. (2000). User Modeling for Adaptive and Adaptable Software Systems. In *ACM Conference on Universal Usability*.

[Langley, 1999] Langley, P. (1999). User Modeling in Adaptive Interfaces. In *Proceedings of the Seventh International Conference on User Modeling*, pages 357–370.

[Lemlouma und Layaida, 2004] Lemlouma, T. und Layaida, N. (2004). Context-aware adaptation for mobile devices. In *IEEE International Conference on Mobile Data Management*, pages 106–111.

[Letier, 2001] Letier, E. (2001). *Reasoning about Agents in Goal-Oriented Requirements Engieering*. PhD thesis, Université Catholique de Louvain.

[Letier und van Lamsweerde, 2002] Letier, E. und van Lamsweerde, A. (2002). Deriving Operational Software Specifications from System Goals. *ACM SIGSOFT Software Engineering Notes*, 27(6):119–128.

[Leuxner, 2006] Leuxner, C. (2006). Adaptation Design in Ubiquitous Computing. Diplomarbeit.

[Liebermann, 2005] Liebermann, G. (2005). *Contemporary Challenges in Information Technology - Results of the MSc- Seminars in SS2004 and WS 2004/05*, chapter The Role of Task-Modeling in Human-Computer Interaction, pages 150 – 160. Fachhochschule Augsburg, University of Applied Sciences.

[Limbourg et al., 2001] Limbourg, Q., Pribeanu, C., und Vanderdonckt, J. (2001). Towards Uniformed Task Models in a Model-Based Approach. In *Proceedings of 8th International Workshop in Interactive Systems: Design, Specification, and Verification, DSV-IS 2001*, pages 164–182. Springer.

[Loucopoulos und Karakostas, 1995] Loucopoulos, P. und Karakostas, V. (1995). *System Requirements Engineering*. McGraw-Hill, Inc., New York, NY, USA.

[Luyten et al., 2003] Luyten, K., Clerckx, T., Coninx, K., und Vanderdonckt, J. (2003). Derivation of a Dialog Model from a Task Model by Activity Chain Extraction. *Interactive Systems: Design, Specification, and Verification*, 262:269.

[McMenamin und Palmer, 1984] McMenamin, S. und Palmer, J. (1984). *Essential systems analysis*. Yourdon Press Upper Saddle River, NJ, USA.

[Merriam-Webster Inc., 2007] Merriam-Webster Inc. (2007). Merriam-Webster Online Dictionary. Online. http://www.webster.com.

[Mitchell et al., 1997] Mitchell, R. K., Agle, B. R., und Wood, D. J. (1997). Toward a Theory of Stakeholder Identification and Salience: Defining the Principle of Who and What Really Counts. *Academy of Management Review*, 22:853–886.

[Mohyeldin et al., 2004a] Mohyeldin, E., Dillinger, M., Fahrmair, M., Dornbusch, P., und Sitou, W. (2004a). Interworking between Link Layer and Application Layer Adaptations in a Reconfigurable Wireless Middleware. In *15th IEEE Intl. Symposium on Personal, Indoor and Mobile Communications – PIMRC*.

[Mohyeldin et al., 2004b] Mohyeldin, E., Dillinger, M., Fahrmair, M., Sitou, W., und Dornbusch, P. (2004b). A Generic Framework for Negotiations and Trading in Context Aware Radio. In *Software Defined Radio Technical Conference – SDR-TC*.

[Mori et al., 2002] Mori, G., Paternò, F., und Santoro, C. (2002). CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. *IEEE Transactions on Software Engineering*, 28(8):797–813.

[Mylopoulos et al., 1999] Mylopoulos, J., Chung, L., und Yu, E. (1999). From object-oriented to goal-oriented Requirements Analysis. *Communications of the ACM*,

42(1):31–37.

[Nicklas, 2005] Nicklas, D. (2005). *Ein umfassendes Umgebungsmodell als Integrationsstrategie für ortsbezogene Daten und Dienste*. PhD thesis, Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart.

[Norman, 2004] Norman, D. A. (2004). *Emotional Design: Why We Love (or Hate) Everyday Things*. Basic Books.

[Nuseibeh, 2001] Nuseibeh, B. (2001). Weaving Together Requirements and Architectures. *Computer*, 34(3):115–117.

[Nuseibeh und Easterbrook, 2000] Nuseibeh, B. und Easterbrook, S. (2000). Requirements Engineering: a Roadmap. In *ICSE '00: Conf. on The Future of Software Engineering*, pages 35–46. ACM Press.

[OMG UML Partners, 2005] OMG UML Partners (2005). Unified Modeling Language Specification. Available Online at http://www.omg.org/docs/formal/05-04-01.pdf. This specification is also available from ISO as ISO/IEC 19501.

[Palmer, 1995] Palmer, E. (1995). Oops, it didn't arm'- A Case Study of two Automation Surprises. In *8th International Symposium on Aviation Psychology, Columbus, OH, United States*, pages 227–232.

[Paradis et al., 2003] Paradis, F., Crimmins, F., und Ozkan, N. (2003). A Task Oriented Approach to Delivery in Mobile Environments. In *Proceedings of the 4th International Conference on Mobile Data Management*, pages 386–390. Springer.

[Pascoe, 1998] Pascoe, J. (1998). Adding Generic Contextual Capabilities to Wearable Computers. In *Proceedings of the 2nd International Symposium on Wearable Computers*, pages 92–99.

[Pascoe et al., 1999] Pascoe, J., Ryan, N., und Morse, D. (1999). Issues in Developing Context-Aware Computing. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, pages 208–221. Springer.

[Paternò, 1999] Paternò, F. (1999). *Model-Based Design and Evaluation of Interactive Applications*. Springer-Verlag, London, UK.

[Paternò et al., 1997] Paternò, F., Mancini, C., und Meniconi, S. (1997). ConcurTaskTree: A Diagrammatic Notation for Specifying Task Models. In *International Conference on Human-Computer Interaction – Interact'97*.

[Paternò et al., 2001] Paternò, F., Mori, G., und Galiberti, R. (2001). CTTE: an Environment for Analysis and Development of Task Models of Cooperative Applications. In *Conference on Human Factors in Computing Systems*, pages 21–22. ACM Press New York, NY, USA.

[Pham, 2001] Pham, T.-L. (2001). *Composite Device Computing Environment – A Context-Aware Augmentation of Handheld Devices by Surrounding Resources*. PhD thesis, Technische Universität München.

[Plihon et al., 1998] Plihon, V., Ralyté, J., Benjamen, A., Maiden, N., Sutcliffe, A., Dubois, E., und Heymans, P. (1998). A Reuse-Oriented Approach for the Construction of Scenario Based Methods. In *Proceedings of Third International Conference on Software Process (ICSP'98)*.

[Pohl und Haumer, 1997] Pohl, K. und Haumer, P. (1997). Modelling Contextual Information about Scenarios. *Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ*, 97:187–204.

[Pohl und Sikora, 2007] Pohl, K. und Sikora, E. (2007). COSMOD-RE: Supporting the Co-Design of Requirements and Architectural Artifacts. In *15th IEEE International Requirements Engineering Conference, RE'07*, pages 258–261.

[Pomberger und Blaschek, 1996] Pomberger, G. und Blaschek, G. (1996). *Software Engineering: Prototyping und objektorientierte Software-Entwicklung*. Carl Hanser.

[Potts, 1995] Potts, C. (1995). Using Schematic Scenarios to Understand User Needs. In *Proceedings of the Conference on Designing Interactive Systems: Processes, Practices, Methods, & Techniques*, pages 247–256. ACM Press New York, NY, USA.

[Potts, 1999] Potts, C. (1999). ScenIC: a Strategy for Inquiry-Driven Requirements Determination. In *Proceedings of IEEE International Symposium on Requirements Engineering*.

[Potts et al., 1994] Potts, C., Takahashi, K., und Anton, A. (1994). Inquiry-based Requirements Analysis. *Software, IEEE*, 11(2):21–32.

[Priyantha et al., 2001] Priyantha, N. B., Miu, A. K., Balakrishnan, H., und Teller, S. (2001). The Cricket Compass for Context-Aware Mobile Applications. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 1–14, New York, NY, USA. ACM.

[Pulvermueller et al., 2002] Pulvermueller, E., Speck, A., Coplien, J. O., D'Hondt, M., und DeMeuter, W. (2002). Feature Interaction in Composed Systems. *LNCS*, 2323:86–97.

[Rechenberg et al., 2002] Rechenberg, P., Pomberger, G., et al. (2002). *Informatik-Handbuch*. Hanser.

[Regnell et al., 1996] Regnell, B., Andersson, M., und Bergstrand, J. (1996). A Hierarchical Use Case Model with Graphical Representation. In *Proceedings of ECBS'96, IEEE Second Symposium and Workshop on Engineering of Computer-Based Systems*, pages 270–277.

[Rieback et al., 2005] Rieback, M. R., Crispo, B., und Tanenbaum, A. (2005). RFID Guardian: A battery-powered mobile device for RFID privacy management. In *Proceedings of the 10th Australasian Conference on Information Security and Privacy – ACISP*, volume 3574, pages 184–194. Springer.

[Robertson und Robertson, 2007] Robertson, J. und Robertson, S. (2007). Volere Requirements Specification Templates. Available Online at http://www.volere.co.uk. Edition 13.

[Robertson und Robertson, 1999] Robertson, S. und Robertson, J. (1999). *Mastering the Requirements Process*. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA.

[Robinson und Pawlowski, 1999] Robinson, W. N. und Pawlowski, S. D. (1999). Managing Requirements Inconsistency with Development Goal Monitors. *IEEE Transactions on Software Engineering*, 25(6):816–835.

[Rodden et al., 1998] Rodden, T., Cheverst, K., Davies, N., und Dix, A. (1998). Exploiting Context in HCI design for Mobile Systems. In *Workshop on Human Computer Interaction with Mobile Devices*.

[Rolland et al., 1999] Rolland, C., Grosz, G., und Kla, R. (1999). Experience with Goal-Scenario Coupling in Requirements Engineering. In *Proceedings of the Fourth IEEE International Symposium on Requirements Engineering (RE'99)*.

[Rolland et al., 1998] Rolland, C., Souveyet, C., und Achour, C. (1998). Guiding Goal Modeling Using Scenarios. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 24(12):1055 – 1071.

[Royce, 1970] Royce, W. (1970). Managing the Development of Large Software Systems. In *Proceedings of IEEE WESCON*, pages 1–9.

[Rumbaugh et al., 1991] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., und Lorensen, W. (1991). *Object-Oriented Modeling and Design*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA.

[Rumbaugh et al., 2004] Rumbaugh, J., Jacobson, I., und Booch, G. (2004). *The Unified Modeling Language Reference Manual, Second Edition*. Addison-Wesley Object Technology.

[Russell und Norvig, 2003] Russell, S. und Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Pearson Education, Upper Saddle River, New Jersey, USA.

[Ryan et al., 1997] Ryan, N., Pascoe, J., und Morse, D. (1997). Enhanced Reality Fieldwork: the Context Aware Archaeological Assistant. In Gaffney, V., van Leusen, M., und Exxon, S., editors, *Computer Applications in Archaeology*.

[Samulowitz, 2002] Samulowitz, M. (2002). *Kontextadaptive Dienstnutzung in Ubiquitous Computing Umgebungen*. PhD thesis, Ludwig-Maximilians-Universität München, Germany.

[Sarter et al., 1997] Sarter, N., Woods, D., und Billings, C. (1997). Automation surprises. *Handbook of Human Factors and Ergonomics*, 2:1926–1943.

[Schilit et al., 1994] Schilit, B. N., Adams, N., und Want, R. (1994). Context-Aware Computing Applications. In *IEEE Workshop on Mobile Computing Systems and Applications*.

[Schilit und Theimer, 1994] Schilit, B. N. und Theimer, M. M. (1994). Disseminating active map information to mobile hosts. *IEEE Network*, 8(5):22–32.

[Schilit et al., 1993] Schilit, B. N., Theimer, M. M., und Welch, B. B. (1993). Customizing Mobile Applications. In *Proceedings of USENIX Symposium on Mobile & Location-Indendent Computing*, pages 129–138.

[Schilit, 1995] Schilit, W. N. (1995). *A System Architecture for Context-Aware Mobile Computing*. PhD thesis, Columbia University.

[Schmidt, 2002] Schmidt, A. (2002). *Ubiquitous Computing - Computing in Context*. PhD thesis, Computing Department, Lancaster University, England, U.K.

[Schmidt et al., 1999a] Schmidt, A., Aidoo, K. A., Takaluoma, A., Tuomela, U., Van Laerhoven, K., und Van de Velde, W. (1999a). Advanced Interaction in Context. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Compu-*

ting, pages 89–101. Springer-Verlag London, UK.

[Schmidt et al., 1999b] Schmidt, A., Beigl, M., und Gellersen, H. (1999b). There is more to context than Location. *Computers & Graphics*, 23(6):893–901.

[Schmidt-Belz, 2005] Schmidt-Belz, B. (2005). User trust in Adaptive Systems. In *13th Workshop on Adaptivity and User Modeling in Interactive Systems – LWA/ABIS 2005*.

[Seyff et al., 2008] Seyff, N., Graf, F., Grünbacher, P., und Maiden, N. (2008). Mobile Discovery of Requirements for Context-Aware Systems. In *14th International Working Conference on Requirements Engineering: Foundation for Software Quality, REFSQ*, pages 183 – 197, Montpellier, France.

[Sharp et al., 1999] Sharp, H., Finkelstein, A., und Galal, G. (1999). Stakeholder identification in the requirements engineering process. In *Proceedings of Tenth International Workshop on Database and Expert Systems Applications*, pages 387–391.

[Shen und Shen, 2001] Shen, J. und Shen, X.-J. (2001). User Requirements in Mobile Systems. In *Proceedings of the 2001 Americas Conference on Information Systems*, pages 1341–1344, Boston.

[Simon, 1969] Simon, H. A. (1969). *The Sciences of the Artificial*. MIT Press.

[Sitou und Spanfelner, 2007] Sitou, W. und Spanfelner, B. (2007). Towards Requirements Engineering for Context Adaptive Systems. In *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, volume 2, pages 593–600, Beijing, China. IEEE Computer Society.

[Somé et al., 1995] Somé, S., Dssouli, R., und Vaucher, J. (1995). Toward an Automation of Requirements Engineering using Scenarios. *Journal of Computing and Information*, 2:1110–1132.

[Sommerville und Sawyer, 1997] Sommerville, I. und Sawyer, P. (1997). *Requirements Engineering - A good practice guide*. John Wiley & Sons.

[Spanoudakis und Finkelstein, 1998] Spanoudakis, G. und Finkelstein, A. (1998). A Semi-automatic Process of Identifying Overlaps and Inconsistencies between Requirement Specifications. In *In Proceedings of the 5th International Conference on Object-Oriented Information Systems (OOIS 98*, pages 405–424.

[Strang und Linnhoff-Popien, 2004] Strang, T. und Linnhoff-Popien, C. (2004). A Context Modeling Survey. In *Workshop on Advanced Context Modelling, Reasoning and Management associated with the Sixth International Conference on Ubiquitous Computing (UbiComp 2004)*.

[Sutcliffe, 1996] Sutcliffe, A. (1996). A Conceptual Framework for Requirements Engineering. *Requirements Engineering*, 1(3):170–189.

[Sutcliffe, 1997] Sutcliffe, A. (1997). A Technique Combination Approach to Requirements Engineering. In *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE'97)*, pages 65 – 74. IEEE Computer Society Washington, DC, USA.

[Sutcliffe et al., 2005] Sutcliffe, A., Fickas, S., und Sohlberg, M. (2005). Personal and Contextual Requirements Engineering. In *13th IEEE Intl. Conf. on Requirements Engineering*, Paris.

[Sutcliffe et al., 2006] Sutcliffe, A., Fickas, S., und Sohlberg, M. M. (2006). PC-RE: a method for personal and contextual requirements engineering with some experience. *Requirements Engineering*, 11(No. 4):157–173.

[Sutcliffe et al., 1998] Sutcliffe, A. G., Maiden, N. A. M., Minocha, S., und Manuel, D. (1998). Supporting scenario-based requirements engineering. *IEEE Transactions on Software Engineering*, 24(12):1072–1088.

[Tarasewich, 2003] Tarasewich, P. (2003). Towards a Comprehensive Model of Context for Mobile and Wireless Computing. In *Americas Conference on Information Systems - AMCIS*.

[Thalheim, 2000] Thalheim, B. (2000). *Entity-Relationship Modeling: Foundations of Database Technology*. Springer.

[Timpf, 2008] Timpf, S. (2008). Location-based Services - Personalisierung mobiler Dienste durch Verortung. *Informatik-Spektrum*, 31(1):70 – 74.

[Trapp, 2005] Trapp, M. (2005). *Modeling the Adaptation Behavior of Adaptive Embedded Systems*. PhD thesis, PhD thesis, University of Kaiserslautern.

[Trapp und Schürmann, 2003] Trapp, M. und Schürmann, B. (2003). On the modeling of adaptive systems. In *International Workshop on Dependable Embedded Systems*, Florence, Italy.

[Tuunanen, 2005] Tuunanen, T. (2005). *Requirements Elicitation for Wide Audience End-users*. PhD thesis, Helsinki School of Economics.

[Uwineza Gatabazi, 2008] Uwineza Gatabazi, M. (2008). Kontextmodellierung für adaptive Systeme. Diplomarbeit.

[van Laerhoven und Aidoo, 2001] van Laerhoven, K. und Aidoo, K. (2001). Teaching Context to Applications. *Personal and Ubiquitous Computing*, 5(1):46–49.

[van Lamsweerde, 2001] van Lamsweerde, A. (2001). Goal-Oriented Requirements Engineering: A Guided Tour. In *5th IEEE Intl. Symposium on Requirements Engineering*, page 249. IEEE Computer Society.

[van Lamsweerde et al., 1998] van Lamsweerde, A., Letier, E., und Darimont, R. (1998). Managing Conflicts in Goal-Driven Requirements Engineering. *IEEE Transactions on Software Engineering*, 24(11):908–926.

[van Welie, 2001] van Welie, M. (2001). *Task Based User Interface Design*. PhD thesis, Vrije Universiteit Amsterdam.

[van Welie et al., 1998] van Welie, M., van der Veer, G. C., und Eliëns, A. (1998). An Ontology for Task World Models. *Design, Specification and Verification of Interactive System*, 98.

[Wahlster und Kobsa, 1989] Wahlster, W. und Kobsa, A. (1989). User Models in Dialog Systems. *User Models in Dialog Systems*, pages 4–34.

[Want et al., 1992] Want, R., Hopper, A., Falcao, V., und Gibbons, J. (1992). The Active Badge Location System. *ACM Transactions on Information Systems*, 10:91–102.

[Weiser, 1991] Weiser, M. (1991). The Computer for the 21st Century. *Scientific American*, 3(265):94–104.

[Welsh und Sawyer, 2008] Welsh, K. und Sawyer, P. (2008). When to Adapt? Identification of Problem Domains for Adaptive Systems. In *14th International Working Conference on Requirements Engineering: Foundation for Software Quality, REFSQ*, pages 198 – 203, Montpellier, France.

[Wiegers, 1999] Wiegers, K. E. (1999). *Software Requirements*. Microsoft Press.

[Wikimedia Foundation, 2007] Wikimedia Foundation (2007). Wikipedia: Die freie Enzyclopädie. Online. http://de.wikipedia.org/wiki/Hauptseite.

[XEROX, 1999] XEROX, P. A. R. C. (1999). Mark D. Weiser – July 23, 1952 - April 27, 1999. http://www2.parc.com/csl/members/weiser/mwbkgd.htm.

[Yourdon, 1989] Yourdon, E. (1989). *Modern Structured Analysis*. Yourdon Press Upper Saddle River, NJ, USA.

[Yu, 1997] Yu, E. S. K. (1997). Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In *3rd IEEE Intl. Symposium on in Requirements Engineering*, pages 226–235.

[Yue, 1987] Yue, K. (1987). What Does It Mean to Say that a Specification is Complete? In *4th Intl. Workshop on Software Specification and Design*, Monterey.

[Zave, 1993] Zave, P. (1993). Feature Interactions and Formal Specifications in Telecommunications. *Computer*, 26(8):20–29.

[Zave und Jackson, 2000] Zave, P. und Jackson, M. (2000). New Feature Interactions in Mobile and Multimedia Telecommunications Services. In Calder, M. und Magill, E. H., editors, *Feature Interactions in Telecommunications and Software Systems VI*, pages 51–66. IOS Press.

[Ziegler et al., 2005] Ziegler, J., Lohmann, S., und Kaltz, J. W. (2005). Kontextmodellierung für adaptive webbasierte Systeme. *Mensch & Computer*, pages 181–189.