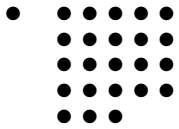

Fachhochschule Köln
Cologne University of Applied Sciences

Fakultät für Informatik und Ingenieurwissenschaften
Studiengang Medieninformatik

Masterarbeit
Master of Science in Media Informatics

Modellierung und Repräsentation des Kontextes von mobilen Nutzungsszenarien – ein Rahmenwerk für mobile kontextsensitive Applikationen

Context modeling and context retrieval in mobile usage scenarios – a framework for context-aware mobile applications

ausgearbeitet von:

Felix Müller
felix.mueller@mac.com

vorgelegt am:

25. Februar 2010

erster Prüfer:

Prof. Dr. Kristian Fischer

zweiter Prüfer:

Prof. Dr. Gerhard Plaßmann

„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 applications themselves.“

– Dey & Abowd, 2000 –

Zusammenfassung

Die fortschreitende Verbreitung drahtloser Kommunikationsnetze sowie immer leistungsfähigerer mobiler Computer schafft ein großes Potenzial für ein breites Spektrum innovativer Anwendungen. Kontextsensitive Applikationen adaptieren die Gegebenheiten der jeweiligen Situation des Nutzers, wodurch neuartige, intelligente Anwendungen und Benutzungsschnittstellen möglich werden. Die zunehmende Menge verfügbarer Sensortechniken und die daraus resultierende Vielfalt erfassbarer Kontextinformationen erschweren jedoch vermehrt die Verbreitung dieser Applikationen.

Ein Rahmenwerk zur Kontextrepräsentation soll die Entwicklung kontextsensitiver Applikationen ohne Berücksichtigung von Details der Kontexterfassung und -verwaltung ermöglichen. Außerdem sollen Austausch und Wiederverwendbarkeit von Kontextinformationen zwischen Applikationen und Nutzern gestattet werden. In dieser Arbeit wird ein solches Rahmenwerk entwickelt. Zu Beginn steht die Untersuchung von Fragen der Kontextmodellierung, auf deren Grundlage anschließend eine Konzeption erarbeitet wird. Der praktische Teil der Arbeit führt eine Referenzimplementation des Systems durch um zu evaluieren, ob die Konzeption auf Basis aktueller Technologien in die Realität umgesetzt, und als Grundlage für weitere Untersuchungen herangezogen werden kann.

Abstract

The progressive spreading of wireless networks and increasingly powerful mobile computers creates a big potential for a wide spectrum of innovative applications. Context-aware applications adapt the circumstances of the user's current situation which enables new and intelligent user interfaces. Nevertheless, the sheer diversity of exploitable contexts and the plethora of sensing technologies are actually working against the deployment of context-aware systems.

A framework for context retrieval should enable the development of context-aware applications without considering details of context acquisition and context management. Moreover, exchange and reusability of context information should be allowed between applications and users. In this work such a framework is developed. At the beginning, an investigation of questions of context modeling is accomplished. After that, a concept is compiled on that foundation. In the practial part of this thesis a reference implementation of the system is conducted to evaluate whether the concept can be put into practise, and be pulled up as a basis for other investigations.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
ASC	Aspect Scale ContextInformation
B2B	Business-to-Business
B2C	Business-to-Consumer
CC/PP	Composite Capability/Preference Profiles
CIS	Contextual Information Service
CoBrA	Context Broker Architecture
CONON	Context Ontology
CSCP	Comprehensive Structured Context Profiles
DAML	Darpa Agent Markup Language
GPS	Global Positioning System
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
Haml	HTML Abstraction Markup Language
HTTP	Hypertext Transfer Protocol
ISO	International Organization for Standardization
JCAF	Java Context Awareness Framework
JSON	JavaScript Object Notation
MVC	Model View Controller
OIL	Ontology Inference Layer
ORM	Object Role Modeling
OWL	Web Ontology Language
QoC	Quality of Context
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
REST	Representational State Transfer
RFID	Radio Frequency Identification
SGML	Standard Generic Markup Language
SPARQL	SPARQL Protocol and RDF Query Language
SSID	Service Set Identifier
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	Extensible Markup Language

1. Einleitung

Dieses Kapitel stellt die ersten Gedanken zum Thema vor. Die Motivation, die zur Anfertigung dieser Arbeit geführt hat, wird anhand aktueller Entwicklungen erläutert. Es folgen die Definition der Ziele der Arbeit, die Eingrenzung der behandelten Themen sowie die Beschreibung der angewandten Vorgehensweise. Schlussendlich wird der strukturelle Aufbau der Arbeit umrissen.

1.1. Motivation

Computer durchdringen zunehmend den menschlichen Alltag. Die entwickeln sich in die Richtung der Allgegenwärtigkeit, in der sie jederzeit und überall verfügbar werden. Die fortschreitende Entwicklung und Verbreitung drahtloser Kommunikationsnetze sowie immer leistungsfähigerer mobiler Computer schafft ein großes Potenzial für ein breites Spektrum innovativer Anwendungen. Sie unterstützen uns in den unterschiedlichsten Lebenssituationen, indem sie Dienste anbieten und auffinden, Informationen liefern und dadurch unsere Bedürfnisse befriedigen.

Schlüsselfunktionalität der Integration dieser Anwendungen in die jeweilige Situation des Benutzers und ihre Autonomität ist das so genannte Kontextbewusstsein (engl. *Context-Awareness*). Benutzer möchten Informationen abhängig von Ort, Zeit, verfügbarer Ressourcen, Umgebung usw. nutzen und bearbeiten. Des Weiteren sollen Applikationen die Charakteristika der physikalischen Umgebung berücksichtigen, um sich der Situation anpassen zu können. Informationen von Sensoren, Netzwerken, Gerätestatus, Benutzerprofilen und anderen Quellen können die Gebrauchstauglichkeit mobiler Applikationen erhöhen, indem sich diese den Gegebenheiten anpassen, die einen Einfluss auf ihre Ausführung haben. Durch dieses adaptive Verhalten werden neuartige intelligente Anwendungen und Benutzungsschnittstellen möglich.

Durch die hohe Verfügbarkeit multifunktionaler mobiler Endgeräte mit einer Vielzahl von Sensoren ist in vergangener Zeit eine Zunahme kontextsensitiver mobiler Applikationen zu beobachten. Das neu aufkommende Paradigma der überall verfügbaren Dienste, die jederzeit, an jedem Ort und an jedes Endgerät ausgeliefert werden, fordert jedoch neue Modelle und Mechanismen zur Gestaltung und Auslieferung dieser kontextbezogenen Dienste. Mit diesen neuen Herausforderungen beschäftigen sich viele wissenschaftliche Arbeiten der letzten Jahre.

Um Interoperabilität zwischen Kontextquellen und den Nutzern dieser Quellen, also den Anwendungen, zu gewährleisten, sowie einen Informationsaustausch zwi-

schen mobilen Teilnehmern zu ermöglichen, entwickelten zahlreiche Autoren Infrastrukturen in Form von Rahmenwerken oder Middleware-Systemen. Diese verfolgen oft unterschiedliche Ziele, ihnen gemein ist jedoch das Anstreben eines effizienten Austauschs von Kontextinformationen in einer dynamischen Umgebung sowie die Abbildbarkeit komplexer Kontextinformationen höherer Ebenen. Ein weiteres Ziel ist die Gewährleistung der Qualität der Kontextinformationen.

1.2. Ziele der Arbeit

Diese Masterarbeit beschäftigt sich mit der Fragestellung, ob und inwieweit sich ein generisches, anwendungsunabhängiges und erweiterbares Rahmenwerk entwickeln lässt, welches Kontextinformationen repräsentiert und verwaltet, um mobilen Applikationen eine möglichst einfache Kontextadaption zu ermöglichen.

Unter generisch ist in diesem Zusammenhang die Abstraktion des Kontextmodells von konkreten Kontextinformationen wie beispielsweise Ort, Zeit, Geschwindigkeit, Temperatur usw. zu verstehen. Die gemeinsamen Eigenschaften der verschiedenen Informationstypen sind also relevant und nicht deren Details. Des Weiteren soll das System unabhängig von den mobilen Applikationen sein, die es verwenden und eine individuelle Adaption des Kontextes, je nach angestrebtem Nutzen, der Applikation selbst überlassen.

Erweiterbarkeit meint die Unbeschränktheit im Hinblick auf bestimmte Typen von Kontextinformationen und die Quellen dieser Informationen. Das Kontextmodell, welches die Kontexte abbildet, soll verschiedenartige Kontextinformationen unterstützen und sich nachträglich um neue Kontexttypen sowie weitere Kontextquellen erweitern lassen.

Ein Rahmenwerk stellt den Rahmen, innerhalb dessen der Entwickler eine Anwendung erstellt, zur Verfügung. Damit ist in diesem Zusammenhang eine gemeinsame Basis gemeint, auf die verschiedene mobile Applikationen zurückgreifen, um Kontextinformationen zu verwalten. Die Begründung besteht zum einen in der Reduktion von Komplexität durch die Übernahme von Aufgaben der Kontextverarbeitung und die Trennung von Kontextquellen, Kontextmodell und Kontextverwaltung sowie zum anderen in der Effizienz durch eine Aggregation von Kontextinformationen aus verschiedenen Quellen.

Eine weitere Zielsetzung ist der Fokus auf aktuelle Technologien und Standards im Bereich mobiler Geräte sowie Infrastrukturen und der Modellierung.

1.3. Eingrenzung des Themas

Diese Arbeit befasst sich mit Fragen der Modellierung und Repräsentation des Kontexes mobiler Nutzungsszenarien sowie mit Systemen zur Unterstützung der Kontextverwaltung für mobile Applikationen. Dabei soll es nicht um konkrete Kontext-

variablen und Kontextquellen gehen, ebenso nicht um konkrete Anwendungsfälle zur Kontextadaption. Diese Themen werden ausschließlich im Grundlagenkapitel sowie im Rahmen der Referenzimplementierung behandelt um einen Überblick über die Problemstellung zu verschaffen.

Kontextinformationen stellen oft sensible, personenbezogene Daten dar. Systeme die solche Informationen verwalten, insbesondere verteilte Systeme, müssen die Schutzziele der Vertraulichkeit und Integrität daher angemessen berücksichtigen. Da es in dieser Arbeit jedoch primär um die Realisierbarkeit eines Rahmenwerks zur Kontextrepräsentation geht, werden die Themenbereiche Sicherheit und Datenschutz nur konzeptionell behandelt und im Rahmen der Umsetzung vernachlässigt.

1.4. Vorgehensweise

Um sich den Antworten auf die in Abschnitt 1.2 formulierte Forschungsfrage anzunähern, erfolgt die Auseinandersetzung mit der Thematik in mehreren Schritten.

Zunächst werden die Möglichkeiten der Modellierung und Repräsentation des Kontextes mobiler Nutzungsszenarien betrachtet. Es geht also um die Art und Weise, wie der mobile Kontext in der Software durch ein geeignetes Modell abgebildet werden kann. In der Literatur werden zahlreiche Ansätze beschrieben, die in dieser Arbeit untersucht und bewertet werden.

Anschließend werden Systeme zum Thema gemacht, die mobilen Applikationen Kontextinformationen bereitstellen, um ihnen eine Adaption des Kontextes zu erleichtern. Die Anforderungen an solche Systeme werden auf Basis der Grundlagenarbeit zusammengetragen. Aus den gewonnenen Erkenntnissen wird sodann das Konzept eines generischen Rahmenwerks ausgearbeitet.

Dieses Konzept wird im Rahmen einer Referenzimplementation prototypisch umgesetzt. Der praktische Teil soll die Frage nach der Realisierbarkeit beantworten. Die Ergebnisse sowie die Schlussfolgerungen daraus werden letztendlich am Schluss dieser Arbeit dargelegt.

1.5. Aufbau der Arbeit

Nachdem dieses Kapitel 1 die ersten Gedanken zum Thema vorgestellt hat, befasst sich Kapitel 2 mit den Grundlagen des Themenkomplexes. In diesem Kapitel werden wichtige Begrifflichkeiten zum Thema definiert, die zum weiteren Verständnis der Arbeit maßgeblich sind. Dazu gehören die Formen der mobilen Nutzung von Computern, die verschiedenen Kontextbegriffe sowie die Möglichkeiten der Adaption des Kontextes. Außerdem wird auf die Qualität von Kontextinformationen sowie auf Möglichkeiten der Struktierung von Kontextvariablen eingegangen.

Das nachfolgende Kapitel 3 befasst sich mit den Einsatzgebieten und Kategorien kontextsensitiver Anwendungen, die anhand einiger Beispiele aufgezeigt werden.

Eine Erörterung der Herausforderungen und der Gestaltungsziele, die bei der Entwicklung kontextsensitiver Applikationen eine Rolle spielen, leitet zur Fragestellung der Modellierung des Kontextes über.

Kapitel 4 behandelt Fragen der Modellierung und Repräsentation des Kontextes. Zunächst werden das Konzept der Kontextmodelle erläutert und etablierte Ansätze zur Kontextmodellierung aus der Literatur beschrieben. Die Modellierungsansätze werden anschließend im Hinblick auf die Zielsetzung dieser Arbeit bewertet. Besondere Aufmerksamkeit wird dabei auf die semantische Kontextmodellierung gelegt.

Das Kapitel 5 befasst sich mit der Frage nach einem Rahmenwerk zur Repräsentation und Verwaltung von Kontextinformationen. Nachdem eine Darlegung der Beweggründe für die Entwicklung eines solchen Systems erfolgt ist, wird auf verschiedene Architekturkonzepte eingegangen. Anschließend werden aus den zuvor gesammelten Erkenntnissen Anforderung an das System zusammengestellt. Bevor eine Konzeption eines generischen Rahmenwerks durchgeführt wird, werden bestehende Rahmenwerke und Systeme aus der Literatur untersucht. Die eigentliche Konzeption des Systems beinhaltet schließlich eine Beschreibung der geplanten Systemarchitektur sowie der Schnittstellen zwischen den einzelnen Systemkomponenten.

Den Praxisteil der Masterarbeit bildet das Kapitel 6, welches die prototypische Realisierung des Rahmenwerks durch eine Referenzimplementierung dokumentiert. In diesem Kapitel werden zunächst die Zielsetzung der Umsetzung und die verwendeten Werkzeuge und Plattformen dargelegt. Anschließend erfolgt eine detaillierte Dokumentation der Implementierung der einzelnen Module.

Kapitel 7 stellt den Schlussteil der Masterarbeit dar. Hier werden die gewonnenen Erkenntnisse wiedergegeben und die daraus resultierenden Schlussfolgerungen dargelegt. Darüber hinaus wird ein Ausblick auf weitere Schritte sowie mögliche Anknüpfungspunkte für weiterführende Arbeiten gegeben.

2. Grundlagen

Nachdem in der Einleitung die ersten Gedanken zum Thema vorgestellt wurden, wird nun eine Basis an Begrifflichkeiten und Grundlagen geschaffen, die für die weiteren Kapitel dieser Arbeit fundamental sind.

Da sich diese Arbeit mit dem Kontext mobiler Nutzungsszenarien beschäftigt, wird zunächst auf die verschiedenen Formen der mobilen Nutzung von Computern eingegangen. Anschließend folgen Definitionen der Kontextbegriffe, wobei vom allgemeinen Kontextbegriff auf den Kontext mobiler Nutzungsszenarien übergeleitet wird. Eine Definition des Kontextbewusstseins schließt den Definitionsteil.

Der nächste Teil des Kapitels befasst sich mit der Adaption des Kontextes. Auch hier existieren verschiedene Begriffe, die zunächst voneinander abgegrenzt werden. Anschließend wird der Begriff der Kontextqualität (engl. *Quality of Context*) aufgegriffen.

Schlussendlich werden verschiedene Kontextvariablen und deren Klassen sowie einige Modelle, die ihre Strukturierung ermöglichen, vorgestellt.

2.1. Mobile Nutzung von Computern

Mobile Nutzungsszenarien ergeben sich aus dem mobilen Einsatz von Computern. Das Forschungsgebiet des *Mobile Computing* beschäftigt sich mit mobilen Endgeräten, der kommunikationstechnischen Infrastruktur sowie mit mobilen Anwendungen. Die Forschungen in dieser Richtung haben einige Begriffe geprägt, die nachfolgend erörtert werden.

2.1.1. Ubiquitous Computing

Ubiquitous Computing (deutsch: *ubiquitär* = „allgegenwärtig“, „überall verbreitet“) oder auch *Rechnerallgegenwart* bezeichnet die überall vorhandene Integration und Vernetzung von „intelligenten Gegenständen“ in der Umgebung eines Menschen. Der Begriff wurde durch die Beiträge von Mark Weiser geprägt [Tur06]. Weiser definiert Ubiquitous Computing wie folgt:

> *„[..] the method of enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user.“* [Wei93]

Demnach wird der Computer als Gerät verschwinden und durch intelligente Gegenstände ersetzt, die sich in das Alltagsleben integrieren. Der Computer ist als

solches also für die Menschen nicht mehr explizit wahrnehmbar, aber unterstützt sie bei der Aufgabenbewältigung ohne aufzufallen.

> *„In the 21st century the technology revolution will move into the everyday, the small and the invisible.“* [Wei91]

Das mobile Internet gewinnt bereits heute immer mehr an Bedeutung, der klassische Personal Computer steht dabei immer weniger im Mittelpunkt.

2.1.2. Pervasive Computing

Pervasive Computing (lat. *pervadere* = „durchdringen“) oder auch *Rechnerdurchdringung* ist die industrielle Ausprägung des Ubiquitous Computing.

Sie beschreibt die alles durchdringende Vernetzung intelligenter Gegenstände. Pervasive Computing wird oft synonym zu Ubiquitous Computing verwendet. Allerdings steht nicht ausschließlich die Integration von Computern in den menschlichen Lebensalltag im Fokus, sondern die Durchdringung allgemeiner Lebensbereiche sowie Geschäftsprozesse mit aktuell verfügbarer Technologie.

Die Kernaspekte des Pervasive Computing werden von Hansmann et al. [Han03] wie folgt zusammengefasst:

> *„Everywhere at anytime - This common slogan expresses in a nutshell the goal of Pervasive or Ubiquitous Computing. Both terms describe the visible and mobile front-end for the next generation of integrated IT applications. Pervasive Computing includes flexible and mobile devices like personal digital assistants, mobile phones, pagers, hand-held organizers and home entertainment systems, which will access or provide a rich diversity of applications.“* [Han03]

2.1.3. Nomadic Computing

Nomadic Computing, ein weiterer Begriff aus dem Mobile Computing, beschreibt die Benutzung portabler Geräte wie Laptops oder Handhelds in Verbindung mit mobiler Telekommunikationstechnologie. Dies soll dem Benutzer Zugriff auf das Internet sowie auf Daten seines Heim- oder Firmennetzwerks ermöglichen. Nomadic Computing legt seinen Schwerpunkt auf die Mobilität des Anwenders, der in diesem Zusammenhang als „Nomade“ bezeichnet wird.

Das Nomadic Computing beschäftigt sich vor allem damit, wie die Heterogenität der Systemkomponenten im Mobile Computing aus Sicht des nomadischen Benutzers transparent zu überwinden ist [Tur06].

2.2. Definition der Kontextbegriffe

Im Zusammenhang mit kontextbezogenen Systemen existieren zahlreiche Fachbegriffe und Definitionen. Die historische Entwicklung dieser Begriffe ist für das weitere

Verständnis dieser Arbeit maßgeblich. Der folgende Abschnitt erläutert die Begrifflichkeiten unter Berücksichtigung ihrer Entstehung und der unterschiedlichen Ansichten der jeweiligen Autoren.

2.2.1. Kontext

Aus allgemeiner Sicht wird Kontext als eine wechselseitige Beziehung zwischen den verschiedenen Bedingungen einer Situation, in der sich jemand oder etwas befindet, oder in der etwas passiert, angesehen. Das „Merriam-Webster Online Dictionary“ [Web05] beschreibt Kontext als

> *„The interrelated conditions in which something exists or occurs.“* [Web05]

Der Duden definiert Kontext als

> *„der umgebende, inhaltliche (Gedanken-, Sinn-, Sach-, Situations-) Zusammenhang“.* [Dud97]

Eine wichtige Rolle im Rahmen des Terminus Kontext spielen also die Begriffe *Bedingung*, *Zusammenhang* und *Umgebung*.

Diese allgemeine Definition ist jedoch zu unpräzise, um im Bereich informationstechnischer Systeme ein eindeutiges Verständnis zu schaffen. Fachautoren haben sich aus diesem Grund damit befasst, die Semantik des Begriffs Kontext im Umfeld eines mobilen Benutzers und einer mobilen Anwendung zu erörtern [Tur06].

2.2.2. Kontext mobiler Nutzungsszenarien

Die ersten Entwickler kontextbezogener Systeme erkannten, dass der allgemeine Kontextbegriff für ihre Forschung zu abstrakt ist und bemühten sich darum, neue Definitionen aufzustellen.

Erste Ansätze bestanden in Aufzählungen im Kontext enthaltener Attribute. Die ISO 13407 [ISO99] versucht Entwicklern von interaktiven Systemen Richtlinien vorzugeben, die festlegen, welche Kontextelemente bei der Entwicklung kontextbezogener Systemen relevant sind. Dazu zählen beispielsweise die Aufgaben und Attribute des Benutzers und die technische und physikalische Umgebung. Da die ISO 13407 jedoch als Leitfaden zu verstehen ist, liefert sie keine allgemeine Definition für Kontext.

Ebenfalls aufzählenden Charakter hat die Arbeit von Brown [Bro96], welche Kontextinformationen als

> *„der Ort, Identitäten von Personen in der Umgebung des Benutzers, Tageszeit, Jahreszeit, Temperatur, usw.“* [Bro96]

beschreibt.

Schilit et al. [Schi94] fassen den Kontextbegriff bereits etwas abstrakter:

> *„Three important aspects of context are: where you are, who you are with, and what resources are nearby.“* [Schi94]

Diese Darstellung beschränkt sich dennoch auf drei wichtige Kontextmerkmale und liefert daher keine vollständige Definition. Sie wird als *benutzerzentrierte* Definition klassifiziert, da sie die Aspekte des Kontextbegriffs aus Sicht des Benutzers einer mobilen Anwendung berücksichtigt.

Einen weiteren Versuch, den Kontext mobiler Nutzungsszenarien abstrakter zu fassen, machten Lieberman et al. [Lieb00]. Ihr Ansatz gilt als *systemzentriert*, da er im Gegensatz zu Schilits Ansatz nicht den Benutzer berücksichtigt, sondern das technische System, das die Kontextinformationen verarbeitet. Kontext ist demzufolge eine Ansammlung von Daten, die die Berechnung eines Systems beeinflussen, ausgenommen der expliziten Ein- und Ausgabedaten:

> *„Context can be considered to be everything that affects the computation except the explicit input and output.“* [Lieb00]

Die wohl etablierteste und am weitesten verbreitete Definition des Kontextbegriffs stammt von Dey und Abowd [Dey00]. In ihrer Arbeit untersuchen die Autoren bisherige Kontextdefinitionen und kommen zu dem Schluss, dass Aufzählungen von Beispielen oder Synonymen den Kontextbegriff niemals vollständig umfassen können. Diese Defizite sind nach Auffassung der Autoren durch folgende abstrakte Beschreibung zu eliminieren:

> *„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 applications themselves.“* [Dey00]

Dieser Definition zur Folge kann es sich bei Kontext um jede Art von Informationen aus dem Umfeld einer Anwendung handeln. Im Gegensatz zu [Lieb00] lässt sie sowohl implizite als auch explizite Informationen zu. Es ist also nicht relevant, ob die Informationen durch das System ermittelt oder den Benutzer angegeben werden. Weiterhin wird von einer konkreten technischen, sozialen oder physikalischen Umgebung abstrahiert, indem abstrakte Entitäten eingeführt werden. Diese schließen auch den Benutzer und die Anwendung selbst mit ein, welche somit auch zum Kontext gehören.

Nach dieser Auffassung ist Kontext also die Menge aller Informationen bezüglich der Situation einer Entität, die für das Verhalten einer Anwendung relevant ist. Einzelne Informationen aus dieser Menge werden als *Kontextinformationen* bezeichnet [Tur06].

2.2.3. Kontextbewusstsein

In Verbindung mit Mobile Computing wird häufig von *Kontextbewusstsein* oder *Context-Awareness* gesprochen. Der Begriff soll an dieser Stelle genauer untersucht werden.

Als *kontextbewusst*, *kontextbezogen* oder *kontextsensitiv* werden mobile Anwendungen bezeichnet, die Kontextinformationen dazu verwenden, um auf die aktuelle Ausführungsumgebung reagieren zu können. Ähnlich wie beim Begriff Kontext existieren für Kontextbewusstsein verschiedene Definitionen.

Schilit et al. [Schi94] beschreiben den Begriff *Context-Awareness* als:

> *„[..] the ability to adapts according to the location of use, the collection of nearby people, hosts, and accessible devices, as well as to changes to such things over time.“* [Schi94]

Ähnlich ihrer Definition des Kontextes wird der benutzerzentrierte Ansatz verfolgt. Der Benutzer steht also bei der Adaption des Kontextes durch das (mobile) System im Vordergrund. Der Ort, an dem er sich befindet, welche anderen Menschen und Geräte sich in seiner Umgebung befinden und welche Veränderungen dieser Entitäten im zeitlichen Verlauf auftreten, ist nach Auffassung der Autoren zentral für das Kontextbewusstsein.

Dey und Abowd [Dey00] stellen in ihrer Arbeit wieder eine abstraktere Beschreibung vor, indem sie bisherige Definitionen genauer betrachten. Anwendungen können ihnen zur Folge Kontextinformationen entweder direkt nutzen oder sich indirekt daran anpassen. Ihre verallgemeinerte Definition für *Context-Awareness* lautet:

> *„A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.“* [Dey00]

Eine ähnliche Definition, die auf einer kontextbezogenen Anwendung basiert, stellen Rothermel et al. [Rot03] auf:

> *„Eine Anwendung ist kontextbezogen (context-aware), wenn ihr Verhalten durch Kontextinformationen beeinflusst wird.“* [Rot03]

Demnach können ein oder mehrere Entitäten für das Verhalten einer Anwendung relevant sein. Kontextbewusste Anwendungen nutzen die Kontextinformationen dieser Entitäten, um sie dem Benutzer direkt bereitzustellen oder selbst adaptiv zu reagieren [Tur06]. Auf die Möglichkeiten der Adaption des Kontextes wird im nächsten Kapitel genauer eingegangen.

2.3. Adaption des Kontextes

Anders als traditionelle Anwendungen, deren Verhalten nur durch Benutzereingaben gesteuert wird, können kontextbezogene Anwendungen die aktuelle Umgebung

des Benutzers sowie die Situation durch Hardware- oder Softwaresensoren wahrnehmen und ihr Verhalten nach vordefinierten Regeln automatisch anpassen. Primäres Ziel kontextbezogener Applikationen ist es, die Geräte intelligenter zu machen und Menschen das Leben zu erleichtern [Du08].

Kontextadaptivität bezeichnet die Fähigkeit technischer Systeme, ihre Struktur, ihre Funktionalität oder ihr Verhalten zur Laufzeit zu ändern, um sich unterschiedlichen Gegebenheiten anzupassen. Das wichtigste Ziel der Kontextadaption ist das Erreichen von Ubiquität[1].

Das adaptive Verhalten kontextsensitiver Applikationen wird oft in zwei Hauptkategorien unterteilt, die *Pull* und *Push*-Typen. Während der Benutzer in der ersten Kategorie selbst Kontextinformationen oder Aktionen anfordert, leitet die Applikation in zweiten Kategorie adäquate Aktionen auf eigene Initiative auf Basis des Kontextes ein [Haek06].

Mehrere Autoren haben sich mit feineren Klassifikationen der Kontextadaption beschäftigt. Darunter fallen die Arbeiten von Schilit et al. [Schi94], Rothermel [Rot08] und Dey et al. [Dey01]. Diese beschreiben die folgenden Kategorien:

Die *kontextbezogene Selektion* beschreibt die Einbeziehung des Kontextes bei der Auswahl von Diensten und Informationen. Die Anwendung erleichtert es dem Benutzer, auf Basis der aktuellen Gegebenheiten eine Auswahl zu treffen. Die Klassifikation von Informationen nach ihrem Ort bzw. der Nähe zu einem Benutzer stellt bei vielen Anwendungen ein wesentliches Kriterium für die Auswahl von Diensten oder Informationen dar. Insbesondere weil Benutzer Informationen über ihre unmittelbare, erreichbare Umgebung benötigen, sind der Ort, die Identität und die Zeit als Primärkontext bei der kontextbezogenen Selektion relevant.

Bei der *kontextbezogenen Präsentation* verändert sich die Darstellung einer Anwendung in Abhängigkeit des Kontextes. Kontextinformationen werden beispielsweise dazu herangezogen, um die Anzeige von Informationen auf wesentliche Elemente zu reduzieren, indem eine Vorauswahl getroffen wird.

Kontextbezogene Aktionen sind von der Applikation auf Basis von Ereignissen – zum Beispiel Änderungen des Kontextes – selbstständig ausgelöste Aktionen.

2.4. Quality of Context

Der Kontext einer Entität ist in mobilen Umgebungen von großer Dynamik geprägt. Deshalb ist nicht nur der Wert eines Kontextattributs von Bedeutung, sondern auch der Zeitpunkt seiner Erfassung. Generell sind Kontextquellen in ihrer Fähigkeit eingeschränkt, den wahren und präzisen Wert eines Kontextattributs zur Verfügung zu stellen. Einer Kontextquelle beispielsweise, die auf Daten aus einem elektronischen Kalender basiert, kann nicht als zuverlässig eingeschätzt werden, wenn der Benutzer diesen nicht regelmäßig pflegt [Sin06]. Insbesondere aus Sensoren ermittel-

[1] Siehe Kapitel 2.1.1: Ubiquitous Computing

te Kontextinformationen können ungenau, unvollständig oder widersprüchlich sein [Baus02].

Um diesen Aspekt bei der Modellierung des Kontextes zu berücksichtigen, definieren Buchholz et al. [Buch03] *Quality of Context (QoC)* als

> *„any information that describes the quality of information that is used as context information“.* [Buch03]

Die Kontextqualität umfasst also zusätzliche Metadaten, die die Qualität einer Kontextinformation beschreiben. Folgende Qualitätsparameter sind nach van Sinderen et al. [Sin06] für die Kontextmodellierung relevant:

- *Accuracy:* Die Differenz zwischen dem Wert des Kontextattributs und dem Aspekt aus der Realität, den es repräsentiert
- *Probability of correctness:* Die von der Kontextquelle eingeschätzte Wahrscheinlichkeit, dass sie den richtigen Wert bereitstellt
- *Trustworthiness:* Die vom Empfänger eines Kontextattributs eingeschätzte Wahrscheinlichkeit, dass die Kontextquelle den richtigen Wert bereitgestellt hat
- *Up-to-dateness:* Das Alter des Wertes eines Kontextattributs, typischerweise repräsentiert als Zeitstempel

2.5. Kontextvariablen

Dieses Kapitel behandelt die Kategorisierung von Kontextinformationen, die in diesem Zusammenhang auch als *Kontextvariablen* oder *Kontextattribute* bezeichnet werden. Mit dieser Thematik haben sich zahlreiche Autoren beschäftigt, deren Arbeiten zu verschiedenen Modellen und Hierarchien geführt haben. Diese werden im Folgenden dargestellt.

2.5.1. Primär- und Sekundärkontext

Dey et al. [Dey00] unterscheiden Kontextinformationen nach *Primär-* und *Sekundärkontexten*. Ihrer Aussage nach besitzen die Primärkontexte *Ort*, *Identität*, *Zeit* und *Aktivität* einen höheren Stellenwert als Sekundärkontexte. Die Begründung liegt neben der häufigeren Verwendung dieser Informationen in der Verdeutlichung von Abhängigkeiten zwischen den Kontexten:

> *„Location, identity, time, and activity are the primary context types for characterizing the situation of a particular entity. These context types not only answer the questions of who, what, when, and where, but also act as indices into other sources of contextual information.“* [Dey00]

Sekundärkontexte sind den Primärkontexten also logisch untergeordnet, wodurch eine Hierarchie entsteht. Primärkontexte können zur Auswahl und Einschränkung

weiterer Kontextinformationen verwendet werden, es können also relevante Kontextinformationen gefiltert werden.

Aus einem Primärkontext können sekundäre Kontextinformationen abgeleitet werden, wofür die Autoren einige Beispiele geben. Die Identität beispielsweise lässt auf Telefonnummer, Adresse, Geburtstadum oder Namen schließen. Ebenso können Ortsinformationen zur Identifizierung anderer Objekte oder Personen in der Umgebung herangezogen werden [Tur06].

2.5.2. Ebenen von Kontexttypen

Chen et al. [Chen00] unterscheiden verschiedene Ebenen von Kontextinformationen. Danach werden Kontexte nach *Low-Level*- und *High-Level*-Typen unterschieden.

Der *Low-Level*-Kontext ist die Menge der Kontextinformationen, die sich unmittelbar durch physische oder logische Sensoren erheben lassen. Physikalische Sensoren liefern Informationen über physikalische Bedingungen der Umwelt und logische Sensoren liefern Informationen aus dem mobilen Gerät selbst. Diese Kontextdaten zeichnen sich dadurch aus, dass sie eine einfache Struktur besitzen. Beispiele für *Low-Level*-Kontexttypen sind Ort, Zeit, optische Informationen, akustische Informationen, biologische Informationen usw.

Durch Verarbeitung und Kombination mehrerer *Low-Level*-Kontexttypen lassen sich komplexe Informationen gewinnen, die nicht mehr unmittelbar von Sensoren ermittelt werden können. Beispielsweise kann durch die Kombination von Ort, Zeit und Kalender eines Benutzers auf die Tatsache zu schließen sein, dass er sich gerade in einer Konferenz befindet. Diese Informationen werden den Autoren zur Folge als *High-Level*-Kontext bezeichnet [Tur06].

2.5.3. Strukturierung von Kontextvariablen

Die Anführung einer Reihe von Beispielen ist ein vielfach angewandter Ansatz zur Beschreibung des Kontextes, wie Kapitel 2.2.2 bereits dargelegt hat. Bisgaard et al. [Bis04] erstellten in ihren Untersuchungen eine Tabelle, in der sie die Beschreibungen der Kontextvariablen verschiedener Arbeiten gegenüberstellen (siehe Tabelle 2.1).

Aus dieser Tabelle können einige zentrale Erkenntnisse gewonnen werden. Zunächst scheinen die Faktoren Ort, Zeit, Identität und Umgebung Schlüsselvariablen zu sein. Des Weiteren fällt auf, dass die ausgewählten Variablen auf den Zielen der Projekte basieren, die die jeweiligen Autoren in ihren Arbeiten beschreiben. Ersichtlich ist auch die Bedeutung der Dynamik des Kontextes. Eine Kontextvariable, die in einer Situation von großer Bedeutung ist, kann in einer anderen Situation keinen Einfluss mehr besitzen.

Die Forschungen in diese Richtung haben die Tragweite der Thematik verdeutlicht und zu verschiedenen Ansätzen geführt, Kontextelemente zu gruppieren. Beigl et al. [Beigl99] entwickelten das Modell des Kontext-Merkmalsraumes, um das Konzept

Tabelle 2.1.: Vergleich von Kontextvariablen (nach [Bis04])

	Ort	Zeit	Identität	Umgebnung	Soziales Umfeld	Netzwerk	Jahreszeit	Geschichte	Aufgabe/Aktivität	Gerät
Ryan et al. [Ryan97]	•	•	•	•						
Brown et al. [Bro97]	•	•	•	•			•			
Schilit et al. [Schi94], [The94]	•		•	•	•	•				
Rahlff et al. [Rah01]	•	•	•	•					•	
Beale et al. [Bea04]								•		
Hess et al. [Hess03]	•	•	•	•					•	•
Dey et al. [Dey99]	•			•						

des Kontextes zu strukturieren. Demnach beschreibt jeder Kontext eine Situation und die Umgebung, in der sich das Gerät oder der Benutzer befinden. Jeder Kontext wird durch einen eindeutigen Namen identifiziert und enthält einen Satz relevanter Merkmale. Für jedes dieser Merkmale wird implizit oder explizit ein Wertebereich definiert.

Auf Basis dieses Modells kann eine hierarchische Struktur aufgebaut werden. Die Autoren verwenden in ihrem Beispiel menschliche Faktoren und die physikalische Umgebung als oberste Ebene, die sich dann weiter verzweigt. Zusammen mit der zeitlichen Dimension des gesamten Merkmalsraumes entsteht der Gesamtkontext (siehe Abbildung 2.1).

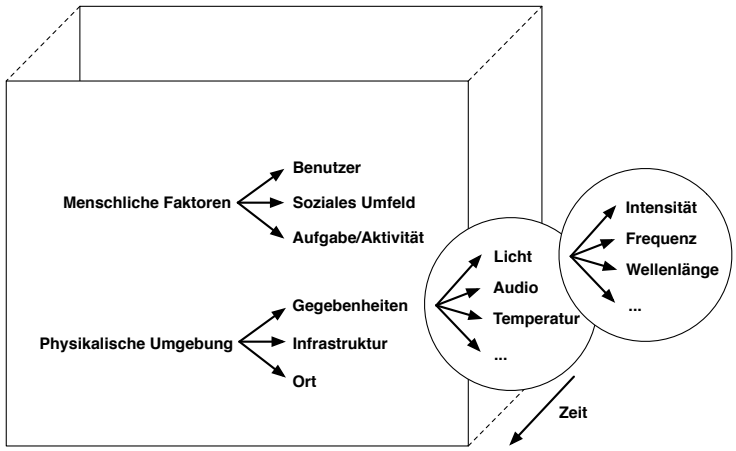

Abbildung 2.1.: Kontext-Merkmalsraum (nach [Beigl99])

Schmidt et al. [Schmi99] unterteilen Kontextinformationen in drei gleichgestellte Dimensionen: die Identität des Benutzers bzw. des Gerätes selbst, die Aktivität sowie die Umgebung.

Dieser Ansatz unterscheidet sich von der hierarchischen Aufteilung nach [Beigl99] darin, dass die Aktivität keine untergeordnete Kategorie der menschlichen Faktoren darstellt, sondern jegliche Aktivitäten umfassen kann, die für mobile Anwendungen relevant sind. Darunter sind also neben Aktivitäten des Benutzers auch Prozesse, die das Gerät ausführt [Tur06] (siehe Abbildung 2.2).

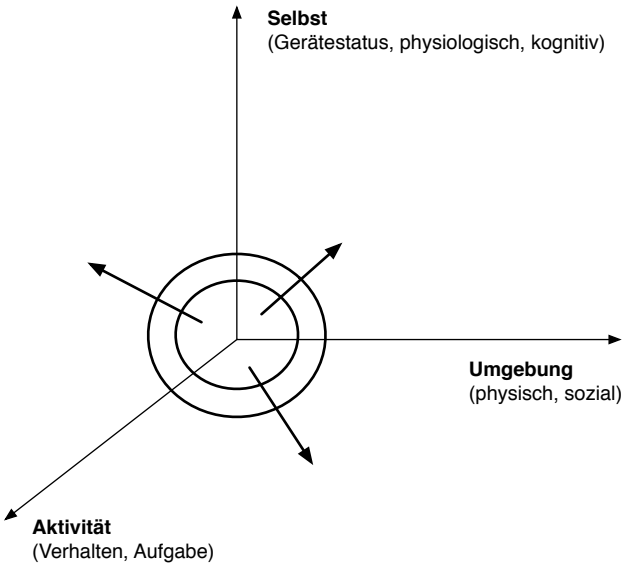

Abbildung 2.2.: 3D-Kontext-Klassifizierung (nach [Schmi99])

3. Kontextsensitive Anwendungen

In Kapitel 2 wurde auf abstrakter Ebene auf die mobile Nutzung von Computern, die verschiedenen Kontextbegriffe, die Adaption des Kontextes sowie verschiedene Modelle zur Gruppierung und Kategorisierung von Kontextvariablen eingegangen. Darauf aufbauend gibt dieses Kapitel einen Überblick über kontextsensitive Anwendungen. Dazu werden verschiedene Kategorien erläutert und Beispiele angeführt. Des Weiteren werden die Herausforderungen und Gestaltungsziele der Entwicklung kontextsensitiver Anwendungen erörtert.

3.1. Einsatzgebiete

Kontextsensitive Anwendungen haben in letzter Zeit viel Aufmerksamkeit erlangt. Damit sind Applikationen gemeint, die sich – bis zu einem gewissen Grad – ihres Nutzungskontextes bewusst sind. Dieses Bewusstsein kann aus Informationen gewonnen werden, wie aus (typischerweise in das Gerät integrierte) Umgebungssensoren für Temperatur, Helligkeit, geografischer Position sowie Mikrofon, Kamera, Netzwerkverbindungen zu anderen Geräten und das Wissen über die Ziele und Präferenzen des Nutzers. Frühe Forschungsansätze haben bereits eine Vielzahl ortsabhängiger Anwendungen wie z.B. Einkaufsassistenten und Städteführer hervorgebracht [Haek06]. Neuere Entwicklungen profitieren von immer mehr Kontextquellen [Ball07]:

- Durch die geographische Position (z.B. aus GPS) kann z.B. der Reisestatus, die Pünktlichkeit für ein Meeting oder die aktuelle Tätigkeit ermittelt werden. Befindet sich die Position z.B. über einer Bahnlinie, fährt der Nutzer gerade Bahn bzw. wartet gerade auf eine Bahn.
- Die exakte Position (z.B. aus WiFi-Netzwerk, Bluetooth oder RFID-Lesern) kann sehr gezielte Informationsübertragungen ermöglichen.
- Bewegungs- oder Temperatursensoren können Bewegungen des Nutzers, Lufttemperatur und Gesten erkennen. Diese können z.B. für die Detektion von Stimmungen herangezogen werden.
- Mit Hilfe von digitalen Kalendern können Nutzeraktivitäten ermittelt werden. Wenn sich der Nutzer in einem Meeting befindet, können z.B. akustische Signale unterbunden werden.
- Kameras können Bilder entweder direkt erfassen oder Bildteile wie Barcodes, Gesichter, Verkehrszeichen oder andere Metadaten der Umgebung erkennen.

Dieser Auszug vermittelt einen Eindruck der Vielfalt möglicher Einsatzgebiete für kontextsensitive Applikationen. Wenn Informationsquellen intelligent kombiniert werden, können sie Benutzern große Vorteile verschaffen, die jetzt erst in ihren Grundzügen erforscht werden [Ball07].

3.2. Kategorien

Bauer et al. [Bau06] haben in ihrer Arbeit eine Kategorisierung kontextsensitiver Applikationen vorgenommen. Auch wenn die Einsatzgebiete dieser Anwendungen ständig in neue Bereiche vordringen, gibt diese Kategorisierung einen Eindruck möglicher Anwendungsszenarien. Tabelle 3.1 gibt einen Überblick über diese Kategorien, auf die nachfolgend kurz eingegangen wird.

Tabelle 3.1.: Kategorien kontextsensitiver Applikationen (nach [Bau06])

Hauptkategorie	**Unterkategorien**
1. Standortlokalisierungsdienste	Dienste zur Lokalisierung von Personen Dienste zur Lokalisierung von Objekten
2. Navigationsdienste	Generische Navigationsdienste Spezifische Navigationsdienste
3. Informationsdienste	Generische Informationsdienste Interaktive Informationsdienste
4. Kommunikationsdienste	B2C Kommunikationsdienste B2B Kommunikationsdienste
5. Unterhaltungsdienste	
6. Transaktionsdienste	

1. **Standortlokalisierungsdienste** decken die exakte Lokalisierung von Personen oder Objekten ab. Beispiele für Personenlokalisierung ist das Auffinden von Kindern, Lokalisierung von verunfallten Autofahrern, Leitsysteme für blinde und bedürftige Menschen, Auffinden von Freunden, standortbezogene Chats, Gruppenmanagement usw. Beispiele für Objektlokalisierung sind das Auffinden von Mobiltelefonen oder die Lokalisierung von Taxis.
2. **Navigationsdienste** führen mobile Nutzer von ihrer aktuellen Position zu einer bestimmten Zielposition. Wenn ein Nutzer die Zielposition explizit angibt, beschreibt dies einen generischen Navigationsdienst. Wenn ein Nutzer sich zum Standort eines bestimmten Dienstes (z.B. Restaurant, Tankstelle) leiten lässt, handelt es sich um einen spezifischen Navigationsdienst.
3. **Informationsdienste** bieten Informationen an, die für die aktuelle Situation des Nutzers relevant sind. Mobile Stadtführer, Wetterinformationen, Stauinformationen sowie Fahrpläne sind Beispiele für generische Informationsdienste. Interaktive Informationsdienste sind Dienste, bei denen der Nutzer nach Erhalt

der gewünschten Information interaktiv reagieren kann. Die Bestellung einer Pizza nach Erhalt der Angaben zur nächstgelegenen Pizzeria oder Ereignis-Informationssysteme mit der Möglichkeit der Einladung von Freunden sind Beispiele für diese Dienstkategorie.

4. **Kommunikationsdienste** für B2C *(Business-to-Consumer)* und B2B *(Business-to-Business)* zielen darauf ab, die Kommunikationsmöglichkeiten zwischen Benutzern und Unternehmen zu optimieren. Nicht angeführt sind hier Kommunikationsdienste zwischen Nutzern untereinander, wie soziale Netzwerke, denen in letzter Zeit viel Aufmerksamkeit zu Teil wurde.
5. **Unterhaltungsdienste** sind zum Beispiel ortsabhängige Spiele und standortbezogene TV- und Radioprogramme.
6. **Transaktionsdienste** beinhalten beispielsweise die Ausführung finanzieller Transaktionen und automatische Buchungen.

3.3. Herausforderungen und Gestaltungsziele

Eine Applikation kontextsensitiv zu machen, ist eine zentrale Herausforderung ihrer Entwicklung. Die Forschung im Bereich des *Context-aware Computing* beschäftigt sich daher mit Fragen der *Kontextermittlung*, der *Kontextrepräsentation*, der *Kontextinterpretation* und der *Kontextabstraktion* sowie mit Programmiermodellen und Prinzipien zur Unterstützung der Entwicklung.

Um Entwickler bei der Erstellung dieser Applikationen zu unterstützen beschreiben Häkkilä et al. [Haek06] in ihrer Arbeit zehn Richtlinien für die Entwicklung kontextsensitiver mobiler Applikationen. Diese wurden auf Basis von Literaturstudien zum aktuellen Stand der Forschung erstellt.

1. **Entscheidungsunsicherheit** Die Entscheidungen kontextbezogener Applikationen können auf unsicheren Informationen in Bezug auf die Kontextermittlung basieren. Entwickler sollten abwägen, wann der Nutzer über diese Unsicherheit informiert werden sollte. Unter Umständen sollte er explizit dazu aufgefordert werden, die Ausführung einer Aktion zu bestätigen.
2. **Vermeidung von Unterbrechungen** Entwickler sollten die Ausführung von Aktionen so priorisieren, dass der Nutzer in seiner Tätigkeit nicht unterbrochen wird. Außerdem sollte er vor unnötigen Informationen – beispielsweise durch Filterung – geschützt werden.
3. **Personalisierung** Personalisierung sollte die individuellen Bedürfnisse des Nutzers befriedigen. Dies kann beispielsweise durch Filterung nach Nutzerpräferenzen geschehen. Dabei sollte berücksichtigt werden, dass sich die Präferenzen mit der Zeit ändern können.
4. **Vermeidung von Informationsüberfluss** Die Applikationen sollten dem Nutzer nicht zu viele Informationen auf einmal präsentieren, die angebotenen Informationen sollten aussagekräftig und verständlich dargestellt werden. Besondere

Vorsicht ist bei Applikationen des *Push*-Typs geboten, die sehr schnell viele Benachrichtigungen und Hinweise erzeugen können.

5. **Schutz persönlicher Daten** Applikationen, die den Informationsaustausch unterstützen, sollten besondere Rücksicht auf die Privatspäre des Nutzers nehmen. Darüber hinaus sollte eine Möglichkeit zur Anonymisierung angeboten werden. Der soziale Kontext kann Auswirkungen auf die wahrgenommene Privatsphäre des Nutzers haben, weshalb sich das Verhalten der Applikationen unterschiedlich auswirken kann.
6. **Berücksichtigung der Mobilität** Eine schnelle und einfache Interaktion mit der Anwendung sollte möglich sein, damit der Nutzer sie auch bedienen kann, wenn er unterwegs ist bzw. sich in Bewegung befindet. Außerdem sollten Auswirkungen der Mobilität auf die Verfügbarkeit von Kontextinformationen – wie Qualität von Netzwerkverbindungen oder Genauigkeit von Ortsinformationen – Berücksichtigung finden.
7. **Sicherung der Nutzerkontrolle** Der Nutzer sollte jederzeit die Kontrolle über das Gerät besitzen. Dies kann auf unterschiedliche Art und Weise anhängig von der Applikation realisiert werden. Adaptive Benutzungsschnittstellen sollten beispielsweise eine nicht-adaptive Option besitzen. Darüber hinaus sollten Applikationen bei Bedarf den Grad der Automation regulieren und sich gegebenenfalls eine Bestätigung des Nutzers einholen, anstatt vollautomatisch Aktionen auszuführen.
8. **Zugriff auf den Kontext** Der Nutzer sollte Kontextattribute und ihre Darstellung verändern können, um die Verständlichkeit der Applikation zu erhöhen. Vordefinierte Einstellungen können vom mentalen Modell des Nutzers abweichen, da das Verständnis von Kontextattributen subjektiv ist.
9. **Sichtbarkeit des Systemstatus** Da die Adaption des Kontextes zu Veränderungen auf Basis der aktuellen Situation führt, ist der Nutzer über den Status des Systems zu informieren, damit er versteht, was die Applikation gerade tut.
10. **Zweckmäßigkeit** Ganzheitlich betrachtet sollte die Zweckmäßigkeit der kontextsensitiven Applikation evaluiert werden. Es sollte geprüft werden, ob die Adaption einen Mehrwert für den Nutzer liefert oder ob sie unter Umständen zu Verunsicherung führen kann.

Diese Richtlinien beziehen sich auf den Entwicklungsprozess für mobile kontextsensitive Applikationen. Da sich diese Arbeit im weiteren Verlauf mit Rahmenwerken befasst, die die Entwicklung dieser Applikationen vereinfachen sollen, bilden diese Richtlinien ein wichtiges Fundament und müssen bei der Konzeption solcher Systeme berücksichtigt werden.

4. Modellierung und Repräsentation des Kontextes

Die vorausgehenden Kapitel haben nun alle wichtigen Grundlagen und Themenkomplexe aufgegriffen, die für die Konzeption eines Systems zur Kontextrepräsentation und -verwaltung maßgeblich sind. Eine erste Herausforderung ist die Frage nach der Modellierung von Kontexten, um sie in einem informationstechnischen System abbilden zu können.

Dieses Kapitel behandelt daher die Modellierung und Repräsentation des Kontextes durch Kontextmodelle. Zunächst werden das Konzept der Kontextmodelle erläutert und etablierte Ansätze zur Kontextmodellierung aus der Literatur beschrieben. Die Modellierungsansätze werden anschließend im Hinblick auf die Zielsetzung dieser Arbeit bewertet. Besondere Aufmerksamkeit wird dabei auf die semantische Kontextmodellierung gelegt. Der letzte Teil resümiert schließlich die Erkenntnisse der Untersuchungen dieses Kapitels.

4.1. Kontextmodelle

Um Kontextinformationen in einer Applikation beziehen zu können, müssen diese durch ein geeignetes Modell repräsentiert werden. Ein Kontextmodell ist ein Informationsmodell, welches explizit zur Repräsentation von Kontext entworfen wurde. Es definiert und speichert sämtliche Kontextdaten in maschinenlesbarer Form und beschreibt folglich Kontextattribute, ihre Beziehungen zueinander und ihre Werte [Sin06]. Die Herausforderung bei der Entwicklung eines solchen Modells liegt in dem praktisch unüberschaubaren Umfang möglicher Kontextfaktoren, weshalb eine möglichst generische und flexible Konzeption anzustreben ist [Weiss07]. Die Gestaltung des Kontextmodells ist ein entscheidender Faktor für die Repräsentation des Kontextes in jedem kontextbezogenen System und aufgrund dessen ein aktives Forschungsgebiet.

Erste Forschungsergebnisse fokussierten die Kontextmodellierung für bestimmte Applikationen oder bestimmte Klassen von Applikationen. Inzwischen sind jedoch generische Kontextmodelle von großem Interesse, da viele Anwendungen von ihnen profitieren können. Aktuelle Forschungsgegenstände sind die Entwicklung von einheitlichen Kontextmodellen, Repräsentations- und Abfragesprachen sowie Reasoning-Algorithmen, um den Austausch von Kontextinformationen und die Interoperabilität zwischen Applikationen zu ermöglichen [Stra04].

4.2. Ansätze zur Kontextmodellierung

Eine Klassifizierung der verschiedenen Ansätze zur Kontextmodellierung lässt sich nach den verwendeten Datenstrukturen vornehmen. Eine solche technologiebasierte Klassifizierung führen Strang et al. [Stra04] durch, indem sie Kontextmodelle in sechs Klassen einordnen. Diese Klassen unterscheiden sich in der verwendeten Technik zur Repräsentation der Kontextinformationen.

4.2.1. Schlüssel-Wert Modelle

Die einfachste Datenstruktur zur Modellierung von Kontextinformationen ist das Modell der Schlüssel-Wert-Paare. Der Kontextwert wird dabei einem Schlüssel zugeordnet und als Umgebungsvariable im System abgelegt. Ein einfaches Beispiel für ein solches Schlüssel-Wert-Paar ist in Quellcode 4.1 zu sehen.

Quellcode 4.1: Beispiel für ein Schlüssel-Wert-Paar

```
1 USER_POS : 52 deg 1' 30.00" N, 8 deg 33' 28.20" E
```

Eine kontextbezogene Anwendung greift über diesen Schlüssel auf den entsprechenden Kontextwert zu. Bereits Schilit et al. [Schi94] wandten diese Methode an, um Anwendungen Kontextinformationen zur Verfügung zu stellen.

Schlüssel-Wert-Paare sind einfach zu verwalten, eignen sich jedoch nicht zur differenzierten Strukturierung der Kontextvariablen, um effiziente Algorithmen zur Auswertung des Kontextes zu ermöglichen [Stra04], [Tur06].

4.2.2. Markup-Schema Modelle

Markup-Schema Modelle repräsentieren Kontextinformationen in Form von Markup-Tags mit Attributen und Inhalten, die wiederum weitere Markup-Tags beinhalten können, was zu einem rekursiven Aufbau der Datenstruktur führt. Diese Modelle werden typischerweise zur Strukturierung von Profilinformationen eingesetzt, da sie sich aufgrund ihrer Universalität und ihrer weiten Verbreitung auf verschiedenen Plattformen zum Austausch von Kontextinformationen eignen. Sie basieren auf einer Serialisierung in SGML[1]-Dialekten.

Vertreter von Markup-Schema Modellen sind *CSCP (Comprehensive Structured Context Profiles)* [Held02] und *CC/PP Context Extension* [Indu03], die sich am W3C-Standard *CC/PP (Composite Capability/Preference Profiles)* [W3C07] orientieren und XML zur Serialisierung verwenden. CC/PP-Profile werden mit RDF[2] beschrieben und lassen sich beispielsweise im HTTP-Header an einen Webserver senden, um den Kontext des Clients mit zu übermitteln. Ein einfaches Beispiel eines CC/PP Profils, welches die Informationen zu Bildschirmauflösung und Arbeitsspeicher eines Gerätes enthält, zeigt Quellcode 4.2. [Stra04], [Tur06]

[1] *Standard Generic Markup Language*, die Ursprache aller Markup-Sprachen (wie z.B. XML)
[2] *Resource Description Framework* (`http://www.w3.org/RDF/`)

Quellcode 4.2: Beispiel für ein CC/PP-Profil (nach [W3C07])

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex="http://example.com/Schema#">
         <rdf:Description rdf:about="http://example.com/HardwareDefaults">
                  <rdf:type rdf:resource="http://example.com/Schema#Hardware" />
                  <ex:displayHeight>400</ex:displayHeight>
                  <ex:displayWidth>600</ex:displayWidth>
                  <ex:memoryMb>32</ex:memoryMb>
         </rdf:Description>
</rdf:RDF>
```

Das Projekt *ConteXtML* [Ryan99] ist ein XML-basiertes Protokoll, welches den Austausch von kontextrelevanten Informationen ermöglicht. Ein mobiler Client kann seinen eigenen Kontext an einen Kontextserver übermitteln und selbst benötigte Informationen dazu anfordern. Ein Beispiel dazu findet sich in Quellcode 4.3. Identität und Position des Benutzers werden an den Server übermittelt und Karteninformationen dazu werden angefordert. [Tur06]

Quellcode 4.3: Beispiel für ein ConteXtML-Protokoll (nach [Ryan99])

```
<context session="new">
         <person role="user" first="Felix" last="Mueller" />
         <spatial proj="UTM" zone="33" datum="2009-02-28␣14-38-04">
                  <point x="271896" y="3686591" z="139" />
         </spatial>
         <require>
                  <map name="contours" />
                  <map name="roads" class="minor" />
                  <map name="buildings" />
         </require>
</context>
```

4.2.3. Grafische Modelle

Ein bekanntes universelles Modellierungsinstrument ist die *Unified Modeling Language (UML)*, welche eine starke grafische Komponente besitzt (UML-Diagramme). Durch ihre generische Struktur ist UML ebenfalls zur Modellierung von Kontext geeignet. Bauer modelliert in seiner Arbeit [Bau03] beispielsweise den Kontext des Luftverkehrsmanagements als Erweiterungen zu UML. Van Sinderen et al. verwenden ebenfalls UML für das Kontextmodell ihres Infrastruktur-Ansatzes [Sin06], [Stra04] (siehe Abbildung 4.1).

4.2.4. Objektorientierte Modelle

Objektorientierte Kontextmodelle machen sich allgemeine objektorientierte Paradigmen zur Repräsenation der Kontextinformationen zu Nutze. Konzepte wie Kapselung, Wiederverwendbarkeit und Vererbung sollen dabei helfen, sich dem Problem der Dynamik des Kontextes in mobilen Umgebungen anzunähern. Die Details der

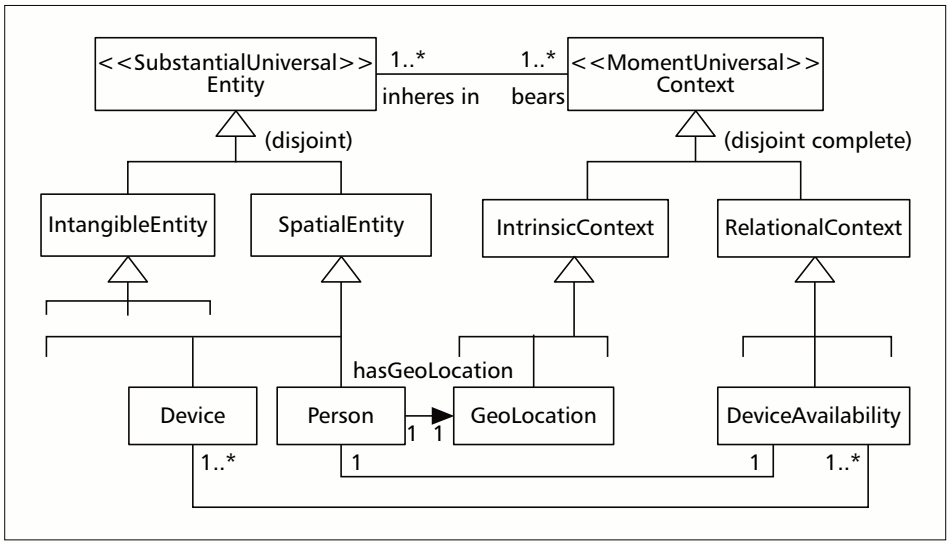

Abbildung 4.1.: Beispiel für ein UML-Kontextmodell (nach [Sin06])

Aufbereitung des Kontextes werden auf Objektebene gekapselt und somit abstrahiert. Der Zugriff wird durch spezifizierte Schnittstellen ermöglicht [Stra04].

Ein typischer Vertreter der objektorientierten Kontextmodellierung ist das TEA-Projekt [Laer01]. Dieser Ansatz nutzt so genannte *Cues*, um physische und logische Sensoren zu abstrahieren. Jedes *Cue* bezieht seine Informationen aus genau einem Sensor, wobei auch mehrere *Cues* auf einen Sensor zugreifen können. Für jedes *Cue* wird eine endliche oder unendliche Menge an Werten definiert, die dieses repräsentieren kann. Komplexe Kontextinformationen werden durch die Zusammenführung der Informationen aus den einzelnen *Cues* – dem Fusionsprozess – gewonnen. Der Kontext wird bei diesem Ansatz als Abstraktionsschicht über den verfügbaren *Cues* modelliert. Die *Cues* sind folglich Objekte, die ihre Informationen durch ihre Schnittstellen zur Verfügung stellen und die Details der Datenermittlung verbergen [Stra04], [Tur06]. Abbildung 4.2 veranschaulicht die Architektur des beschriebenen Systems.

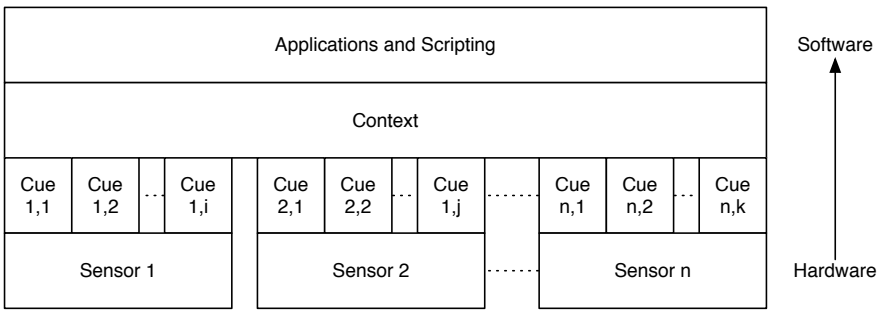

Abbildung 4.2.: Architektur des TAE-Systems (nach [Laer01])

Ein weiterer Ansatz in der objektorientierten Kategorie ist das *Active Object Model* aus dem GUIDE-Projekt [Chev99]. Auch hier bestand die Motivation darin, eine Vielzahl von Kontextinformationen zu Personen und Umgebung zu verwalten und

gleichzeitig die Skalierbarkeit zu erhalten. Alle Details zur Ermittlung und Fusion von Daten werden in Objekten gekapselt und vor den sonstigen Teilen des Systems verborgen.

Eine ähnliche Intention verfolgen auch Bouzy und Cazenave [Bou97]. Sie benutzen generische objektorientierte Mechanismen, um kontextrelevantes Wissen über Zeit, Ziele, Orte und globale Kontexte zu repräsentieren. Sie begründen die Verwendung des objektorientierten Ansatzes damit, die Definition der kleinsten Anzahl an Eigenschaften, Funktionen und Regeln anzuwenden, um die Wissensrepräsentation in komplexen Domänen und Systemen zu ermöglichen [Stra04].

4.2.5. Logikbasierte Modelle

Eine Logik definiert die Bedingungen, auf Basis derer ein Ausdruck oder eine Tatsache von einer Reihe anderer Ausdrücke oder Tatsachen abgeleitet werden kann. Dieser Vorgang des logischen Schließens wird als *Reasoning* oder *Inferencing* bezeichnet. Um diese Bedingungen in einer Reihe von Regeln zu beschreiben, wird ein formales System angewandt. In einem logikbasierten Kontextmodell wird der Kontext folglich als Tatsachen, Ausdrücke und Regeln definiert. Kontextinformationen werden durch Hinzufügen, Modifizieren und Entfernen von Fakten und Logikregeln in das System eingebracht, verändert bzw. entfernt. Durch logisches Schließen lassen sich Kontextinformationen höheren Grades ermitteln.[3] [Stra04]

Aufgrund des hohen Formalitätsgrades eignen sich logikbasierte Modelle ebenfalls zur Beschreibung des Kontextes im Bereich der Künstlichen Intelligenz. McCarthy und Buvac [McCa97], als Vertreter dieses Modells, geben ein Formalisierungskonzept an, welches eine Darstellung einfacher, allgemein geltender, statischer Erscheinungen als Axiome ermöglicht. Diese lassen sich zu einem komplexen und sich verändernden Kontext abstrahieren [Tur06].

Ein weiterer Ansatz ist das *Sensed Context Model* von Gray und Salber [Gray01]. Sie verwenden Prädikatenlogik erster Ordnung, um kontextuelle Aussagen und Relationen formal zu beschreiben. Ein ähnlicher Ansatz dieser Kategorie ist das *Multimedia System* von Bacon et al. [Bac97]. In diesem System wird der Ort als ein Aspekt des Kontextes als Menge von Fakten in einem regelbasierten System ausgedrückt.

Logikbasierte Modelle ermöglichen kaum Möglichkeiten zur Strukturierung komplexer Kontextinformationen. Des Weiteren sind diese Systeme komplex und daher sehr ressourcenintensiv, was den Einsatz in mobilen Systemen erschwert [Tur06].

4.2.6. Ontologiebasierte Modelle

Ontologien sind Instrumente der Wissensrepräsentation und dienen der formalen Beschreibung von Systemen mit Hilfe von Konzepten und Relationen. Darüber hinaus beinhalten sie Inferenz- und Integritätsregeln. Da ihre Datenstrukturen maschinell

[3] Siehe Kapitel 4.4.1: Context Reasoning

zu verarbeiten sind, eignen sie sich insbesondere für Aufgaben der Informationsbeschreibung. Getrieben durch die Vision des *Semantic Web* sind formale Sprachen wie *RDF-Schema*[4], *DAML+OIL*[5] und *OWL*[6] entstanden.

Ontologiebasierte Modelle nutzen sowohl objektorientierte als auch logikbasierte Prinzipien, da die Kontexte als Klassen von Objekten sowie ihren expliziten Beziehungen zueinander modelliert werden können.

Erste Ansätze zur Kontextmodellierung mit Ontologien wurden von Öztürk und Aamodt vorgestellt. Sie untersuchten den Unterschied zwischen dem Erinnern und dem Erkennen von unterschiedlichen Sachverhalten in Verbindung mit Kontextinformationen in ihren psychologischen Studien [Oez97]. Der Notwendigkeit, Wissen verschiedener Domänen zu normieren und zu kombinieren, konnte mit Ontologien aufgrund ihrer Möglichkeiten der Wissensmodellierung Rechnung getragen werden.

Das *ASC-Modell (Aspect Scale ContextInformation)* [Stra03] spezifiziert ein einheitliches Vorgehen, um die Kernkonzepte und eine beliebige Anzahl von Subkonzepten mit Hilfe von Ontologien zu definieren. Der Ansatz ermöglicht es dadurch, Kontextwissen zu teilen und wieder zu verwenden. Eine Evaluation des Kontextwissens findet durch Ontologie-Reasoner statt. Dieses Modell wurde unter Anwendung von ausgewählten Ontologie-Sprachen implementiert [Stra04].

Quellcode 4.4 zeigt ein Beispiel aus der *Context Broker Architecture (CoBrA)* [CheFi04]. In CoBrA wird ein OWL-basierter Ansatz verwendet, wobei die Struktur und das Vokabular in RDF beschrieben sind.

Quellcode 4.4: Beispiel für eine Kontext-Ontologie (nach [Bald06])

```
<loc:LocationContext>
        <rdf:type rdf:resource="&tme;InstantThing" />
        <loc:locationContextOf>
                <per:Person>
                        <per:name rdf:datatype="&xsd;string">
                                Felix Mueller
                        </per:name >
                </per:Person>
        </loc:locationContextOf>
        <loc:boundedWithin rdf:resource="&ebgeo;Germany" />
        <tme:at rdf:datatype="&xsd;dateTime">
                2010-02-28 -23T11:23:00
        </tme:at>
</loc:LocationContext>
```

Wang et al. entwickelten mit dem *CONON-Modell (CONtext ONtology)* [Wang04] ebenfalls einen Ansatz auf Basis von Ontologien, der das Teilen, Wiederverwenden und Schlussfolgern von Kontextwissen ermöglicht. Das Modell ist in der Abbildung 4.3 zu sehen.

Die Klassen der Ontologie sind in zwei Ebenen unterteilt, die *höhere Ontologie*

[4] *Resource Description Framework Schema* (`http://www.w3.org/TR/rdf-schema/`)
[5] *Darpa Agent Markup Language - Ontology Inference Layer* (`http://www.w3.org/TR/daml+oil-reference`)
[6] *Web Ontology Language* (`http://www.w3.org/TR/owl-features/`)

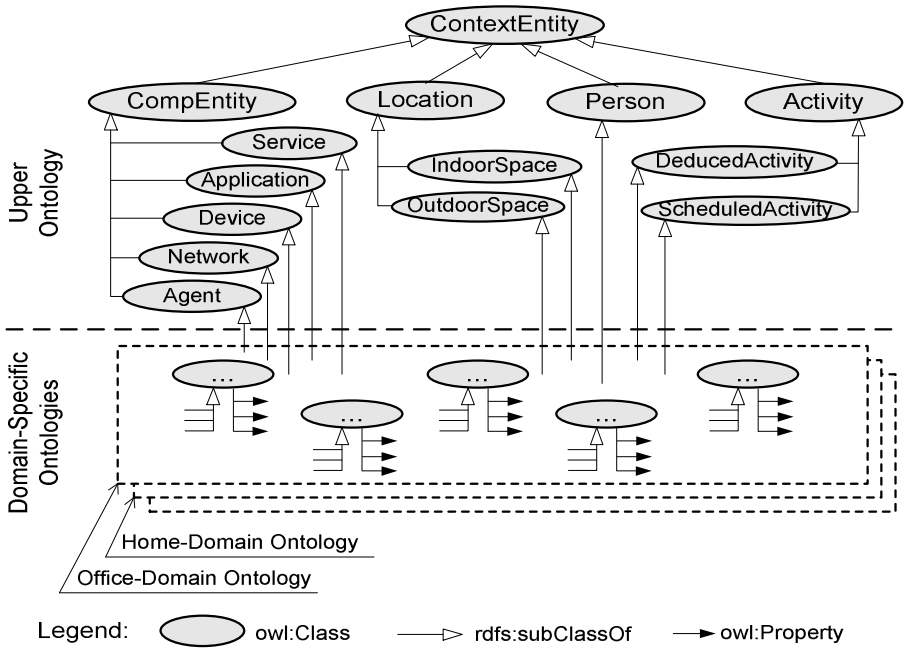

Abbildung 4.3.: Das CONON-Modell (Quelle: [Wang04])

beinhaltet die Grundkonzepte *Person*, *Aktivität*, *Ort* und *Gerät*. Deren Eigenschaften und Beziehungen sowie ihre Unterklassen werden durch die Ontologie modelliert. Die *domänenspezifischen Ontologien* enthalten spezifische Wissensrepräsentationen konkreter Domänen.

4.2.7. Gegenüberstellung

In der Tabelle 4.1 werden die verschiedenen Modellierungsansätze nochmals anhand ihrer *Repräsentationsschemata*, *Komponenten*, *Beispiele* und *Beschreibungen* gegenübergestellt, um im nächsten Kapitel eine Bewertung zu ermöglichen.

4.3. Bewertung der Modellierungsansätze

Nachdem das vorangegangene Kapitel die verschiedenen Modellierungsansätze vorgestellt und gegenübergestellt hat, soll nun eine Beurteilung der Tauglichkeit im Hinblick auf die Zielsetzung dieser Arbeit vorgenommen werden. Um diese Bewertung vornehmen zu können, bedarf es Kriterien als Metrik. In ihren Untersuchungen zu diesem Thema führen Strang et al. [Stra04] sechs Kriterien ein, anhand derer die verschiedenen Ansätze zur Kontextmodellierung bewertet werden können. Diese Kriterien werden wie folgt beschrieben (nach [Weiss07]):

K 1: Verteilte Komposition Die Administration und Komposition eines Kontextmodells und dessen Daten erfolgt verteilt, dass heißt einerseits nicht nur zu verschiedenen Zeitpunkten sondern auch in verschiedenen, teilweise extern ge-

Tabelle 4.1.: Gegenüberstellung der Modellierungsansätze (angelehnt an [Wang04])

Modell	Repräsentation	Komponenten	Beispiele	Beschreibung
Schlüssel-Wert Modelle	Schlüssel-Wert Paare	Schlüssel Werte Umgebungs-variablen	Schilit et al.	Einfache Struktur Keine komplexe Modellierung möglich
Markup-Schema Modelle	Markup-Sprachen (XML, SGML)	Profile - Sitzung - Generisch	CC/PP UAProf SCSP GPM	Strikte hierarchische Struktur Mangelnde Flexibilität
Grafische Modelle	Grafische Objekte (UML, ORM, Vektoren)	UML-Erweiterungen Stereotypen Tags Fakten Vektor-komponenten	ORM-Kontext-erweiterung Vector Space Model	Geeignet für Informationsstrukturen Mangelnde Kontext-instanziierung
Objekt-orientierte Modelle	Objektorientierte Sprachen	Objekte Assoziationen	TAE-Projekt GUIDE-Projekt	Objektorientierte Konzepte Mangelnde Flexibilität
Logik-basierte Modelle	Axiome	Objekte Regeln Annahmen	Kontext-formalisierung Extended Situation Theory	Unterstützt Reasoning Schwer zu implementieren Mangelnde Skalierbarkeit
Ontologie-basierte Modelle	Ontologien (CoOL, RDF, OWL)	Klassen Eigenschaften - Objekt - Datentyp	ASC SOCAM GcOM CoBrA	Breite Abdeckung von Kontexten Unterstützt Reasoning Validierung möglich

legenen Programmkomponenten. Genauso sind Kontextfaktoren oft nicht aus einer Quelle beziehbar, sondern müssen auf Grund von mehreren Datenquellen ermittelt werden.

K 2: Partielle Validierung Kontextinformationen sollten sowohl auf Struktur- als auch auf Instanzebene gegen ein bestehendes Kontextmodell zu validieren sein. Dies ist aufgrund der unterschiedlichen, verteilt und temporal wechselnden Kontextquellen nötig um sicherzustellen, dass die Kontextfaktoren für das Kontextmodell korrekt zu verarbeiten sind.

K 3: Informationsqualität Die Qualität von Kontextinformationen kann über die Zeit gesehen differieren, ebenso können unterschiedliche Sensoren im Laufe der Zeit unterschiedlich reichhaltige Daten liefern. Daher sollten Kontextmodelle Techniken zur Qualitätsbestimmung und Qualitätssicherung unterstützen.

K 4: Unvollständigkeit & Mehrdeutigkeit Da die Daten zur Bestimmung eines Kontextfaktors oft unvollständig oder mehrdeutig sind, sollten im Kontextmodell Mechanismen zur Interpolation vorgesehen sein, durch die unvollständige Daten vervollständigt werden können.

K 5: Formalisierungsgrad Die exakte und nachvollziehbare Beschreibung von Kontextfaktoren und ihrer Beziehungen untereinander ist eine große Herausforderung. Dafür ist es notwenig, ein gemeinsames Verständnis der Terminologie zu besitzen, die in der Beschreibung verwendet wird.

K 6: (Wieder)Verwendbarkeit in anderen Systemen Aus der Perspektive der Implementierung ist eine Wiederverwendbarkeit der Kontextmodelle in neuen sowie bestehenden Systemen anzustreben.

Strang et al. bewerten die Tauglichkeit der Modellierungsansätze im Hinblick auf die Erfüllung der von ihnen aufgestellten Kriterien wie in der Tabelle 4.2 dargestellt.

Tabelle 4.2.: Bewertung der Modellierungsansätze (nach [Stra04])

	K 1: Verteilte Komposition	K 2: Partielle Validierung	K 3: Informationsqualität	K 4: Unvollständigkeit & Mehrdeutigkeit	K 5: Formalisierungsgrad	K 6: (Wieder)Verwendbarkeit
Schlüssel-Wert Modelle	⊖	⊖	⊖⊖	⊖⊖	⊖⊖	⊕
Markup-Schema Modelle	⊕	⊕⊕	⊖	⊖	⊕	⊕⊕
Grafische Modelle	⊖⊖	⊖	⊕	⊖	⊕	⊕
Objektorientierte Modelle	⊕⊕	⊕	⊕	⊕	⊕	⊕
Logikbasierte Modelle	⊕⊕	⊖	⊖	⊖	⊕⊕	⊖⊖
Ontologiebasierte Modelle	⊕⊕	⊕⊕	⊕	⊕	⊕⊕	⊕

⊕: geeignet, ⊕⊕: besonders geeignet, ⊖: ungeeignet, ⊖⊖: besonders ungeeignet

Ontologiebasierte Modelle werden ihren Untersuchungen zur Folge im Bereich der ubiquitären Systeme als vielversprechender Ansatz zur Modellierung und Abbildung von Kontextinformationen eingestuft. Ontologien bieten durch die Kombination aus logikbasierten und objektorientierte Prinzipien und durch ihre universell einsetzbaren Möglichkeiten der Wissensmodellierung in diesem Bereich eine große Flexibilität.

Aus diesem Grund soll die semantische Kontextmodellierung, die Ontologien als Kontextmodelle verwendet, nachfolgend genauer untersucht werden. Zunächst wird

das Prinzip des *Context Reasoning* behandelt. Im Anschluss daran wird erläutert, inwieweit sich semantische Technologien im Umfeld mobiler Nutzungsszenarien einsetzen lassen.

4.4. Semantische Kontextmodellierung

In vergangener Zeit wurden zahlreiche kontextbezogene Systeme, auf Basis von Techniken des Semantic Web wie Ontologien, RDF und OWL entwickelt. Beispiele dafür sind *CoBrA* [CheFi04], *Gaia* [Rang03] oder *SOCAM* [Gu05]. Diese Systeme verwenden ontologiebasierte Kontextmodelle und unterstützen Abfrageausdrücke und logisches *Context Reasoning*[7]. Typischerweise werden Standard-Werkzeuge wie Jena[8] und XSB[9] eingesetzt, um die Syntax der ontologiebasierten Daten zu analysieren und das Reasoning auszuführen [Gu07].

Semantische Modellierungssprachen erlauben die Beschreibung des Kontextes, insbesondere für Metadaten, Eigenschaften von Benutzern, Geräten und Ressourcen auf einer hohen Abstraktionsebene. Besonders in mobilen Umgebungen ist dies von Vorteil, da sich Benutzer und Geräte durch unterschiedliche Umfelder bewegen und die Verfügbarkeit von Ressourcen und der Zugriff auf Dienste von hoher Dynamik und Unvorhersehbarkeit abhängen. Die Anwendung semantischer Technologien zur Spezifikation und Verwaltung von Metadaten stellt sicher, dass beim Austausch von Kontextinformationen ein gemeinsames Verständnis zwischen den verschiedenen Entitäten existiert [Corr07].

4.4.1. Context Reasoning

Context Reasoning beschreibt die automatische Ableitung impliziter Fakten aus explizit definierten Kontextinformationen [Ay07]. Dabei werden zwei Ziele verfolgt: die Überprüfung der Konsistenz von Kontextinformationen und die Ableitung von impliziten *High Level*-Kontexten als expliziten *Low Level*-Kontexten [Wang04].

Einige Kontextinformationen, die für kontextsensitive Applikationen von Interesse sind, können nicht unmittelbar erhoben werden, sondern lassen sich nur durch Ableitung aus anderen Informationen ermitteln. Die Informationen beispielsweise darüber, ob eine Person in Bewegung ist und welches Transportmittel sie verwendet, kann nicht direkt wahrgenommen sondern muss aus anderen Kontextinformationen wie GSM-Zelle, WiFi-Zugangspunkt, Kalenderinformationen usw. abgeleitet werden. Diese Folgerungen von Kontextinformationen höherer Ebenen sowie Vorhersagen über zukünftige Werte von Kontextattributen fallen in das Gebiet des *Context Reasoning* [Sin06].

[7] Siehe Kapitel 4.4.1: Context Reasoning
[8] *Jena Semantic Web Framework for Java* (`http://jena.sourceforge.net/`)
[9] `http://www.xsb.com/`

Wang et al. [Wang04] teilen die Techniken des *Context Reasoning* in zwei Kategorien auf, das *Ontology Reasoning* und das *benutzerdefinierte Reasoning*. Van Sinderen et al. [Sin06] beschreiben darüber hinaus die Technik des *maschinellen Lernens*.

Ontology Reasoning umfasst die Ableitung neuer Fakten auf Basis einer vorhandenen Wissensbasis aus spezifizierten Fakten und einer Ontologie, die Klassen und ihre Relationen einer Anwendungsdomäne formal beschreibt.

Techniken des *maschinellen Lernens* (z.B. bayes'sche Netze, künstliche neuronale Netze oder Entscheidungsbäume) können zur Konstruktion von Modellen mit Kontextattributen hörerer Ebenen und ihrer qualitativen Eigenschaften[10] aus Kontextattributen niedriger Ebenen eingesetzt werden. Abbildung 4.4 veranschaulicht ein Beispiel für *Context Reasoning* aus [Korp03]. In diesem wird durch Bayes-Klassifikation[11] aus so genannten Kontext-Atomen auf Kontexte zweiter Ebene – in diesem Fall Informationen über das das Gebäude – geschlossen.

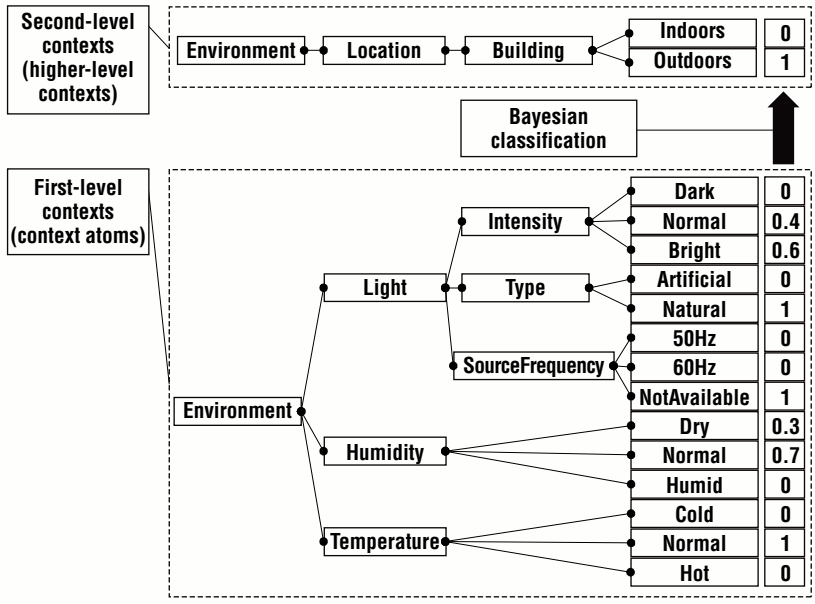

Abbildung 4.4.: Context Reasoning durch Bayes-Klassifikation (Quelle: [Korp03])

Eine flexiblere Technik des *Context Reasoning* stellt das *benutzerdefinierte Reasoning* dar. Über benutzerdefinierte Regeln lässt sich unter Anwendung von Logik erster Ordnung ein breites Spektrum von *High Level*-Kontexten aus relevanten *Low Level*-Kontexten herleiten [Wang04]. Die Tabelle 4.3 zeigt einige Beispiele dieser Regeln (nach [Wang04]).

Das *Context Reasoning* stellt im Hinblick auf Systeme zur Repräsentation und Verwaltung von Kontextinformationen eine Schlüsselfunktionalität dar. Wie die vorangegangenen Kapitel gezeigt haben, bietet sich zu deren Realisierung die Verwen-

[10] Siehe Kapitel 2.4: Quality of Context

[11] Mittels des naiven Bayes-Klassifikators ist es möglich, die Zugehörigkeit eines Objektes zu einer Klasse zu bestimmen. Er basiert auf dem bayes'schen Theorem.

Tabelle 4.3.: Benutzerdefinierte Context Reasoning-Regeln (nach [Wang04])

	Reasoning-Regeln
Schlafen	(?u locatedIn Bedroom) ∧ (Bedroom lightLevel LOW) ∧ (Bedroom drapeStatus CLOSED) ⇒ (?u situation SLEEPING)
Duschen	(?u locatedIn Bathroom) ∧ (WaterHeater locatedIn Bathroom) ∧ (Bathroom doorStatus CLOSED) ∧ (WaterHeater status ON) ⇒ (?u situation SHOWERING)
Kochen	(?u locatedIn Kitchen) ∧ (ElectricOven locatedIn Kitchen) ∧ (ElectricOven status ON) ⇒ (?u situation COOKING)
Fernsehen	(?u locatedIn LivingRoom) ∧ (TVSet locatedIn LivingRoom) ∧ (TVSet status ON) ⇒ (?u situation WATCHINGTV)
Abendessen	(?u locatedIn DiningRoom) ∧ (?u owl:differentFrom ?v) ⇒ (?u situation HAVINGDINNER)

dung eines semantischen Ansatzes an.

Der Einsatz semantischer Technologien ist jedoch im Kontext mobiler Nutzungsszenarien nicht trivial. Nachfolgend wird darauf eingegangen, inwieweit semantische Technologien auf mobilen Geräten eingesetzt werden können.

4.4.2. Mobiler Einsatz semantischer Technologien

Trotz vielversprechender Eigenschaften semantischer Technologien muss berücksichtigt werden, dass der hohe Grad an Heretogenität und Dynamik in mobilen Umgebungen den Einsatz von semantischen kontextbezogenen Systemen verkompliziert. Darüber hinaus haben mobile Geräte begrenzte Fähigkeiten bezüglich Prozessorleistung, Speicher und Batterielaufzeit und sind typischerweise ungeeignet, traditionelle semantikbasierte Dienste zu betreiben, die für stationäre Netzwerke entwickelt wurden. Dienste zur Unterstützung von Semantik wie *Ontology-Repositories*, *Inference-Engines* und Wissensmanagement-Tools benötigen üblicherweise eine große Menge an Berechnungs- und Speicherressourcen, die sich nicht mit den Eigenschaften mobiler Geräte vereinen lassen. Es gibt also strenge Limitationen für semantikunterstützende Systeme, die auf ressourcenbeschränkten Geräten betrieben werden können [Corr07].

Jüngste Fortschritte in der Entwicklung mobiler Geräte haben Handhelds und Smartphones hervorgebracht, deren Leistungsfähigkeit sich derer stationärer Sys-

teme annähert. Aufgrund dieser Tatsache gibt es zwar bereits Bestrebungen, die rechenintensiven semantische Technologien und Algorithmen auf portablen Geräten zu betreiben, die jedoch zum jetzigen Forschungsstand nicht praktikabel sind. Um die Charakteristika mobiler Umgebungen zu berücksichtigen, müssen diverse Rahmenbedingungen beachtet werden, wie der hohe Grad an Verteiltheit und Heterogenität der involvierten Geräte und Nutzer.

Ein möglicher Ansatz, um Reasoning zu unterstützen, ist der Einsatz eines separaten Servers, der über ein Netzwerk mit den mobilen Teilnehmern verbunden ist [Corr07]. Eine Untersuchung dieses Ansatzes soll im weiteren Verlauf dieser Arbeit durchgeführt werden.

4.5. Zusammenfassung

Um sich der Beantwortung der Forschungsfrage zu nähern, hat sich dieses Kapitel zunächst mit Fragen der Modellierung des Kontextes befasst. Unter den vorgestellten Ansätzen zur Kontextmodellierung wurde der ontologiebasierte, semantische Ansatz für am geeignetsten befunden, um eine möglichst generische und flexible Konzeption für ein System zur Kontextverwaltung zu entwickeln.

Nach einer genaueren Betrachtung der semantischen Modellierung stellte sich heraus, dass ontologiebasierte Ansätze sowohl automatisches als auch manuelles *Context Reasoning* möglich machen. Um semantische Technologien trotz Limitationen portabler Hardware in mobilen Umgebungen einsetzen zu können, wird eine zentrale Architektur in Erwägung gezogen, um das *Context Reasoning* serverseitig zu ermöglichen.

Das nachfolgende Kapitel befasst sich nun, auf diese Erkenntnisse aufbauend, mit der nächsten Phase der Bearbeitung des Themas dieser Arbeit. Diese besteht in der Untersuchung von Rahmenwerken für mobile kontextsensitive Applikationen, welche Kontextinformationen repräsentieren und verwalten.

5. Ein Rahmenwerk für mobile kontextsensitive Applikationen

Das vorangegangene Kapitel 4 hat Fragen zur Modellierung und Repräsentation von Kontextinformationen behandelt. Neben der grundlegenden Frage nach der Abbildbarkeit von Kontexten durch Modelle wurden konkrete Ansätze zur Modellierung erläutert und bewertet. Die semantische Kontextmodellierung besitzt demnach das größte Potenzial in Bezug auf Flexibilität der Abbildung von Kontextinformationen.

Dieses Kapitel beschäftigt sich nun mit Systemen, die die Kontextverwaltung und die Kontextrepräsentation ermöglichen sollen. Zunächst wird erläutert, aus welchen Gründen ein Bedarf an solchen Systemen besteht und welche Merkmale sie charakterisieren. Fragen der Architektur dieser Systeme werden im Anschluss daran gestellt. Weiterhin werden Anforderungen zusammengetragen, die sie erfüllen sollten. Darüber hinaus werden einige Ansätze aus der Forschung vorgestellt und in Bezug auf die Zielsetzung dieser Arbeit untersucht.

Der zweite Schritt besteht in der Planung und Konzeption eines eigenen, generischen Rahmenwerks, wobei die zuvor gewonnenen Erkenntnisse einbezogen werden. Das Konzept beinhaltet eine detaillierte Darstellung der Systemarchitektur mit ihren Modulen sowie eine Definition der Schnittstellen zwischen diesen Komponenten.

5.1. Gründe für Systeme zur Kontextverwaltung

Für die Entwicklung von Systemem zur Kontextverwaltung gibt es zahlreiche Gründe. Diese werden nachfolgend dargelegt, und mit Aussagen von Autoren gestützt, die sich in ihren Arbeiten mit der Entwicklung dieser Rahmenwerke befasst haben.

Jacob E. Bardram, Autor des *Java Context Awareness Frameworks (JCAF)*, schreibt in seiner Arbeit, ein häufiges Ziel für die Entwicklung von Rahmenwerken für kontextsensitive Anwendungen sei die Ermöglichung einer einfachen Entwicklung und Verbreitung kontextsensitiver Applikationen [Bard05]. Entwickler dieser Applikationen können sich demnach durch solche Systeme auf den Kontext in Bezug auf die Funktionalität ihrer eigenen Applikationen konzentrieren und sich dabei auf eine Basis-Infrastruktur verlassen, die diese Informationen verwaltet und bereitstellt.

Pierre-Charles David, Entwickler des Kontextsystems *WildCAT* [David05], detailliert diese Aussage, indem er verdeutlicht, dass der Code um den Kontext zu erfassen auf tiefe Ebenen des Betriebssystems oder auch direkt auf die Hardware zugreift. David hält es für unnötig, dass Entwickler jeder Applikation diesen Code

integrieren müssen, da er für viele Anwendungen identisch ist. Statt dessen sollte die Kontextermittlung den Gegebenheiten der Kontextquellen entsprechend durch das System implementiert und den Entwicklern eine simple und einheitliche Schnittstelle zur Verfügung gestellt werden.

Die Auswahl an Kontextinformationen, von denen Applikationen profitieren können, wird häufig von den Mechanismen zur Kontexterfassung limitiert, die die jeweilige Hard- und Software zur Verfügung stellt. Ein *Bottom-Up*-Ansatz verhindert die Flexibilität und Erweiterbarkeit der Applikationen, da er die Details und die Komplexität der Sensoren auf die Ebene der Applikationslogik bringt. Konkrete Probleme gibt es dann, wenn neue Kontexte eingeführt werden sollen, wenn die Verbindung zwischen Kontexten und Verhalten geändert werden muss oder wenn neues Verhalten in Bezug auf den Kontext hinzugefügt werden soll [Du08].

Entwickler werden daran gehindert, neue und innovative kontextsensitive Anwendungen zu entwickeln, wenn ihre Kreativität von den zur Verfügung stehenden Sensoren begrenzt wird [Bald06]. Durch die immer größer werdende Vielfalt von durch Sensorsysteme wahrnehmbaren Informationen wird die Entwicklung kontextbezogener Systeme immer aufwändiger und komplizierter, wie Jason Pascoe in seiner Arbeit [Pasc98] prägnant beschreibt:

> *"It is observed that the sheer diversity of exploitable contexts and the plethora of sensing technologies are actually working against the deployment of context-aware systems"* [Pasc98]

Ein letzter wichtiger Grund ist die Interoperabilität zwischen kontextsensitiven Applikationen und den zu Grunde liegenden Kontextquellen. Ein System, welches Kontextinformationen verwaltet, macht eine Generalisierung möglich und erlaubt Applikationen, Kontextinformationen untereinander auszutauschen. Dafür ist es notwendig, die Applikation und die Komponenten, die den Kontext ermitteln, voneinander zu entkoppeln [Chen00].

5.2. Architekturfragen

Nachdem die Gründe für Systeme zur Kontextverwaltung dargelegt wurden, wird nun auf Fragen der Architektur dieser Systeme eingegangen.

Chen unterscheidet in seiner Arbeit über seine Broker-Architektur für kontextsensitive Systeme [Chen04] drei verschiedene Methoden der Bereitstellung von Kontextinformationen für mobile Applikationen, die er unter dem Begriff *Context Acquisition* zusammenfasst. Die erste Methode, *Direct Sensor Access*, bezeichnet die direkte Erfassung und Bereitstellung von *Low Level*-Kontextinformationen mit Hilfe von Hardware-Sensoren, die im mobilen Gerät integriert sind. Mobile Anwendungen beziehen dabei die benötigten Sensordaten auf niedriger Ebene durch die Nutzung einer gerätsspezifischen API. Die zweite Methode, die *Middleware Infrastructure*, stellt

eine Architektur bereit, die die technischen Details sensorbezogener Daten verbirgt und höherwertige Kontextinformationen für eine mobile Anwendung liefert. Weiterhin erlaubt eine solche Infrastruktur die Observation von Kontextinformationen, um Anwendungen die Möglichkeit zu geben auf Kontextänderungen zu reagieren. Die dritte Methode nach Chen ist die Nutzung eines entfernten *Context Servers*, der die Umgebung mobiler Clients beobachtet und auf deren Anfrage Kontextinformationen liefern kann [Tur06].

Da der direkte Zugriff von Applikationen auf Sensordaten, wie bereits beschrieben, auf ein verwaltendes System verlagert werden soll, wird die Methode des *Direct Sensor Access* an dieser Stelle nicht weiter verfolgt. In den nachfolgenden Abschnitten sollen Middleware-Infrastrukturen und entfernte Kontextserver genauer betrachtet werden.

5.2.1. Middleware-Infrastrukturen

Um die Entkopplung von Applikationen und Kontextquellen zu ermöglichen, beschreiben einige Autoren Abstraktionsebenen, die zwischen mobilen Applikationen und den Quellen der Kontextinformationen – also unter anderem der Sensoren – angesiedelt sind. Diese werden oft als Middleware-Schichten bezeichnet. Chen et al. [Chen00] schreiben in ihrer Untersuchung zum Forschungsstand kontextsensitiver Systeme:

> *„To separate the low-level sensor data processing from high-level applications, it is necessary to introduce a middleware layer whose functionalities are collecting raw sensor information, translating it to an application-understandable format, and disseminating it to interested applications.“* [Chen00]

Demnach sollte diese Abstraktionsebene die Rohdaten von Sensoren erfassen, sie verarbeiten und den Applikationen in verständlicher Art und Weise zur Verfügung stellen bzw. an sie weiter geben. In der Abbildung 5.1 wurde die zwischen Applikationen und Sensoren gelegene Ebene, die eine Infrastruktur zur Kontextverwaltung darstellt, visualisiert.

Diese zusätzliche Ebene löst zugleich mehrere Probleme, die im vorherigen Abschnitt dargelegt wurden. Zunächst werden die Sensoren für die Applikationen abstrahiert. Applikationen greifen also nicht mehr unmittelbar über die Sensoren, sondern über die Verwaltungskomponente auf die aufbereiteten Kontextdaten zu. Weiterhin sind die Applikationen durch das System implizit miteinander verbunden, was einen Austausch von Kontextdaten möglich macht. Letzendlich fordert die Anbindung an die Applikationen und die Sensoren ein einheitliches Format, in welchem die Daten ausgetauscht werden. Es gibt also standardisierte Schnittstellen zwischen Applikationen und Kontextverwaltung sowie zwischen Kontextverwaltung und Sensoren (in der Abbildung 5.1 als dünne Linien dargestellt).

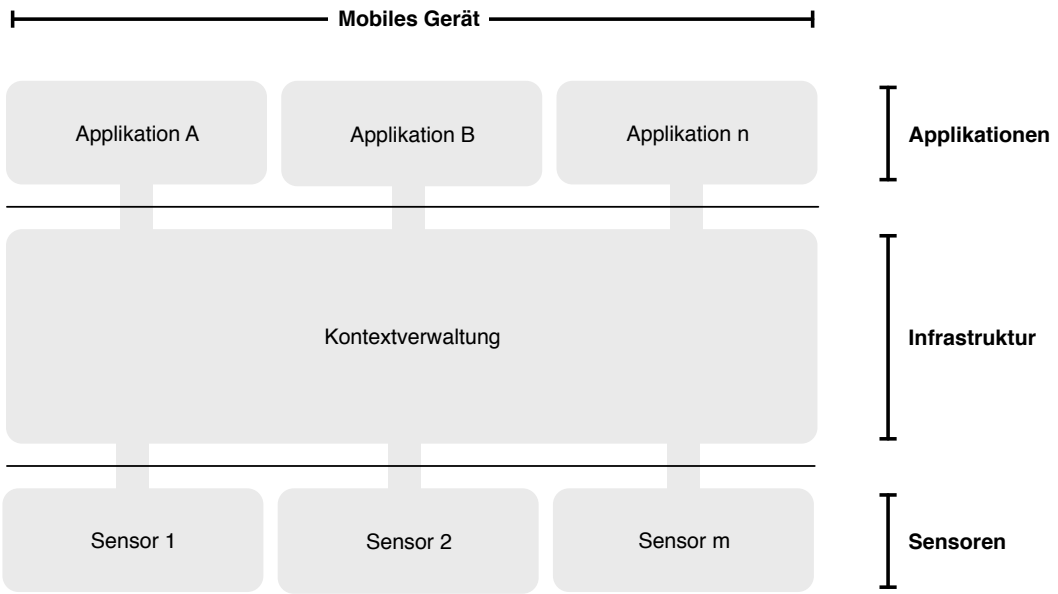

Abbildung 5.1.: Kontextverwaltung als Abstraktionsebene

In Bezug auf die Zielsetzung dieser Arbeit stellt die Einführung einer Software-Schicht zur Verwaltung und Bereitstellung von Kontextinformationen einen ersten wichtigen Schritt dar. Bevor die Konzeption jedoch weitergeführt wird, soll ein Blick auf das Prinzip der entfernten Kontextserver gerichtet werden.

5.2.2. Entfernte Kontextserver

Die Vorteile der in Abbildung 5.1 skizzierten Architektur, Kontextdaten zwischen mehreren Applikationen und mehreren Kontextquellen austauschen zu können, beziehen sich ausschließlich auf mobile Applikationen innerhalb eines mobilen Endgeräts. Ein wichtiger Faktor ist jedoch auch der geräteübergreifende Austausch von Kontextinformationen. Einerseits können damit viele Nutzer von den Kontextinformationen profitieren, andererseits entstehen gerade durch die Interaktion zwischen mehreren Geräten wieder neue Kontexte. Der zweite wichtige Faktor ist die Frage nach der Leistungsfähigkeit des kontextrepräsentierenden Systems. Aus diesem Grund beschreiben zahlreiche Autoren zentrale und verteilte Architekturen, bei denen das Kontextwissen über entfernte Kontextserver verwaltet und bereitgestellt wird.

Der einfachste Weg ist es, *zentrale* Kontextserver für diese Aufgabe einzusetzen. Das *Mobile Application Customization System* von Schilit et al. [Schi93] verwendet diesen Ansatz. Ein solcher Server verwaltet eine bestimmte Menge von Kontextvariablen und stellt sie angemeldeten Clients zur Verfügung. Dabei existieren typischerweise mehrere Umgebungen – eine für jeden Benutzer – sowie gemeinsame Umgebungen. Ähnliche Ansätze verfolgen die Systeme *SitComp (Situated Computing)* [Hull97] und *CIS (Contextual Information Service)* [Pasc98]. Diesen Ansätzen

gemein ist die Interpretation der Kontextinformationen und die Bereitstellung dieser Informationen für die mobilen Applikationen über standardisierte Schnittstellen.

Wie viele zentralisierten Systeme birgt auch dieser Ansatz das Problem der Skalierbarkeit. Anstatt alle Kontextinformationen zentral vorzuhalten, ermöglicht eine *verteilte* Architektur, den Kontext an mehreren Orten zu speichern und dadurch Engstellen zu vermeiden. Das *Rome System* [Hua99] beispielsweise, welches an der Stanford University entwickelt wurde, setzt auf eine solche verteilte Architektur. [Chen00]

Für die Konzeption des eigenen Systems stellt ein entfernter Kontextserver als Dienstanbieter eine einfache Möglichkeit dar, ein semantisches Kontextmodell zu verwenden, welches auch bei großen Mengen gespeicherter Kontextdaten und vielen Benutzern effizient bleibt. Dieser Ansatz soll daher genau wie das Konzept der Middleware-Schichten in die weitere Entwicklung des angestrebten Systems einfließen.

5.3. Anforderungen an ein generisches Rahmenwerk

Aus den bisherigen Erkenntnissen werden in diesem Abschnitt Anforderungen an ein generisches Rahmenwerk zur Kontextrepräsentation zusammengestellt. Diese sollen anschließend in der Konzeption Berücksichtigung finden.

A 1: Verwaltung von Kontextinformationen Wie bereits in der Begründung dargelegt, hat das System die primäre Aufgabe, Kontextinformationen für mobile Applikationen bereitzustellen. Dabei soll eine Abstraktion vorgenommen werden, die die Details der Kontextermittlung und -verarbeitung vor den Applikationen verbirgt.

Der Zugriff auf die Kontextinformationen soll sowohl über explizites Abfragen des Kontextdienstes *(Pull-Mode)* als auch über eine Möglichkeit der Benachrichtigung *(Push-Mode)* erfolgen können. Darüber hinaus müssen mobile Applikationen in die Lage versetzt werden, neue Kontexte in das System einzufügen, zu verändern und zu entfernen.

A 2: Austausch von Kontextinformationen Kontextinformationen sollen sowohl zwischen mobilen Applikationen als auch zwischen mehreren Nutzern des Systems geteilt bzw. ausgetauscht werden können.

A 3: Trennung von Kontextmanagement und Kontextmodell Kontextmodell und Kontextverwaltung sollen konzeptionell voneinander getrennt werden. Das Kontextmodell beinhaltet das eigentliche Kontextwissen und bildet somit die Beziehungen zwischen den elementaren Kontextinformationen ab. Die Verwaltung des Kontextmodells enthält modellspezifische Funktionalitäten. Beide dieser Komponenten sollten austauschbar sein.

A 4: Erweiterbarkeit und Skalierbarkeit Das Rahmenwerk soll durch seine Modularität und durch seine Schnittstellen so konzipiert sein, dass es sich jederzeit um neue Kontextquellen erweitern lässt. Zum einen wird dadurch der fortschreitenden Entwicklung neuer Sensortechniken und Dienste Rechnung getragen, zum anderen werden diese Komponenten somit wiederverwendbar.
Das System soll aufgrund seines generischen Charakters seine Informationen in einer Weise zur Verfügung stellen können, die unabhängig von konkreten Kontextinformationen und ihrer Darstellung ist. Dadurch soll es möglich sein, beliebige Kontexte und beliebige Kontexthierarchien abzubilden.

A 5: Kompaktheit und Portierbarkeit Das System sollte so kompakt gestaltet sein, dass es sich einfach implementieren und portieren lässt. Das ist aufgrund der verschiedenen, existierenden mobilen Plattformen sowie der stetigen Weiterentwicklung mobiler Hard- und Software wichtig.

A 6: Sicherheit und Privatsphäre Kontextinformationen stellen oft sensible personenbezogene Daten dar. Deshalb ist eine möglichst umfassende Sicherung dieser Daten anzustreben. Persönliche Kontextinformationen sollten also vor unberechtigtem Zugriff durch Dritte sowie durch andere Nutzer des Systems geschützt sein.

A 7: Berücksichtigung der Kontextqualität Insbesondere aus Sensoren ermittelte Kontextinformationen können ungenau, unvollständig oder widersprüchlich sein [Baus02]. Wie bereits in Kapitel 2.4 beschrieben, sind Informationen, die die Qualität von Kontextinformationen beschreiben, wichtige Metadaten. Das System sollte diese Angaben zur Qualität der Kontextinformationen berücksichtigen.

5.4. Bestehende Rahmenwerke und Systeme

Nachdem bis zu dieser Stelle die grundlegende Fragen zu Architektur und Anforderungen zusammengetragen wurden, befasst sich dieses Kapitel nun mit der Darstellung bereits existierender Ansätze für Rahmenwerke und Systeme zur Kontextrepräsentation. Dies dient dem Zweck, einen Überblick über bestehende Systeme zu vermitteln und die jeweiligen Umsetzungen anhand ihrer Vor- und Nachteile sowie hinsichtlich der Zielsetzung dieser Arbeit einzuordnen.

Da die Forschungen in diesem Bereich zu zahlreichen Ansätzen geführt haben, die im Rahmen dieser Arbeit nicht alle im Detail wiedergegeben werden können, wurde eine repräsentative Auswahl in Form von drei Rahmenwerken getroffen. Diese basiert auf den Untersuchungen von [Bald06], [Sing06] und [Kapp03]. Die Ausführungen zu diesen Systemen erfolgen in Anlehnung an [Weiss07].

5.4.1. Context Toolkit von Dey et al.

Die Zielsetzung der Entwicklung des *Context Toolkit Frameworks* [Salb99], [Dey01] bestand darin, ein umfangreiches und anwendungsunabhängiges, konzeptuelles Rahmenwerk für die Repräsentation und Verarbeitung von Kontextinformationen bereitzustellen.

Das Rahmenwerk verwendet so genannte *Context Widgets*, welche über eine verteilte Infrastruktur verfügbar sind. Diese stellen Softwarekomponenten dar, die Kontextinformationen liefern und dabei den Vorgang der eigentlichen Erfassung kapseln. Die Quellen dieser Kontextinformationen können verschiedener Art sein, beispielsweise Sensoren, Geräteinformationen oder Netzwerkinformationen. Die Idee dieser *Context Widgets* entstand aus den Widgets grafischer Benutzungsschnittstellen (GUIs), welche ebenfalls als Teile der Benutzeroberfläche von der Anwendung gelöst betrachtet und somit wiederverwendet werden können.

Zur Erfassung des Kontextes stellt das System zwei Mechanismen der Abstraktion zur Verfügung. Die *Aggregatoren* sammeln und speichern logisch zusammengehörige Kontexte verschiedener Widgets und die *Interpreter* ermitteln übergeordnete Kontexte aus Widgets sowie Aggregatoren. Durch das Konzept der Widgets und der Aggretatoren bzw. Interpreter wird eine explizite Trennung zwischen physischem und logischem Kontext vorgenommen.

Ein Beispiel der Konfiguration der verschiedenen Komponenten des Context Toolkits ist in Abbildung 5.2 dargestellt.

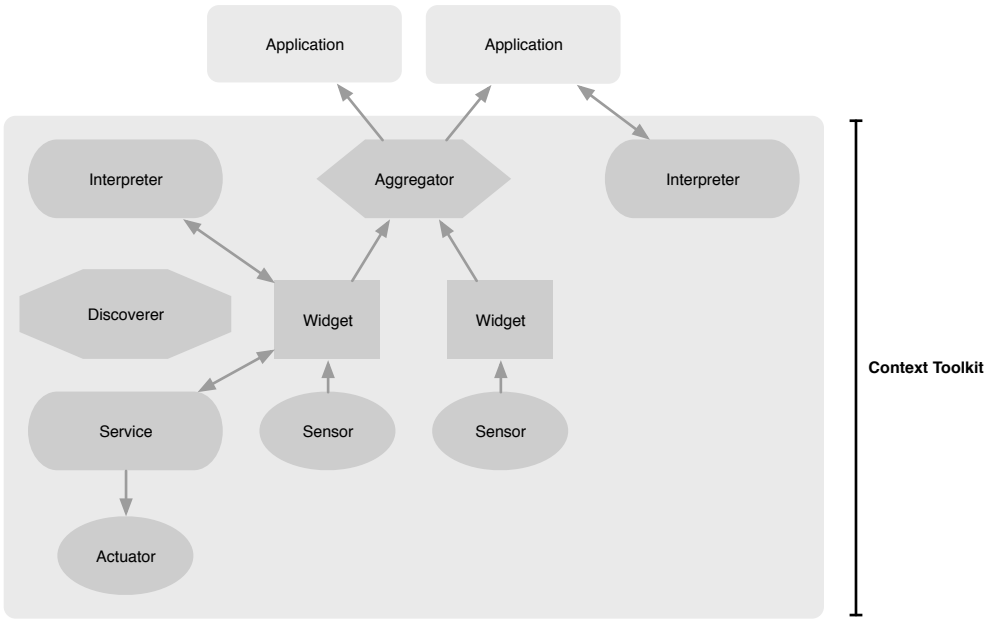

Abbildung 5.2.: Beispiel einer Konfiguration des Context Toolkits (nach [Salb99])

Zur Adaption des Kontextes verfügt das Rahmenwerk über so genannte *Context Services*, welche die Ausführung der Adaptionsvorgänge für die Applikationen

übernehmen. Diese werden in drei Kategorien unterteilt, die *Recommendations* (zur Anzeige möglicher Aktionen auf Basis des Kontextes), die *Trigger* (zum automatischen Auslösen von Aktionen) und die *Tagger* (zum Anreichern von Objektdaten mit Kontextinformationen zur späteren Verwendung). Die Ausführung der kontextbezogenen Adaptionsvorgänge führen so genannte *Aktuatoren* durch.

Alle Komponenten des Rahmenwerks sind in einer Registrierung, dem *Discoverer*, verzeichnet, über welchen der Zugriff auf das Context Toolkit mittels *Lookup-Service* erfolgt. [Weiss07], [Kapp03], [Sing06]

Das Context Toolkit verwendet das Modell der Schlüssel-Wert-Paare[1] zur Modellierung und Repräsentation der Kontexte. Diese Tatsache schränkt die Möglichkeiten der differenzierten Strukturierung von Kontextvariablen ein [Stra04] und ist daher als Rahmenwerk unter Berücksichtigung der Zielsetzung dieser Arbeit in Bezug auf die Flexibilität der möglichen abbildbaren Kontexte als ungeeignet einzustufen. Das Grundprinzip der Context Widgets zur Kapselung der Sensoren soll jedoch weiter verfolgt werden.

5.4.2. Hydrogen Framework von Hofer et al.

Das *Hydrogen Framework* von Hofer et al. [Hof03] ist für die reine Anwendung auf mobilen Endgeräten konzipiert worden und besitzt keinerlei zentrale Komponente zur Kontextverwaltung. Es basiert auf einem objektorientierten Kontextmodell. Die Repräsentation des Kontextes sowie sämtliche Aktionen finden innerhalb der mobilen Geräte statt. Zur Abstraktion der Sensoren setzt dieses Modell eine Middleware-Infrastruktur[2] ein. Die Abbildung 5.3 zeigt einen Überblick über die Architektur des Hydrogen Frameworks.

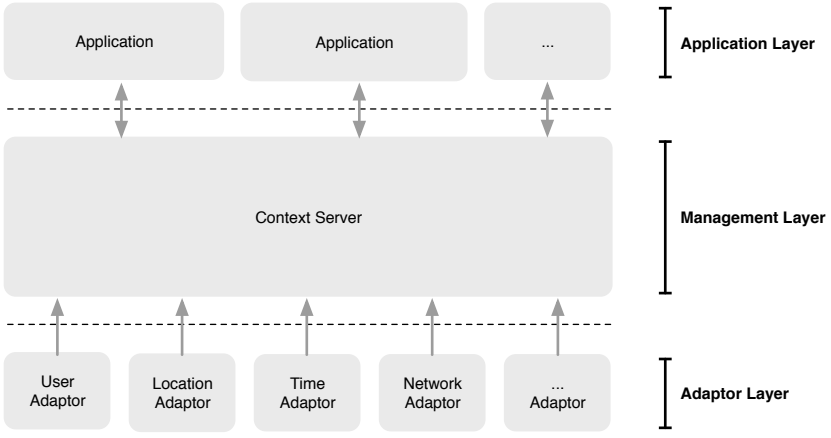

Abbildung 5.3.: Architektur des Hydrogen Frameworks (nach [Hof03])

Die unterste Ebene, der *Adaptor Layer*, ist für die Ermittlung physischer Kontextdaten auf Basis der Sensoren zuständig. Die Adaptoren können die Kontextdaten

[1] Siehe Kapitel 4.2.1: Schlüssel-Wert Modelle
[2] Siehe Kapitel 5.2.1: Middleware-Infrastrukturen

jedoch bis hin zu logischen Kontexten anreichern. Der darüber liegende *Management Layer* beinhaltet den *Context Server*, welcher sämtliche Kontextinformationen speichert, die den in der *Application Layer* angesiedelten Anwendungen zur Verfügung stehen.

Hydrogen unterscheidet zwischen zwei Arten des Kontextes. Der *Local Context* beschreibt Kontextinformationen des eigenen Geräts und der *Remote Context* bezeichnet Kontextinformationen anderer Geräte. *Context Sharing* bezeichnet den Austausch von Kontextinformationen zwischen den Endgeräten. Dieser wird vom *Context Server* gesteuert. [Weiss07]

Das Hydrogen Framework beinhaltet ausschließlich Komponenten, die auf der mobilen Hardware ablaufen. Der Austausch von Kontextinformationen erfolgt adhoc, also unmittelbar zwischen den Endgeräten. Die gemeinsame Nutzung einer zentralen Kontextbasis ist somit nicht möglich. Ebenso sind komplexe Reasoning-Algorithmen aufgrund der begrenzten Ressourcen bei diesem Ansatz nicht anwendbar. Das Konzept des *Adaptor Layer*, dessen Adaptoren jeweils Kontextinformationen einer Domäne erheben und verarbeiten, soll bei den weiteren Untersuchungen im Fokus der Betrachtung bleiben.

5.4.3. Context Framework von Korpipää et al.

Das *Context Framework* von Korpipää et al. [Korp03], [Korp05] setzt auf einen *Blackboard*[3]-basierten Ansatz. Eine schematische Darstellung des Rahmenwerks zeigt die Abbildung 5.4.

Der *Context Manager* hat die Aufgabe Kontextdaten zu speichern, Anfragen zu empfangen, Abonnements zu verwalten und Antworten auf Kontextanfragen von Komponenten, Applikationen oder *Application Controller* zu beantworten. Die Nutzung von Kontextinformationen kann durch die Applikationen selbst oder durch die Steuerung durch den *Application Controller* erfolgen. Dieser wiederum lässt sich zur Laufzeit über den *Customizer* den gegebenen Anforderungen anpassen.

Der Zugriff auf den Kontext erfolgt über den zentralen *Context Manager Blackboard Server*. In diesem laufen die Kontextdaten der einzelnen Komponenten zusammen und werden dort ausgewertet. Applikationen können die Daten aus der Datenbank abfragen, können sich zu Benachrichtigungsdiensten anmelden (zum Beispiel für Benachrichtigungen bei definierten Kontextänderungen) oder können *High Level*-Kontexte über so genannte *Recognition Services* ermitteln.

Eine Besonderheit dieses Ansatzes ist, dass bei ungenauen Sensordaten eine spezielle *Fuzzy Logic*[4] eingesetzt wird, um entsprechende logische Kontexte zu ermitteln.

[3] Das Zentrum des *Blackboard-Architekturmodells* bildet ein zentrales *Message-Board* auf welchem diverse Komponenten, wie Dienste oder Anwendungen, Nachrichten eintragen und abonnieren können [Weiss07].

[4] *Fuzzylogik* ist eine Theorie, welche vor allem für die Modellierung von Unsicherheiten und Unschärfen von umgangssprachlichen Beschreibungen entwickelt wurde. Sie ist eine Verallgemeinerung der zweiwertigen *Booleschen Logik*.

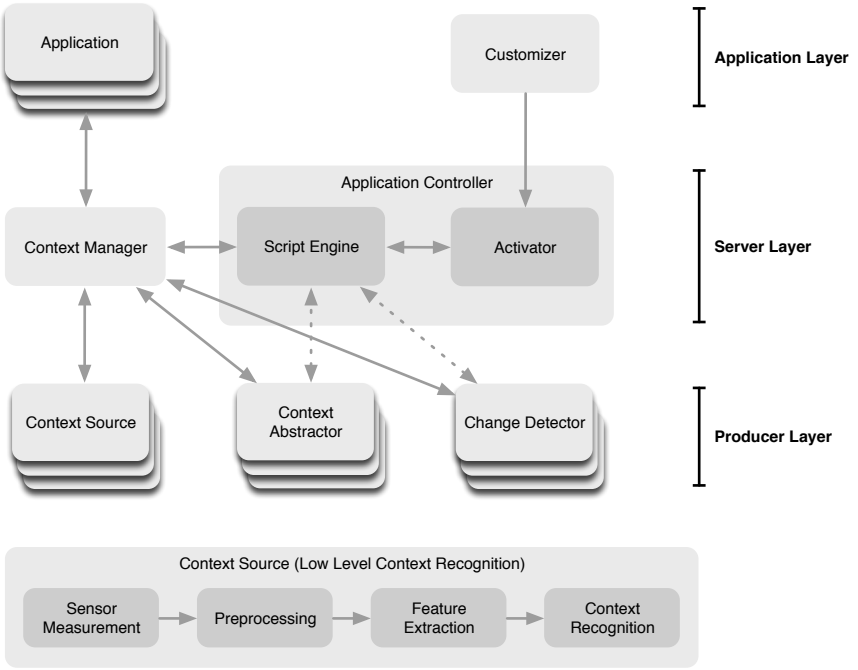

Abbildung 5.4.: Schema des Context Frameworks (nach [Korp05])

[Weiss07], [Sing06], [Bald06]

Das vorgestellte Context Framework ermöglicht durch seine umfangreiche Programmierschnittstelle weitreichende Konfigurationsmöglichkeiten. Da es im Hinblick auf vielfältige Einsatzmöglichkeiten konzipiert wurde, kann der in Kapitel 5.3 aufgestellten Anforderung nach Kompaktheit jedoch nicht entsprochen werden. Die einzelnen Komponenten enthalten komplexe Logik, weshalb sich ihre Implementierung aufwändig gestaltet. Das Framework setzt auf einen ontologiebasierten Ansatz zur Modellierung der Kontextinformationen. In Bezug auf die Abbildbarkeit möglicher Kontexte sowie Kontexthierarchien entspricht es damit den Anforderungen nach Erweiterbarkeit und Skalierbarkeit. Ein ontologiebasierter Ansatz wird daher auch für die weitere Umsetzung in Betracht gezogen.

5.4.4. Zusammenfassung

Der vorausgehende Abschnitt hat die Darstellung einer repräsentativen Auswahl von Ansätzen bestehender Systeme bzw. Rahmenwerke vorgenommen. Die Ansätze differieren in Architektur, Technik der Kontexterfassung, Kontextmodell sowie Methodik der Kontextauswertung. Gemein ist ihnen jedoch die saubere Trennung von Kontexterfassung bzw. -auswertung und Komponenten zum Zugriff auf das System. Das *Context Toolkit* von Dey et al. berücksichtigt als einziger dieser Ansätze ein Konzept zum Schutz der Privatsphäre, indem es einen expliziten Inhaber für jeden Kontext modelliert.

Die in Kapitel 5.3 definierten Anforderungen können von keinem der vorgestell-

ten Rahmenwerke vollständig erfüllt werden. Jeder dieser Ansätze beinhaltet jedoch Techniken, die für eine weitere Konzeption von Interesse sind. Sowohl das *Context Toolkit* von Dey et al. als auch das *Hydrogen Framework* von Hofer et al. umfassen Konzepte zur Kapselung von Sensoren. Diese *Widgets* bzw. *Adaptoren* sind für die Erhebung von Kontextdaten einer bestimmten Domäne zuständig und tragen zur Erweiterbarkeit, Austauschbarkeit und Wiederverwendbarkeit dieser Systeme bei. Aus diesen Gründen soll im Rahmen der Konzeption auf ein solches Konzept zurückgegriffen werden.

Im Hinblick auf die Modellierung von Kontextinformationen hat sich, wie bereits mehrfach erwähnt, der ontologiebasierte Ansatz als besonders erweiterbar und skalierbar erwiesen. Der umfangreichste der vorgestellten Ansätze, das *Context Framework* von Korpipää et al., verwendet ein solches semantisches Kontextmodell. Im Rahmen der weiteren Betrachtung soll dieser Modellierungsansatz ebenfalls zum Einsatz kommen.

5.5. Konzeption des Rahmenwerks

Nachdem eine Abwägung der geeigneten Konzepte und Techniken aus bestehenden Ansätzen vorgenommen wurde, soll in diesem Abschnitt ein eigenes Rahmenwerk konzipiert werden. Der erste Teil dieser Konzeption befasst sich mit Fragen der Systemarchitektur. Eine Beschreibung der Schnittstellen zwischen den einzelnen Komponenten des Systems erfolgt im zweiten Teil. Eine kritische Betrachtung der Konzeption schließt das Kapitel ab.

5.5.1. Systemarchitektur

Da eine Anforderung darin besteht, Kontextinformationen sowohl zwischen Applikationen und Kontextquellen als auch zwischen mobilen Nutzern austauschen zu können, wird eine Kombination der Konzepte der Middleware-Infrastrukturen und der entfernten Kontextserver intendiert. Die detaillierte Betrachtung aus den Perspektiven der einzelnen Komponenten wird nachfolgend – beginnend mit dem mobilen Gerät – vorgenommen.

5.5.1.1. Lokaler Kontextdienst

Zunächst soll die Betrachtung aus der Perspektive des mobilen Geräts im Vordergrund stehen. Um die Kontexterfassung von den mobilen Applikationen zu entkoppeln, wird, wie in Kapitel 5.2.1 erläutert, eine zusätzliche Ebene eingeführt, welche die Kontextverwaltung übernimmt. Da diese durch die Bereitstellung von Kontextinformationen einen Dienst für die mobilen Applikationen darstellt, wird sie nachfolgend mit *Lokaler Kontextdienst* bezeichnet. Die Interaktion zwischen mobilen Applikationen und dem lokalen Kontextdienst erfolgt über eine wohldefinierte

Programmierschnittstelle, die in Kapitel 5.5.2.1 beschrieben wird.

Eine weitere Zielsetzung besteht darin, beliebige Informationsquellen an die Kontextverwaltung anbinden zu können, die Kontextinformationen erfassen. Da diese Quellen diverser Ausprägung sein können, ist eine weitere Generalisierung notwendig. Dafür wurde unterhalb der Kontextverwaltung eine weitere Schicht ergänzt, die die Logik der Informationsquellen kapselt. Die Entitäten dieser Ebene werden nachfolgend als *Kontextquellen* bezeichnet. Aufgabe der Kontextquellen ist die Erfassung und Aufbereitung von Daten, die Kontextinformationen liefern. Diese Informationen werden auf Anfrage erfasst und an den Kontextdienst weitergegeben. Die Quellen dieser Daten können unterschiedlicher Herkunft sein. So ist neben Hardwaresensoren des mobilen Geräts oder Daten, die auf dem Gerät gespeichert sind, eine Anbindung an entfernte Informationsdienste realisierbar. Ähnlich der Ansätze der *Cues* von [Laer01], der *Widgets* von [Dey01] und der *Adaptoren* von [Hof03], werden physische und logische Sensoren abstrahiert. Eine Kontextquelle bündelt die Erfassung von *Attributen* einer Domäne. So ist beispielsweise der Breitengrad (`latitude`) ein Attribut der Kontextquelle Ort (`Location`).

Um die angedachte Architektur zu veranschaulichen, wurde der Aufbau aus der Sicht des mobilen Gerätes auf der Abbildung 5.5 zusammengefasst.

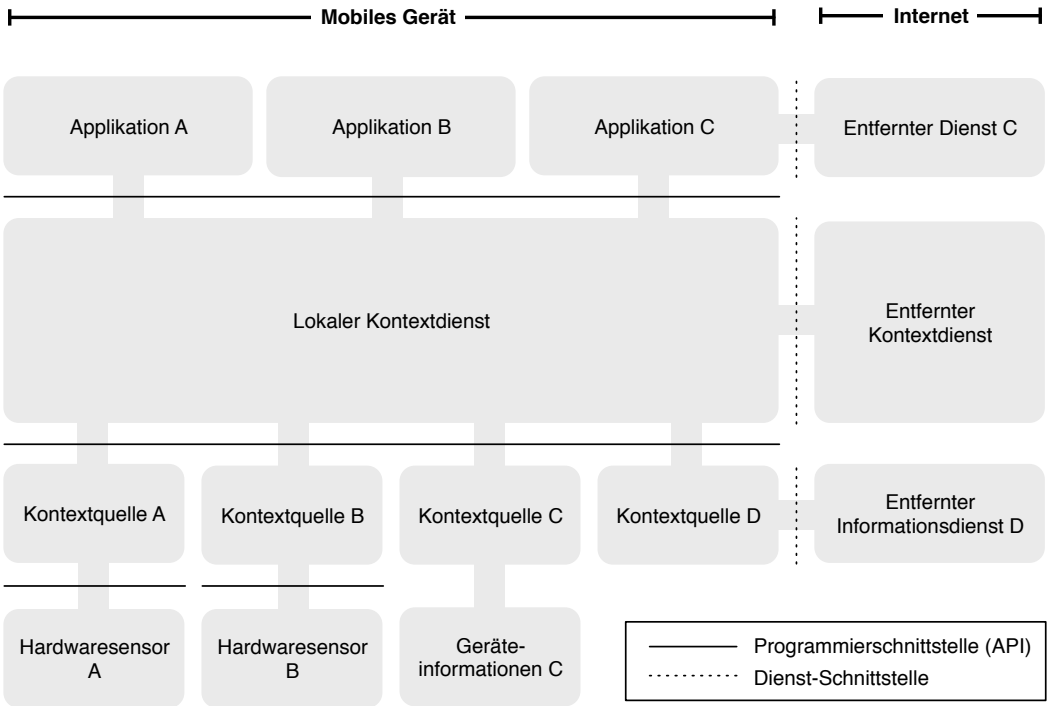

Abbildung 5.5.: Architektur des Rahmenwerks aus Sicht des mobilen Endgeräts

Die obere Ebene beinhaltet die mobilen Applikationen. Beispielhaft sind drei Applikationen A bis C dargestellt, von denen Applikation C einen entfernten Dienst nutzt. Um von Kontextinformationen profitieren zu können, nutzen diese Applikatio-

nen den lokalen Kontextdienst, der auf der Abbildung die darunter liegende Ebene darstellt. Die Kommunikation erfolgt, wie bereits beschrieben, über eine Programmierschnittstelle. Eine weitere Ebene tiefer sind die Kontextquellen angesiedelt. Hier dienen wieder verschiedene Beispiele der Veranschaulichung. Die Kontextquellen A und B beziehen ihre Daten über Plattform-APIs des mobilen Gerätes aus Hardwaresensoren. Die Kontextquelle C liefert Informationen, die im Gerät gespeichert sind und Kontextquelle D bezieht ihre Informationen von einem entfernten Informationdienst D. Die Kontextquellen sind ebenfalls über eine wohldefinierte Programmierschnittstelle an den Kontextdienst angebunden.

Der lokale Kontextdienst hat die Aufgabe, die Kontextinformationen von den Kontextquellen zu aggregieren, sie zwischenzuspeichern, und den mobilen Applikationen zur Verfügung zu stellen. Das eigentliche Kontextwissen jedoch wird aus Gründen des Informationsaustauschs sowie der Charakteristika des Kontextmodells nicht auf dem Gerät vorgehalten, sondern durch einen entfernten Kontextdienst. Dieser wird im nächsten Kapitel beschrieben.

5.5.1.2. Entfernter Kontextdienst

Dieser Abschnitt betrachtet das Rahmenwerk aus der Perspektive des entfernten Kontextdienstes. Er hat die Aufgabe, das Kontextwissen vorzuhalten und den mobilen Geräten – den Dienstnutzern – zur Verfügung zu stellen. Kapitel 4.2 hat verschiedene Modellierungsansätze für Kontextinformationen aufgegriffen. Auch wenn ontologiebasierte Modelle die größte Flexibilität ermöglichen, ist der Einsatz verschiedener Kontextmodelle je nach Zielsetzung und Anwendungsfall abzuwägen. Da eine möglichst generische Konzeption angestrebt wird, soll das System unabhängig vom eingesetzten Kontextmodell sein. Dafür wird eine Entkopplung der Modell-Komponente und des eigentlichen Dienstes auf der Seite des Kontextservers vorgenommen.

Die Abbildung 5.6 veranschaulicht den entfernten Kontextdienst sowie das davon konzeptionell getrennte Kontextmodell, welches das eigentliche Kontextwissen beinhaltet.

Auf der Abbildung sind beispielhaft vier mobile Endgeräte dargestellt, die den entfernten Kontextdienst nutzen. Die Interaktion zwischen diesen Geräten und dem Kontextdienst sowie zwischen Kontextdienst und Kontextmodell erfolgt über definierte Dienst-Schnittstellen. Die Schnittstellen des Rahmenwerks werden im nachfolgenden Kapitel erörtert.

5.5.2. Schnittstellen

Dieses Kapitel befasst sich mit den verschiedenen Schnittstellen zwischen den Modulen des konzipierten Systems. Diese nehmen einen hohen Stellenwert ein, da sie die Grundlage für Interoperabilität und Modularität bilden und somit die Wiederver-

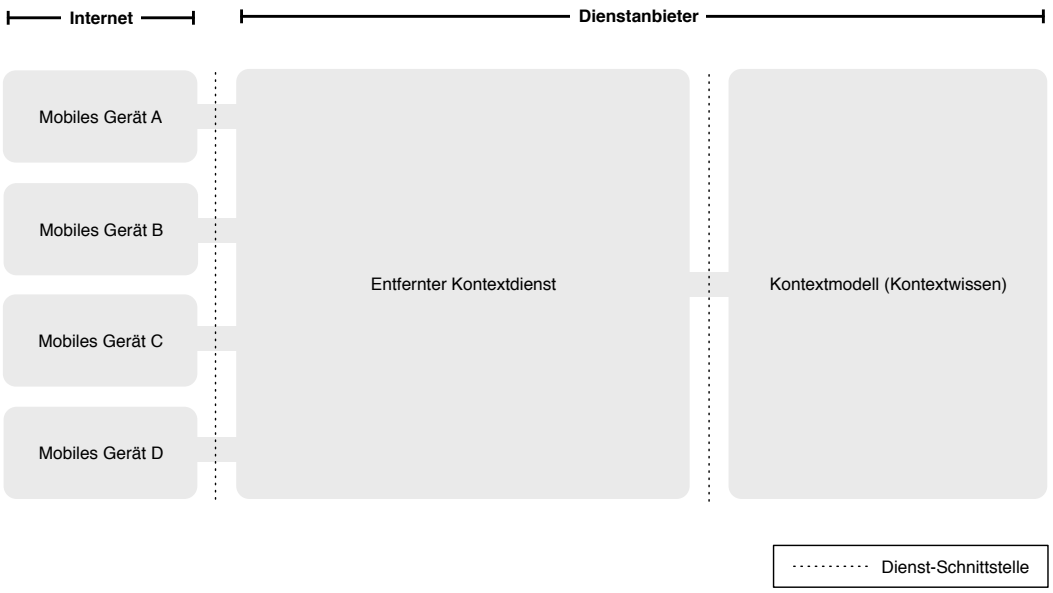

Abbildung 5.6.: Architektur des Rahmenwerks aus Sicht des Dienstanbieters

wendbarkeit und Austauschbarkeit der einzelnen Systemkomponenten ermöglichen.

5.5.2.1. Schnittstelle für mobile Applikationen

Zunächst wird die Programmierschnittstelle beschrieben, über die mobile Applikationen Kontextinformationen vom lokalen Kontextdienst beziehen können. Sie bildet den Übergangspunkt zwischen mobilen Applikationen und der Ebene, die das Rahmenwerk zwischen Sensoren und Applikationen bereitstellt.

Zum Abrufen von Kontextinformationen sind zwei Möglichkeiten vorgesehen: ein *Abfrage-Mechanismus* und ein *Benachrichtigungs-Mechanismus*. Der Abfrage-Mechanismus ermöglicht einen expliziten Abruf von Kontextattributen und ihren Werten durch die Applikationen *(Pull-Mode)*. Der Benachrichtigungs-Mechanismus erlaubt es Applikationen, sich beim Kontext-Dienst anzumelden und bei bestimmten Ereignissen (z.B. Kontextänderungen) benachrichtigt zu werden *(Push-Mode)*.

Die Nachfolgende Auflistung fasst alle zu unterstützenden Funktionen der Schnittstelle zusammen:

- Bereitstellung des Kontextes
  - Explizite Abfrage *(Pull-Mode)*
  - Benachrichtigung bei Änderungen *(Push-Mode)*
- Bearbeitung von Kontexten
  - Hinzufügen von Kontexten
  - Modifizieren von Kontexten
  - Entfernen von Kontexten

- Steuerung der Kontextquellen
  - Information über verfügbare Kontextquellen
  - Abfragen des Status von Kontextquellen
  - Aktivieren von Kontextquellen
  - Deaktivieren von Kontextquellen
  - Abfragen von Attributwerten
- Verwaltung von Benutzern
  - Informationen über Benutzer
  - Hinzufügen von Benutzern
  - Modifizieren von Benutzern
  - Entfernen von Benutzern

5.5.2.2. Schnittstelle für Kontextquellen

Die Schnittstelle für Kontextquellen definiert die Funktionalität, die jede Kontextquelle implementieren muss, um an den Kontextdienst angebunden werden zu können. Da eine Kontextquelle mehrere Attributwerte liefern kann, sind neben der Abfrage dieser Werte die Bezeichnungen aller zur Verfügung stehenden Attribute bereitzustellen. Nachfolgende Liste fasst die zu realisierenden Funktionen der Schnittstelle für Kontextquellen zusammen:

- Ermittlung aller verfügbaren Kontextattribute
- Abfrage aller aktuellen Attributwerte
- Abfrage eines spezifischen aktuellen Attributwerts

5.5.2.3. Schnittstelle des entfernten Kontextdienstes

Die Dienst-Schnittstelle des entfernten Kontextdienstes dient der Kommunikation zwischen mobilen Endgeräten und dem Kontextserver, der die Kontextinformationen in einem Kontextmodell vorhält. Die folgende Auflistung beinhaltet alle Funktionen, die der entfernte Kontextdienst diesen Dienstnutzern bereitstellen sollte:

- Ermittlung von Informationen über das Kontextmodell
  - Bereitstellung aller Kontexttypen des Modells
  - Bereitstellung aller Kontexte des Modells
  - Bereitstellung aller Benutzer des Modells
- Ermittlung zutreffender Kontexte
  - Bereitstellung aller zutreffenden Kontexte auf Basis übermittelter Attributwerte
- Bearbeitung des Kontextmodells
  - Einfügen eines Kontextes
  - Bearbeiten eines Kontextes
  - Entfernen eines Kontextes
  - Einfügen eines Benutzers

- Bearbeiten eines Benutzers
- Entfernen eines Benutzers

5.5.2.4. Schnittstelle des Kontextmodells

Die Schnittstelle des Kontextmodells hängt maßgeblich vom eingesetzten Modellierungsansatz ab. Aus diesem Grund kann an dieser Stelle keine allgemein gültige Definition der Schnittstelle vorgenommen werden. Das Kontextmodell muss dem Kontextdienst sowohl den *lesenden* als auch den *schreibenden* Zugriff auf seine gespeicherten Daten gewähren. Die genaue Funktionalität hängt von der konkreten Implementierung ab.

5.6. Kritische Betrachtung

Dieses Kapitel hat sich mit Rahmenwerken zur Repräsentation und Verwaltung von Kontextinformationen für mobile kontextsensitive Applikationen befasst. Dafür wurde zunächst eine Darlegung diverser Gründe für solche Rahmenwerke vorgenommen. Anschließend wurden Fragen der Architektur dieser Systeme aufgegriffen. Nach einer Zusammenstellung der Anforderungen folgte die Untersuchung und Bewertung bestehende Rahmenwerke im Hinblick auf Verwendbarkeit im Rahmen der Zielsetzung dieser Arbeit.

Schlussendlich erfolgte die Ausarbeitung eines theoretisches Konzepts. Dieses dient als Grundlage für die im nachfolgenden Kapitel dokumentierte prototypische Umsetzung in Form einer Referenzimplementation. Es wurde eine bewusste Abstraktion von konkreten Plattformen und Modellen vorgenommen, um den angestrebten, generischen Charakter zu erzielen.

Anforderungen an Sicherheit und Datenschutz werden im Rahmen dieser Arbeit vorerst vernachlässigt. Daher beinhaltet das Konzept keinerlei Bestrebungen in diese Richtung.

Die Konzeption stützt sich auf Verfahren und Architektureigenschaften, die teilweise von bestehenden Ansätzen adaptiert wurden. Sie stellt lediglich eine vieler Möglichkeiten dar, um den Aufbau und die Funktionen eines Rahmenwerks zu umreissen.

Durch die Referenzimplementation soll evaluiert werden, ob und inwieweit sich das theoretische Konzept in die Praxis umsetzen lässt.

6. Referenzimplementierung

Dieses Kapitel hat die Dokumentation der prototypischen Realisierung des konzipierten Rahmenwerkes in Form einer Referenzimplementation zum Thema. Zunächst wird ein Überblick über die Zielsetzung der Implementierung und den Umfang des umgesetzten Systems gegeben. Anschließend wird erläutert, welche Werkzeuge und Plattformen für die Umsetzung zum Einsatz kamen. Im Hauptteil wird das System detailliert aus den Perspektiven des mobilen Geräts sowie des Serverdienstes beschrieben. Ebenso erfolgt anhand von Ablaufdiagrammen eine Veranschaulichung der Funktionsweise und des Zusammenwirkens der Komponenten. Eine kritische Betrachtung des implementierten Systems schließt das Kapitel ab.

6.1. Zielsetzung und Umfang

Die Referenzimplementierung des Systems soll die Schlüssigkeit des Konzepts durch eine praktische Umsetzung veranschaulichen. Es soll gezeigt werden, ob die skizzierte Systemarchitektur in die Realität umgesetzt und als Grundlage für weitere Untersuchungen herangezogen werden kann. Darüber hinaus lässt sich evaluieren, ob und inwieweit das angestrebte Ziel einer kompakten Architektur erreicht werden kann.

Das implementierte System besteht aus drei Komponenten: einer *Kontextbibliothek für das mobile Gerät*, einem *entfernten Kontextdienst* und einer mobilen *Applikation zu Demonstrationszwecken*. Der Funktionsumfang besteht grob umrissen in der Abfrage von Kontextinformationen, die in einem semantischen Kontextmodell hinterlegt sind, welches sich auf der Seite des entfernten Kontextdienstes befindet. Die Kontextermittlung erfolgt auf Basis von Attributwerten, die durch Kontextquellen auf der Seite des mobilen Geräts erhoben werden. Im Rahmen der praktischen Umsetzung wurden insgesamt sechs dieser Kontextquellen implementiert. Das umgesetzte System unterstützt die Abfrage beliebiger Kontexte auf Basis beliebiger Attributwerte der Kontextquellen. Ein schreibender Zugriff auf das Kontextmodell, also die Möglichkeit, Kontexte und Benutzer über mobile Applikationen direkt einzufügen bzw. zu ändern, wurde nicht realisiert.

6.2. Werkzeuge und Plattformen

Bei der Wahl der Werkzeuge und Plattformen für die Umsetzung wurde Wert darauf gelegt, zeitgemäße Produkte einzusetzen, um dem Ziel der Realisierbarkeit mit

aktuell verfügbarer Technologie gerecht zu werden.

Als mobiles Gerät wurde das *Apple iPhone*[1] ausgewählt, welches als Repräsentant moderner Smartphones derzeit eines der fortschrittlichsten Geräte darstellt. Das iPhone verfügt über eine Vielzahl von Sensoren, die sich durch gut dokumentierte Programmierschnittstellen mit geringem Aufwand ansprechen lassen. Das Gerät wurde von Beginn an für einen permanent verfügbaren Internetzugang konzipiert, weshalb es sich besonders für den Einsatz in einem verteilten System eignet. Zur Entwicklung von Software liefert der Hersteller eine kostenlose Sammlung von Werkzeugen aus, wozu unter anderem die integrierte Entwicklungsumgebung *Xcode*[2], der *Interface Builder* und der *iPhone Simulator* zählen. Die Anwendungsentwicklung auf der iPhone-Plattform ist einigen Einschränkungen unterlegen, auf die in Kapitel 6.3.1 genauer eingegangen wird. iPhone-Anwendungen werden in der Programmiersprache *Objective C*[3] entwickelt.

Die Seite des entfernten Kontextdienstes besteht aus einer *Web-Applikation* für die Anwendungslogik sowie einem *Triple-Store*, der das semantische Kontextmodell in Form einer *Ontologie* beinhaltet.

Die Web-Applikation wurde mit dem *Sinatra-Framework*[4] realisiert. Sinatra verfolgt das Ziel, mit minimalem Aufwand Web-Applikationen zu entwickeln, die dem ressourcenorientierten Softwarearchitekturstil REST[5] genügen. Dabei wird auf Konzepte wie das MVC[6]-Paradigma bewusst verzichtet. Sinatra-Anwendungen werden in der Programmiersprache *Ruby*[7] entwickelt.

Das Kontextmodell wird durch eine *Ontologie* repräsentiert. Diese wurde mit dem *Protégé Ontology Editor*[8] erstellt, einem Open Source Werkzeug der Stanford University. Bereitgestellt wird diese Ontologie durch einen *Triple-Store*, welcher die semantischen Tripel der Ontologie vorhält. Als Triple-Store kommt für diese Umsetzung *OpenRDF Sesame*[9] zum Einsatz, ein Open Source RDF-Framework, welches einen *SPARQL*[10]-Endpunkt für semantische Anfragen als Web Service zur Verfügung stellt. Sesame selbst ist eine Java Web-Applikation, die einen Java Servlet-Container als Laufzeitumgebung voraussetzt. Als Servlet-Container dient für die Beispielumsetzung *Apache Tomcat*[11], ein etabliertes, kostenloses und ebenfalls quelloffenes Produkt der Apache Software Foundation.

[1] `http://www.apple.com/de/iphone/specs.html`
[2] `http://developer.apple.com/tools/xcode/`
[3] `http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/ObjectiveC/`
[4] `http://www.sinatrarb.com/`
[5] *Representational State Transfer*
[6] *Model View Controller*
[7] `http://www.ruby-lang.org/de/`
[8] `http://protege.stanford.edu/`
[9] `http://www.openrdf.org/`
[10] *SPARQL Protocol and RDF Query Language*
[11] `http://tomcat.apache.org/`

6.3. Implementation für das mobile Gerät

Dieses Kapitel beschreibt die Implementation zunächst aus der Perspektive des mobilen Geräts. Um zu Beginn einen Überblick zu schaffen, wird die gesamte Architektur der mobilen Komponenten als Schaubild vorangestellt (siehe Abbildung 6.1). Die einzelnen Komponenten werden nachfolgend genauer erläutert.

In Abschnitt 6.3.1 wird auf den Kontextdienst eingegangen, welcher den Kern des Systems auf mobiler Seite darstellt. Nach einer Einführung folgt die Dokumentation der Schnittstellen für mobile Applikationen und für Kontextquellen. Anschließend werden die im Rahmen dieser Umsetzung implementierten Kontextquellen behandelt. Schlussendlich erfolgt eine Beschreibung der Klassen des Systems durch UML-Diagramme.

Abschnitt 6.3.2 besteht in der Beschreibung der Demo-Applikation, die zu Testzwecken entwickelt wurde, mithilfe derer die Funktionsweise des Kontextdienstes anhand verschiedener Kontexte getestet werden kann. In diesem Abschnitt wird der Funktionsumfang und die Funktionsweise dieser Demo-Applikation erläutert.

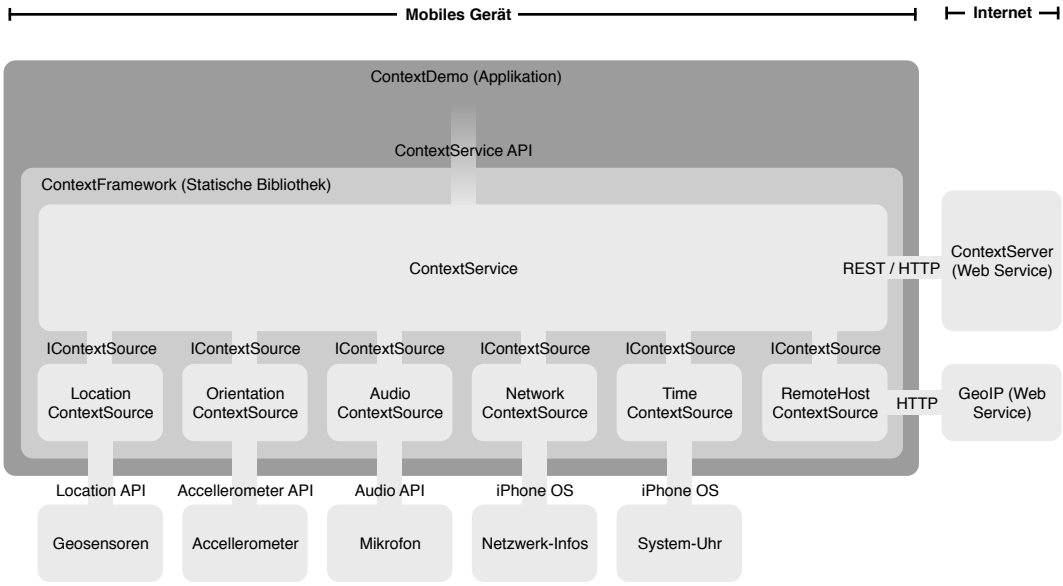

Abbildung 6.1.: Architektur der Referenzimplementation für das iPhone

6.3.1. Kontextdienst als statische Bibliothek

iPhone OS, das Betriebssystem des iPhones, wurde als *Single-Tasking*-System entworfen. Bis auf einige essentielle Hintergrunddienste wie Telefonfunktion, SMS und E-Mail können Anwendungen nicht gleichzeitig, sondern ausschließlich einzeln betrieben werden. Für die Implementation des Kontextdienstes bedeutet das, dass es nicht möglich ist, eine Softwarekomponente als Abstraktionsebene zur Kontextverwaltung permanent als separaten Dienst bereitzustellen.

Um trotz dieser Einschränkung eine universell einsetzbare Komponente zu realisieren, wurde eine *Static Library*[12] implementiert. Eine statische Bibliothek lässt sich als kompiliertes Softwaremodul beschreiben, welches während der Entwicklung einer mobilen Applikation als externes Framework eingebunden werden kann (in der Abbildung 6.1 mit `ContextFramework` bezeichnet). Die Programmlogik des Rahmenwerks ist nach dem Kompiliervorgang ein Teil der iPhone-Applikation selbst, stellt aber ein in sich abgeschlossenes Modul dar, welches unabhängig von der Applikation ausgetauscht und aktualisiert werden kann. Die Funktionen dieses Moduls lassen sich durch eine definierte Programmierschnittstelle von der iPhone-Applikation nutzen (in der Abbildung 6.1 mit `ContextService API` bezeichnet).

Die statische Bibliothek und ihr Quellcode wurden zusammen mit einer Anleitung zur Integration in eigene iPhone-Projekte im Internet veröffentlicht und lassen sich über die folgende URL beziehen:

`http://github.com/flxmllr/mobile-context-iphone-lib/`

Das Modul enthält die Anwendungslogik zur Bereitstellung und Verwaltung des Kontextes für den mobilen Teil des Systems (`ContextService` in der Abbildung 6.1) sowie die Anwendungslogik der einzelnen Kontextquellen (`[Typ]ContextSource` in der Abbildung 6.1). Der `ContextService` bildet den Kern des Dienstes und hat die Aufgabe, den Kontext bereitzustellen und mit dem entfernten Kontextdienst sowie den Kontextquellen zu kommunizieren. Eine Kontextquelle ermittelt Attributwerte einer beliebigen Quelle (z.B. Sensor, Gerät, Web Service) und gibt sie auf Anfrage an den Kontextdienst weiter. Zur Kommunikation zwischen mobilen Applikationen und dem Kontextdienst und zwischen dem Kontextdienst und den Kontextquellen wurden Programmierschnittstellen definiert, die aus vordefinierten Methoden bestehen. Diese werden nachfolgend vorgestellt.

6.3.1.1. Schnittstelle für mobile Applikationen

Die Programmierschnittstelle für die mobilen Applikationen (siehe `ContextService API` in Abbildung 6.1) hat das Ziel, den Entwicklern von iPhone-Anwendungen alle Funktionen des Kontextdienstes über definierte Methoden bereitstellen.

Der nachfolgende Quellcode 6.1 zeigt einen Auszug aus der Klasse `ContextService`, deren öffentliche Methoden die Schnittstelle für die mobilen Applikationen definieren. Die Kommentare erläutern jeweils die Aufgabe der Methoden, ihre Parameter sowie ihre Rückgabewerte.

[12] Eine *statische Bibliothek* – oder auch *statisch gelinkte Bibliothek* – ist eine Sammlung von Routinen, externen Funktionen und Variablen, welche zum Zeitpunkt des Kompilierens aufgelöst, und in eine Zielapplikation kopiert werden, um Objektcode zu erzeugen. Statische Bibliotheken werden nach dem Kompiliervorgang durch einen so genannten *Linker* mit dem Kompilat verbunden. Anschließend setzt der Linker daraus ein ausführbares Programm oder auch eine andere ausführbare Komponente zusammen.

Quellcode 6.1: Öffentliche Methoden der Klasse `ContextService`

```
1  //
2  // These are the context service API methods that mobile applications can use
3  //
4
5  /*
6   * This method requests all existing users from the context model.
7   *
8   * Parameters:
9   *   none
10  *
11  * Returns:
12  *   An NSArray of NSStrings with all user names
13  *   nil if no users were found or an error did occur
14  *
15  */
16 - (NSArray *)getUsers;
17
18 /*
19  * This method adds a user to the context model.
20  *
21  * Parameters:
22  *   userName: An NSString containing the user name
23  *
24  * Returns:
25  *   YES if the user was added successfully
26  *   NO if the user does already exist or an error did occur
27  *
28  */
29 - (BOOL)addUser:(NSString *)userName;
30
31 /*
32  * This method removes a user from the context model.
33  *
34  * Parameters:
35  *   userName: An NSString containing the user name
36  *
37  * Returns:
38  *   YES if the user was removed successfully
39  *   NO if the user was not found an error did occur
40  *
41  */
42 - (BOOL)removeUser:(NSString *)userName;
43
44 /*
45  * This method requests all global contexts from the context model.
46  *
47  * Parameters:
48  *   none
49  *
50  * Returns:
51  *   An NSDictionary of Context with keys = context types and values = Context objects
52  *   nil if no contexts were found or an error did occur
53  *
54  */
55 - (NSDictionary *)getContexts;
56
57 /*
58  * This method requests all contexts for a given user from the context model.
59  *
60  * Parameters:
61  *   userName: An NSString containing the user name
62  *
63  * Returns:
64  *   An NSDictionary of Context with keys = context types and values = Context objects
65  *   nil if no contexts were found or an error did occur
66  *
67  */
68 - (NSDictionary *)getContextsForUser:(NSString *)userName;
69
70 /*
71  * This method requests all contexts for a given user and a given type from the context
        model.
72  *
73  * Parameters:
74  *   userName: An NSString containing the user name
75  *   contextType: An NSString containing the context type
76  *
77  * Returns:
```

```
78  *    An NSDictionary of Context with keys = context types and values = Context objects
79  *    nil if no contexts were found or an error did occur
80  *
81  */
82 - (NSDictionary *)getContextsForUser:(NSString *)userName withType:(NSString *)contextType
        ;
83
84 /*
85  * This method adds a global context with a given type and given condidions to the context
        model.
86  *
87  * Parameters:
88  *    contextName: An NSString containing the context name
89  *    contextType: An NSString containing the context type
90  *    conditions: An NSDictionary of NSString with keys = subjects and values = objects
91  *
92  * Returns:
93  *    YES if context was successfully added
94  *    NO of context exists or an error did occur
95  *
96  */
97 - (BOOL)addContext:(NSString *)contextName withType:(NSString *)contextType withConditions
        :(NSDictionary *)conditions;
98
99 /*
100  * This method adds a context with a given type and given condidions for a given user to
        the context model.
101  *
102  * Parameters:
103  *    contextName: An NSString containing the context name
104  *    contextType: An NSString containing the context type
105  *    conditions: An NSDictionary of NSString with keys = subjects and values = objects
106  *    userName: An NSString containing the user name
107  *
108  * Returns:
109  *    YES if context was successfully added
110  *    NO of context exists or an error did occur
111  *
112  */
113 - (BOOL)addContext:(NSString *)contextName withType:(NSString *)contextType withConditions
        :(NSDictionary *)conditions forUser:(NSString *)user;
114
115 /*
116  * This method removes a context with a given type from the context model.
117  *
118  * Parameters:
119  *    contextName: An NSString containing the context name
120  *
121  * Returns:
122  *    YES if the context was removed successful
123  *    NO if the context was not found an error did occur
124  *
125  */
126 - (BOOL)removeContext:(NSString *)contextName;
127
128 /*
129  * This method removes a context with a given type for a given user from the context model
        .
130  *
131  * Parameters:
132  *    contextName: An NSString containing the context name
133  *    userName: An NSString containing the user name
134  *
135  * Returns:
136  *    YES if the context was removed successfully
137  *    NO if the context was not found an error did occur
138  *
139  */
140 - (BOOL)removeContext:(NSString *)contextName forUser:(NSString *)userName;
141
142 /*
143  * This method requests all context sources currently available in the context service.
144  *
145  * Parameters:
146  *    none
147  *
148  * Returns:
149  *    An NSArray of NSString with all context source names
150  *    nil if no context sources were found or an error did occur
```

```
151  *
152  */
153 - (NSArray *)getContextSources;
154
155 /*
156  * This method requests all attributes the context source can deliver.
157  *
158  * Parameters:
159  *   source: An NSString containing the context source name
160  *
161  * Returns:
162  *   An NSArray if NSString with the names of the context attributes
163  *   nil if no attributes were found or an error did occur
164  *
165  */
166 - (NSArray *)getSourceAttributes:(NSString *)source;
167
168 /*
169  * This method checks if a given context source is enabled.
170  *
171  * Parameters:
172  *   contextSource: An NSString containing the name of the context source
173  *
174  * Returns:
175  *   YES if the context source is enabled
176  *   NO if the context source is disabled
177  *
178  */
179 - (BOOL)contextSourceEnabled:(NSString *)contextSource;
180
181 /*
182  * This method enables a given context source.
183  *
184  * Parameters:
185  *   contextSource: An NSString containing the name of the context source
186  *
187  * Returns:
188  *   YES if the context source was enabled successfully
189  *   NO if the context source was not found an error did occur
190  *
191  */
192 - (BOOL)enableContextSource:(NSString *)contextSource;
193
194 /*
195  * This method disables a given context source.
196  *
197  * Parameters:
198  *   contextSource: An NSString containing the name of the context source
199  *
200  * Returns:
201  *   YES if the context source was disenabled successfully
202  *   NO if the context source was not found an error did occur
203  *
204  */
205 - (BOOL)disableContextSource:(NSString *)contextSource;
206
207 /*
208  * This method requests all context source attribute values currently available by the
         context service.
209  *
210  * Parameters:
211  *   none
212  *
213  * Returns:
214  *   An NSDictionary of Attribute with keys = source and values = Attribute objects
215  *   nil if no context source attributes were found or an error did occur
216  *
217  */
218 - (NSDictionary *)getSourceAttributeValues;
219
220 /*
221  * This method requests a specific context attribute value from the context service.
222  *
223  * Parameters:
224  *   contextSourceType: An NSString containing the context source type
225  *
226  * Returns:
227  *   An Attribute with the current context data gathered by the source
228  *   nil if the context attribute was not found or an error did occur
```

```
229  *
230  */
231 - (Attribute *)getSourceAttributeValue:(NSString *)contextSourceType;
232
233 /*
234  * This method registers an object for context change notifications for a given context.
235  *
236  * Parameters:
237  *   observer: An Object registering as an observer
238  *   contextName: An NSString containing the context name
239  *
240  * Returns:
241  *   YES if the registration was successful
242  *   NO if an error did occur
243  *
244  */
245 - (BOOL)registerForContextChangeNotifications:(id)observer forContext:(NSString *)
        contextName;
246
247 /*
248  * This method registers an object for context change notifications for a given context
        and a given user.
249  *
250  * Parameters:
251  *   observer: An Object registering as an observer
252  *   contextName: An NSString containing the context name
253  *   userName: An NSString containing the user name
254  *
255  * Returns:
256  *   YES if the registration was successful
257  *   NO if an error did occur
258  *
259  */
260 - (BOOL)registerForContextChangeNotifications:(id)observer forContext:(NSString *)
        contextName forUser:(NSString *)userName;
261
262 /*
263  * This method unregisters an object from context change notifications.
264  *
265  * Parameters:
266  *   observer: An Object registering as an observer
267  *   contextName: An NSString containing the context name
268  *   userName: An NSString containing the user name
269  *
270  * Returns:
271  *   YES if the unregistration was successful
272  *   NO if an error did occur
273  *
274  */
275 - (BOOL)unregisterFromContextChangeNotifications:(id)observer;
```

6.3.1.2. Schnittstelle für Kontextquellen

An den Kontextdienst auf dem mobilen Gerät lassen sich beliebige Kontextquellen anbinden, die das Interface `IContextSource` implementieren. Diese Schnittstelle definiert die Methoden, die jede Kontextquelle bereitstellen muss, um in das System integriert werden zu können.

Im Rahmen der praktischen Umsetzung wurden sechs Kontextquellen implementiert, die als Teil der statischen Bibliothek ausgeliefert werden. Auf diese wird im nächsten Abschnitt (Kapitel 6.3.1.3) eingegangen.

Der Quellcode 6.2 zeigt einen Auszug aus dem Interface `IContextService`, welcher die zu implementierenden Methoden enthält.

Quellcode 6.2: Methoden der Schnittstelle `IContextSource`

```
1  //
2  // These are the context source API methods that can be called on every context source
3  //
4
5  /*
6   * This method requests all attributes the context source can deliver.
7   *
8   * Parameters:
9   *   none
10  *
11  * Returns:
12  *   An NSArray of NSString with the names of the context attributes
13  *   nil if no context attribute was found or an error did occur
14  *
15  */
16 - (NSArray *)getAttributes;
17
18 /*
19  * This method requests all current attribute values from the context source.
20  *
21  * Parameters:
22  *   none
23  *
24  * Returns:
25  *   An NSDictionary of Attributes with all current attribute values
26  *   nil if no attribute values were found or an error did occur
27  *
28  */
29 - (NSMutableDictionary *)getAttributeValues;
30
31 /*
32  * This method requests a specific current attribute value from the context source.
33  *
34  * Parameters:
35  *   attribute: An NSString containing the name of the requested attribute
36  *
37  * Returns:
38  *   An Attribute with the current context attribute value
39  *   nil if no context attribute value was found or an error did occur
40  *
41  */
42 - (Attribute *)getAttributeValue:(NSString *)attribute;
```

6.3.1.3. Implementierte Kontextquellen

Bei der Referenzimplementierung wurden insgesamt sechs Kontextquellen entwickelt, die in der statischen Bibliothek enthalten sind. Die nachfolgende Auflistung fast diese Kontextquellen und ihre Attribute zusammen:

1. `LocationContextSource` *(Domäne: Geografische Informationen)*
   - `longitude` *(Längengrad)*
   - `latitude` *(Breitengrad)*
   - `altitude` *(Höhenlage)*
   - `speed` *(Geschwindigkeit)*
2. `OrientationContextSource` *(Domäne: Ausrichtung des Geräts)*
   - `orientationX` *(Ausrichtung in X-Richtung)*
   - `orientationY` *(Ausrichtung in Y-Richtung)*
   - `orientationZ` *(Ausrichtung in Z-Richtung)*
3. `AudioContextSource` *(Domäne: Akustische Informationen)*
   - `averageLevel` *(Lautstärke-Durchschnittswert)*
   - `peakLevel` *(Lautstärke-Spitzenwert)*

4. `NetworkContextSource` *(Domäne: Netzwerkinformationen)*
   - `ipAddress` *(Lokale IP-Adresse)*
   - `wlanName` *(SSID des WiFi-Netzwerks)*
5. `TimeContextSource` *(Domäne: Zeitinformationen)*
   - `date` *(Datum)*
   - `time` *(Uhrzeit)*
   - `timezone` *(Zeitzone)*
   - `week` *(Woche des Jahres)*
   - `weekday` *(Tag der Woche)*
6. `RemoteHostContextSource` *(Domäne: Informationen zur Internetverbindung)*
   - `countryCode` *(Ländercode der Internetverbindung)*

Um möglichst repräsentative Beispiele für Kontextquellen zu veranschaulichen, verwenden die implementierten Kontextquellen unterschiedliche Techniken zur Ermittlung ihrer Attributwerte.

Die Kontextquellen `LocationContextSource`, `OrientationContextSource` und `AudioContextSource` greifen über die Plattform-API des iPhone OS auf Hardwaresensoren zurück *(Geo-, Beschleunigungssensor bzw. Mikrofon)*. Die Quellen `NetworkContextSource` und `TimeContextSource` ermitteln ihre Attribute *(Netzwerk bzw. Zeitinformationen)* direkt über das iPhone OS. Um die Anbindung einer Kontextquelle an einen entfernten Dienst zu demonstrieren, wurde mit der `RemoteHostContextSource` eine Kontextquelle umgesetzt, die Informationen zur Internetverbindung auf Basis der IP-Adresse von einem Web Service[13] über das Internet abfragt.

6.3.1.4. Klassenmodell

Das Klassenmodell, welches alle Klassen beinhaltet, die in der statischen Bibliothek enthalten sind, ist auf dem UML-Diagramm in Abbildung 6.2 dargestellt. Die Schnittstellen-Methoden für mobile Applikationen finden sich in der Hauptklasse `ContextService` wieder. Die Schnittstelle für Kontextquellen wird durch das Interface `IContextSource` beschrieben. Die sechs implementierten Kontextquellen sind von der Klasse `ContextSource` abgeleitet, um gemeinsame Eigenschaften wie das Verzeichnis der Attribute nicht individuell implementieren zu müssen. Auf der Ebene des iPhone OS gibt es zahlreiche Plattform-APIs, von denen drei durch Kontextquellen genutzt werden *(Audio API, Location API und Accellerometer API)*.

Zur Repräsentation von Kontexten und Attributen wurden die Datenstrukturen `Context` und `Attribute` entwickelt. Diese besitzen alle Eigenschaften, die einen Kontext bzw. ein Attribut charakterisieren.

Die Kommunikation mit dem entfernten Kontextdienst über REST erfolgt mittels des quelloffenen `ObjectiveResource`[14]-Frameworks, welches in die statische Bibliothek integriert wurde.

[13] `http://www.hostip.info/`
[14] `http://www.iphoneonrails.com/`

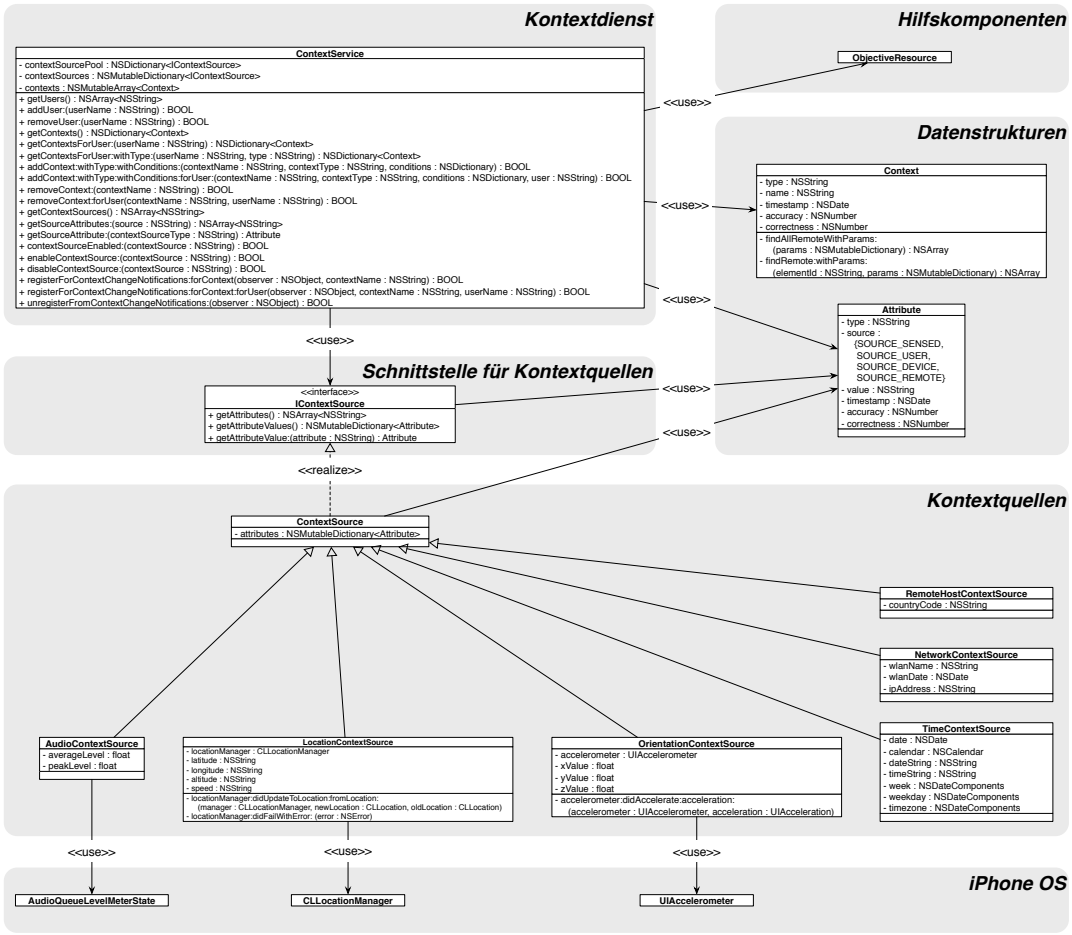

Abbildung 6.2.: UML-Klassendiagramm der statischen Bibliothek für das iPhone

6.3.2. Demo-Applikation

Zum Testen des Kontextdienstes und für Demonstrationszwecke wurde eine eigenständige iPhone-Applikation entwickelt, die die statische Bibliothek beinhaltet (siehe `ContextDemo` in der Abbildung 6.1).

Die Demo-Applikation und ihr Quellcode wurden zusammen mit einer Kurzbeschreibung zur Inbetriebnahme im Internet veröffentlicht und lassen sich über die folgende URL beziehen:

`http://github.com/flxmllr/mobile-context-iphone-demo/`

6.3.2.1. Funktionsumfang

Die Demo-Applikation verfügt über folgende Funktionalität:

1. Die Anzeige und Aktualisierung aller verfügbaren *Kontextquellen* mit der Möglichkeit, diese zu aktivieren bzw. zu deaktivieren (siehe Abbildung 6.3, links)
2. Die Anzeige und Aktualisierung aller verfügbaren *Attribute* und ihrer Werte mit der Möglichkeit, diese automatisch aktualisieren zu lassen (siehe Abbildung 6.3, Mitte)

3. Die Anzeige und Aktualisierung aller aktuellen *Kontexte* und deren Typen mit der Möglichkeit, diese automatisch aktualisieren zu lassen (siehe Abbildung 6.3, rechts)

Abbildung 6.3.: Funktionen der Demo-Applikation

Die Applikation verfügt über kein selbstadaptives Verhalten in Bezug auf den Kontext. Sie dient lediglich dazu, die Kontextermittlung auf Basis der aktivierten Kontextquellen und der erhobenen Attributwerte zu veranschaulichen und alle bzw. die aktuellen im System modellierten Kontexte anzuzeigen.

6.3.2.2. Funktionsweise

Es folgt eine kurze Erläuterung der Funktionsweise der Demo-Applikation.

- Zur Anzeige bzw. Aktualisierung aller Kontextquellen (Abbildung 6.3, links) wird zunächst über die Methode `getContextSources` eine Liste aller verfügbaren Quellen vom Kontextdienst abgefragt. Bei Aktivierung bzw. Deaktivierung einer Kontextquelle durch den Benutzer wird die Methode `enableContextSource:` bzw. `disableContextSource:` aufgerufen. Die Anzeige der Attribute der jeweiligen Quelle erfolgt mittels `getSourceAttributes:`.
- Die Anzeige und Aktualisierung aller Attribute und deren Werte (Abbildung 6.3, Mitte) erfolgt über die Methode `getSourceAttributeValues`.
- Um alle aktuellen Kontexte abzufragen bzw. zu aktualisieren (Abbildung 6.3, rechts), wird die Methode `getContextsForUser:` aufgerufen.

Die Funktionsweise der Demo-Applikation zeigt, wie die bereitgestellten Methoden der Programmierschnittstelle des lokalen Kontextdienstes von einer mobilen Appli-

kation genutzt werden können, um Kontextquellen, Attributwerte und Kontexte abzufragen.

6.4. Implementation des entfernten Kontextdienstes

Dieses Kapitel erläutert die Implementation aus der Perspektive des Dienstanbieters für den entfernten Kontextdienst. Abbildung 6.4 visualisiert die Gesamtarchitektur der Serverseite. Die einzelnen Komponenten werden im Folgenden aufgegriffen.

Der Abschnitt 6.4.1 behandelt den Web-Service, der die Anwendungslogik des entfernten Kontextdienstes darstellt. Dafür kommt eine Web-Applikation zum Einsatz, deren Aufbau, Ablauf und Schnittstelle beschrieben werden.

Das semantische Kontextmodell, welches das eigentliche Kontextwissen repräsentiert, wird im Abschnitt 6.4.2 dokumentiert. Die eingesetzte Ontologie und ihre Bereitstellung durch einen Triple-Store werden hier aufgegriffen.

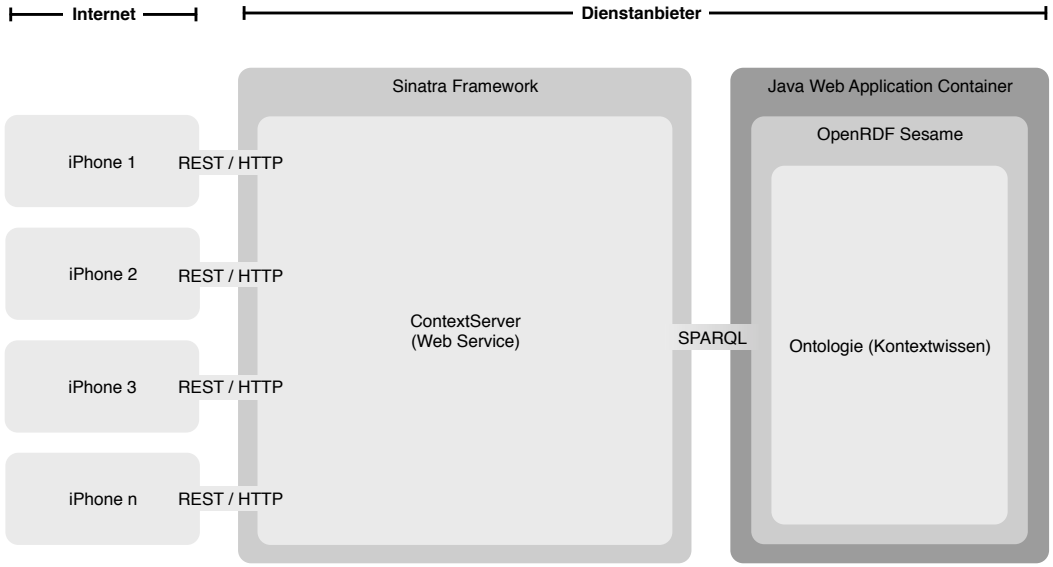

Abbildung 6.4.: Architektur der Referenzimplementation des entfernten Kontextdienstes

6.4.1. Kontextdienst als Web Service

Der entfernte Kontextdienst (in der Abbildung 6.4 mit `ContextServer` bezeichnet) hat die Aufgabe, Kontextanfragen zu beantworten. Er fungiert als Vermittler zwischen den mobilen Dienstnutzern und dem Triple-Store, welcher das semantische Kontextmodell beinhaltet.

Der als Web Service konzipierte und umgesetzte Kontextdienst implementiert das ressourcenorientierte *REST-Paradigma*. Kontexte können vom Dienstnutzer über die Ressource `context` angefordert werden, der Server ermittelt die entsprechenden Informationen aus dem Kontextmodell und liefert sie als *XML-Repräsentation* an den

Dienstnutzer zurück. Die Ermittlung der Kontextinformationen aus dem Kontextmodell erfolgt über *SPARQL-Abfragen*, deren Ergebnisse auf Basis der Attributwerte nach den in der Ontologie enthaltenen Regeln gefiltert werden.

Die Web-Applikation und ihr Quellcode wurden zusammen mit einer Kurzbeschreibung zur Inbetriebnahme im Internet veröffentlicht und lassen sich über die folgende URL beziehen:

`http://github.com/flxmllr/mobile-context-server/`

6.4.1.1. Sinatra Web-Applikation

Der Web Service wurde als Web-Applikation realisiert, welche, wie zuvor beschrieben, das Sinatra-Framework als Laufzeitumgebung verwendet. Auf der Abbildung 6.5 sind die Komponenten der Web-Applikation dargestellt.

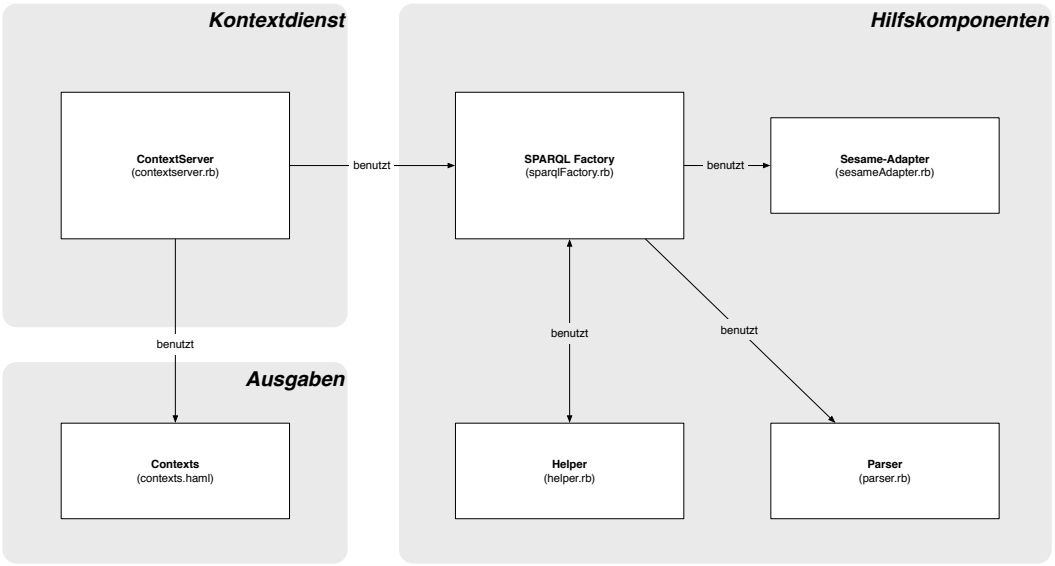

Abbildung 6.5.: Komponenten der Web-Applikation des entfernten Kontextdienstes

Der Web Service besteht aus einer Hauptkomponente, dem eigentlichen Kontextdienst (`ContextServer`), und einigen Hilfskomponenten. Die `SparqlFactory` generiert die SPARQL-Anfragen für die Abfrage des semantischen Kontextmodells, der `SesameAdapter` stellt die eigentliche Verbindung zum Triple-Store her, der `Helper` enthält einige Hilfsmethoden – wie Filter für die Kontexte – und der `Parser` verarbeitet die Antworten des Triple-Stores. Für die XML-Repräsentation der Ressource `context` wurde ein Template namens `Contexts` im *Haml*[15]-Format angelegt.

6.4.1.2. REST-Schnittstelle

Nachfolgend wird die REST-Schnittstelle beschrieben, die der entfernte Kontextdienst anbietet. Diese wird von den lokalen Kontextdiensten der Dienstnutzer zur

[15] *HTML Abstraction Markup Language* (`http://haml-lang.com/`)

Abfrage von Kontextinformationen genutzt.

Im Rahmen der Referenzimplementation wurde die Bereitstellung von zwei der in Kapitel 5.5.2.3 konzipierten Funktionen als Ressourcen implementiert, da diese zur Inbetriebnahme des Systems notwendig sind. Diese Ressourcen und ihre Aufrufe sowie die entsprechenden Rückgaben werden nachfolgend dargestellt.

Um alle im System hinterlegten Kontexte eines Benutzers abzufragen, wird die Ressource `contexts`, wie in der Übersicht in Tabelle 6.1 dargestellt, abgerufen.

Tabelle 6.1.: REST-API: Abfrage aller Kontexte eines Benutzers

URL	`http://[hostname]/contexts.xml`	
Methode	`GET`	
Parameter	`user=`	Benutzername
Rückgabe	`200 OK & XML (contexts.xml)`	
	`404 Not Found`	

Quellcode 6.3 zeigt ein Beispiel einer REST-Anfrage, welche den Benutzernamen als Parameter übergibt.

Quellcode 6.3: Beispiel einer REST-Abfrage aller Kontexte eines Benutzers

```
1 GET http://contextserver.felixmueller.name/contexts.xml?user=FelixMueller
```

Quellcode 6.4 stellt ein Beispiel einer Antwort des Dienstes auf diese Anfrage im XML-Format dar.

Quellcode 6.4: Beispiel einer REST-Rückgabe aller Kontexte eines Benutzers

```
1  <?xml version="1.0"?>
2  <contexts type="array">
3          <context>
4                  <timestamp type="datetime">Thu Jan 21 05:11:21 -0800 2010</timestamp>
5                  <type>LocationContext</type>
6                  <value>Home</value>
7          </context>
8          <context>
9                  <timestamp type="datetime">Thu Jan 21 05:11:21 -0800 2010</timestamp>
10                 <type>LocationContext</type>
11                 <value>Work</value>
12         </context>
13         <context>
14                 <timestamp type="datetime">Thu Jan 21 05:11:21 -0800 2010</timestamp>
15                 <type>TimeContext</type>
16                 <value>Morning</value>
17         </context>
18         <context>
19                 <timestamp type="datetime">Thu Jan 21 05:11:21 -0800 2010</timestamp>
20                 <type>DerivedContext</type>
21                 <value>MeetingAtWork</value>
22         </context>
23 </contexts>
```

Um die im System hinterlegten Kontexte, die auf die aktuelle Situation zutreffen, auf Basis übermittelter Attributwerte eines Benutzers abzufragen, wird die Ressource `contexts`, wie in der Übersicht in Tabelle 6.2 dargestellt, abgerufen.

Quellcode 6.5 zeigt ein Beispiel einer REST-Anfrage, welche den Benutzernamen, den gewünschten Kontexttyp und die Attribute als Parameter übergibt.

Tabelle 6.2.: REST-API: Abfrage der Kontexte eines bestimmten Typs auf Basis von Attributwerten

URL	`http://[hostname]/contexts.xml`	
Methode	`GET`	
Parameter	`user=`	Benutzername
	`type=`	Kontexttyp
	`attributes=`	Attribute als Hashtabelle
Rückgabe	`200 OK & XML (contexts.xml)`	
	`404 Not Found`	

Quellcode 6.5: Beispiel einer REST-Abfrage der Kontexte eines bestimmten Typs auf Basis von Attributwerten

```
1 GET http://contextserver.felixmueller.name/contexts.xml?user=FelixMueller&type=
      LocationContext&attributes={'longitude'=>'50.9','latitude'=>'6.9','time'=>'16:56:03'}
```

Quellcode 6.6 stellt ein Beispiel einer Antwort des Dienstes auf diese Anfrage im XML-Format dar.

Quellcode 6.6: Beispiel einer REST-Rückgabe der Kontexte eines bestimmten Typs auf Basis von Attributwerten

```
1 <?xml version="1.0"?>
2 <contexts type="array">
3         <context>
4                 <timestamp type="datetime">Thu Jan 21 05:11:21 -0800 2010</timestamp>
5                 <type>LocationContext</type>
6                 <value>Work</value>
7         </context>
8 </contexts>
```

6.4.2. Semantisches Kontextmodell

Dieser Abschnitt geht auf die Repräsentation des eigentlichen Kontextwissens ein. Um dieses Wissen zu modellieren, wurde im Rahmen der Implementierung eine *Ontologie* erstellt, welche mehrere Kontexte[16] als Beispiele enthält (siehe `Ontologie` in Abbildung 6.4). Die Bereitstellung der Ontologie erfolgt über den Triple-Store *OpenRDF Sesame*.

6.4.2.1. Kontext-Ontologie

Die Ontologie, die im Rahmen dieser Referenzimplementation erstellt wurde, besitzt die URI `http://contextserver.felixmueller.name/context`. Sie fungiert als Beispiel zur Abbildung des Kontextwissens in Form semantischer Tripel und zur Demonstration des implementierten Kontextdienstes.

Um zu verdeutlichen, wie sich konkrete Kontexte formulieren lassen, wurde in Abbildung 6.6 ein Auszug aus der Ontologie visualisiert, welcher drei Kontexttypen sowie drei Beispiele für konkrete Kontexte enthält. Die *Klassen*, *Eigenschaften* und *Individuen* der Ontologie werden nachfolgend beschrieben.

[16] Siehe Anhang B: Kontexte der Beispiel-Ontologie

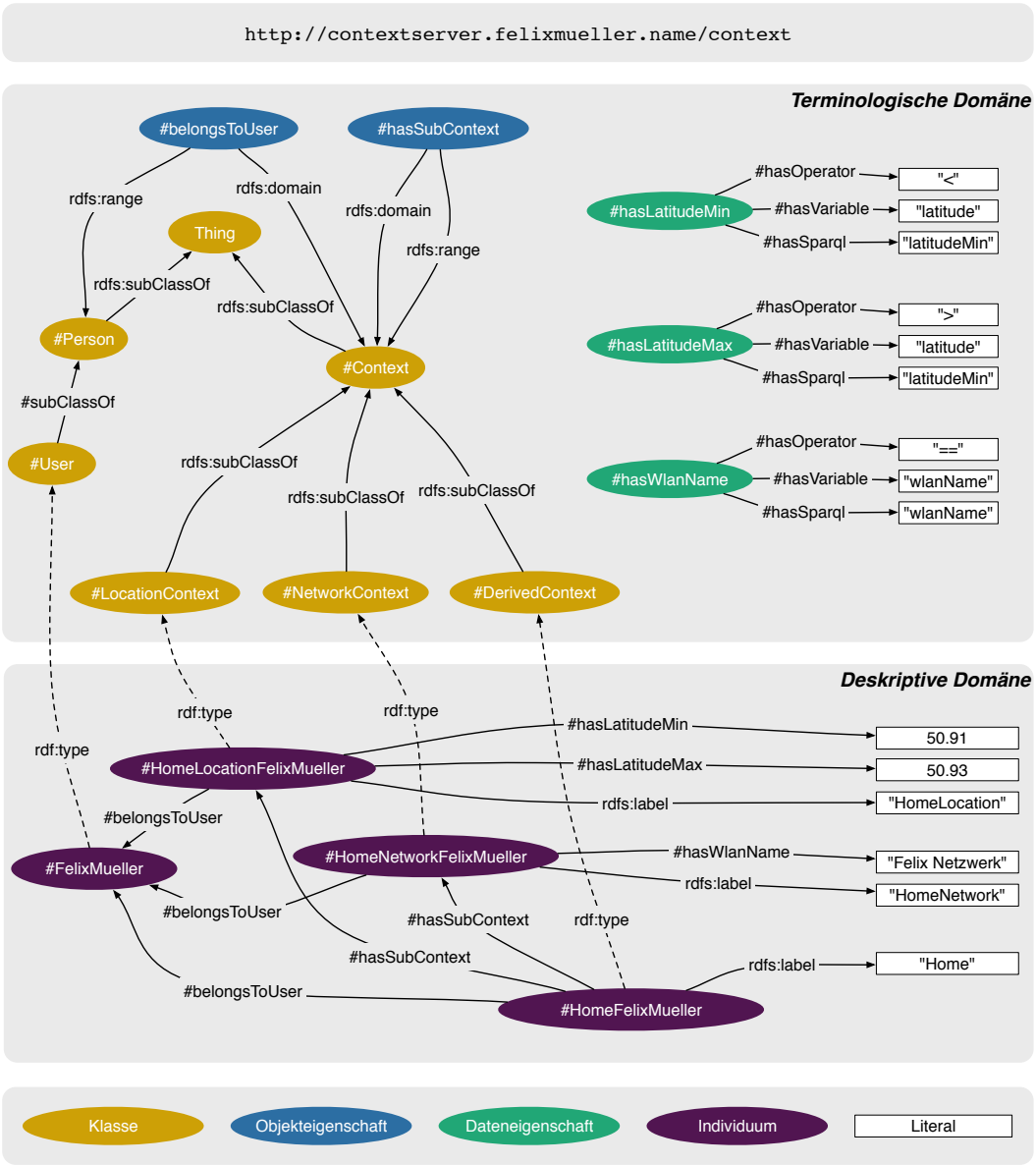

Abbildung 6.6.: Beispielauszug aus der Ontologie der Referenzimplementation

Klassen

Da jeder Kontext benutzerabhängig sein kann, wurde eine Klasse `#User` definiert, über die individuelle Benutzer modelliert werden können.

Die Oberklasse `#Context` steht für Kontexte jeglichen Typs. Spezifische Kontexttypen werden durch Unterklassen ausgedrückt. Im abgebildeten Beispiel sind es die Klassen `#LocationContext` *(für ortsbezogene Kontexte)*, `#NetworkContext` *(für netzwerkbezogene Kontexte)* und `#DerivedContext` *(für abgeleitete Kontexte, also High Level-Kontexttypen)*.

Objekteigenschaften

Zur Abbildung der Benutzerabhängigkeit von Kontextobjekten wurde die Eigen-

schaft `#belongsToUser` modelliert, über welche Kontextobjekte Benutzerobjekten zugeordnet werden können.

Über die Eigenschaft `#hasSubContext` können Kontexte höherer Ebenen modelliert werden, indem Kontextobjekten ein oder mehrere untergeordnete Kontextobjekte zugeordnet werden.

Dateneigenschaften

Um die Regeln für *Low Level*-Kontexttypen auszudrücken, wurden Dateneigenschaften für alle relevanten Attribute definiert. Im dargestellten Auszug stehen dafür die Eigenschaften `#hasLatitudeMin` *(minimaler Breitengrad)*, `#hasLatitudeMax` *(maximaler Breitengrad)* und `#hasWlanName` *(SSID des WiFi-Netzwerks)*, die sich auf die Attribute `latitude` und `wlanName` beziehen. Über diese Eigenschaften wird bei der Kontextermittlung die Filterung nach zutreffenden Kontexten durchgeführt. Sie umfassen jeweils einen Vergleichsoperator (`#hasOperator`) sowie Bezeichnungen für die Zuordnung im Rahmen der SPARQL-Abfragen (`#hasVariable` und `#hasSparql`).

Individuen

Konkrete Benutzer und Kontexte werden über die Individuen in der Ontologie modelliert. Das dargestellte Beispiel enthält einen Benutzer (`#FelixMueller`) und drei Kontexte.

Der Kontext `#HomeLocationFelixMueller` mit dem Typ `#LocationContext` beschreibt die geografische Position des Heimat des Benutzers `#FelixMueller`. Er entspricht einem *Low Level*-Kontexttyp und wird definiert durch die Bedingungen `#hasLatitudeMin:50.91` und `#has LatitudeMax:50.93`, er tritt also ein wenn der Breitengrad zwischen 50.91 und 50.93 liegt.

Der Kontext `#HomeNetworkFelixMueller` vom Typ `#NetworkContext` beschreibt die Erreichbarkeit des Heimnetzes des Benutzers `#FelixMueller` und ist ebenfalls ein *Low Level*-Kontexttyp. Dieser hat die Bedingung `#hasWlanName:"Felix Netzwerk"`. Er ist also bei einem erreichbaren WiFi-Netzwerk mit der SSID „Felix Netzwerk“ zutreffend.

Um die Modellierung von *High Level*-Kontexttypen zu demonstrieren wurde ein weiterer Kontext vom Typ `#DerivedContext` mit dem Namen `#HomeFelixMueller` angelegt. Dieser beschreibt die Lokation des mobilen Geräts am Heimatort und ist zutreffend, wenn die Kontexte `#HomeLocationFelixMueller` und `#HomeNetwork-FelixMueller` eintreten. Zur Ableitung dieses Kontextes aus den untergeordneten Kontexten wurden die Bedingungen `#hasSubContext:#HomeLocationFelixMueller` und `#hasSubContext:#HomeNetworkFelixMueller` formuliert.

Über die Ausdrücke `#belongtsToUser:#FelixMueller` sind alle beschriebenen Kontexte dem Benutzer `#FelixMueller` zugeordnet.

6.4.2.2. Triple-Store

Zur Bereitstellung und Abfrage des in der Ontologie enthaltenen Kontextwissens kommt ein *Triple-Store* zum Einsatz. Dieser besteht aus einer speziellen Datenbank zur Speicherung und zum Abruf von RDF-Metatataten in Form von Tripeln. Für die praktische Umsetzung im Rahmen dieser Arbeit wurde ein Produkt benötigt, welches nach Möglichkeit kostenlos ist und unkompliziert aufgesetzt werden kann. Die Wahl fiel deshalb auf *OpenRDF Sesame* (siehe `OpenRDF Sesame` in der Abbildung 6.4).

Sesame ist ein in Java entwickelter Open Source Triple-Store. Er wird als Java Web-Applikation ausgeliefert und wird in einem Java Servlet Container betrieben (siehe `Java Web Application Container` in der Abbildung 6.4). Für die Tests im Rahmen dieser Arbeit wurde das Produkt *Tomcat* der Apache Foundation als Servlet-Container eingesetzt.

Die aktuelle Ontologie, die das Kontextwissen repräsentiert, kann im `OWL`[17]-Format über die Administrationsoberfläche in den Triple-Store geladen werden. Für semantische Anfragen bietet Sesame einen SPARQL-Endpunkt an, der als Web Service über HTTP angesprochen werden kann. Diesen benutzt die in Kapitel 6.4.1.1 beschriebene Sinatra Web-Applikation für die Abfragen der Kontextinformationen.

6.5. Funktionsweise und Ablauf

Nachdem die einzelnen Komponenten des implementierten Systems nun beschrieben wurden, soll dieser Abschnitt die Funktionsweise sowie den Ablauf der Ermittlung von Kontextinformationen durch einen mobilen Dienstnutzer, beispielhaft anhand von UML-Sequenzdiagrammen, veranschaulichen.

6.5.1. Ablauf der mobilen Komponenten

Zunächst wird der Ablauf aus der Perspektive des mobilen Geräts beschrieben. Die Abbildung 6.7 zeigt ein Sequenzdiagramm, auf dem eine mobile Applikation und die statische Bibliothek mit dem Kontextdienst für das iPhone aufgetragen ist. Die einzelnen Schritte, die nun erläutert werden, sind hervorgehoben und mit den Buchstaben **A** bis **E** gekennzeichnet.

Schritt A Nachdem die mobile Applikation gestartet wurde, wird eine Instanz des Kontextdienstes (`ContextService`) erzeugt, um den Kontextdienst verwenden zu können. Dieser steht anschließend für weitere Anfragen zur Verfügung.

Schritt B Die Applikation fordert eine Liste aller verfügbaren Kontextquellen vom Kontextdienst an. Diese Liste wird als Verzeichnis an die Applikation zurück gegeben.

[17] *Web Ontology Language* (`http://www.w3.org/TR/owl-features/`)

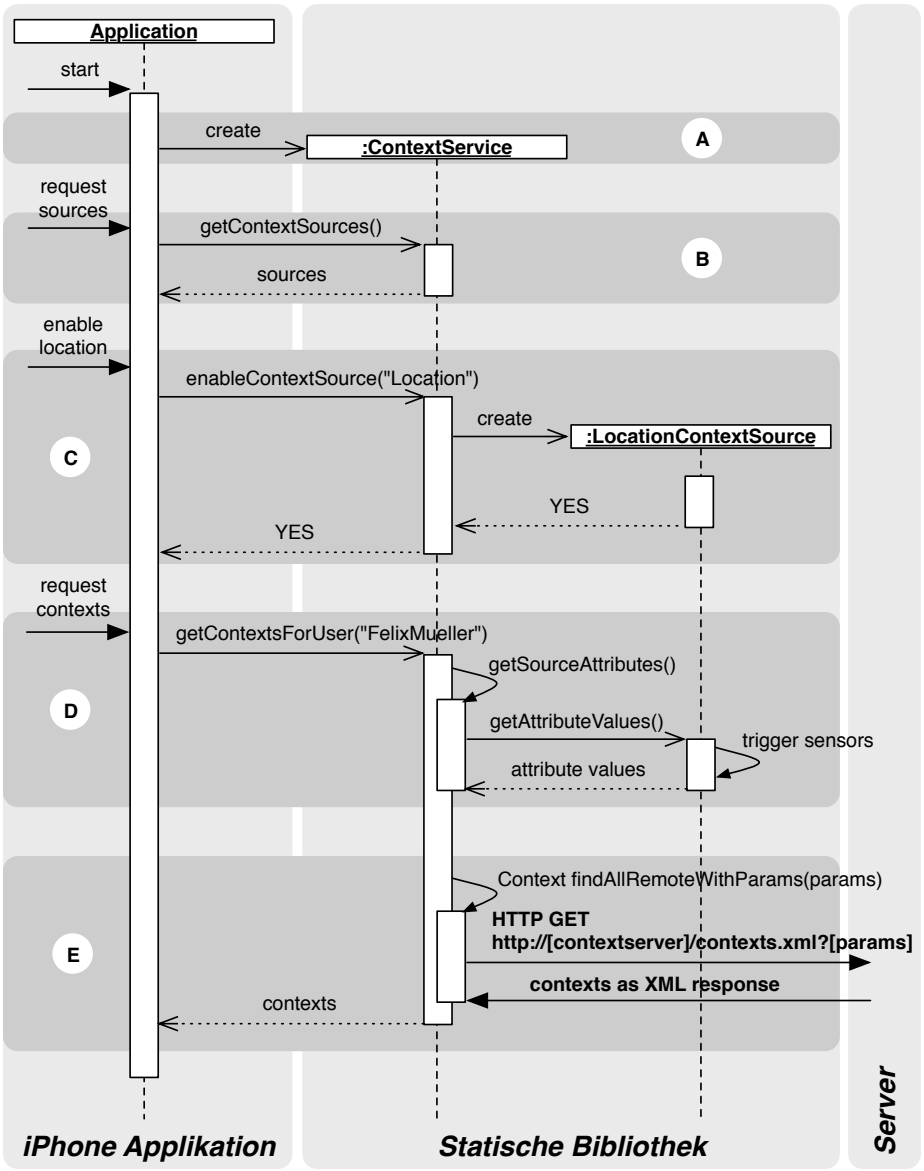

Abbildung 6.7.: Sequenzdiagramm der mobilen Komponenten

Schritt C Die Applikation leitet die Aktivierung der Kontextquelle `Location` ein. Daraufhin erzeugt der Kontextdienst eine Instanz der Kontextquelle `LocationContextSource`.

Schritt D Die Applikation fordert alle Kontexte für den Benutzer `FelixMueller` an. Der Kontextdienst prüft daraufhin die Attribute aller aktiven Kontextquellen und fordert die Werte dieser Attribute von den Kontextquellen an (im Falle dieses Beispiels von der Kontextquelle `LocationContextSource`). Die Quellen ermitteln ihre Attributwerte (hier durch Abfragen der Geoinformationen durch die iPhone Location API). Ein Verzeichnis mit den erhobenen Werten wird an den Kontextdienst zurück gegeben.

Schritt E Nachdem die Attributwerte ermittelt wurden, besteht der letzte Schritt in der Abfrage der Kontexte vom entfernten Kontextdienst auf Basis dieser Werte. Dafür wird zunächst eine Anfrage-URL erzeugt, die alle Attribute und deren Werte sowie den Benutzernamen als Parameter enthält. Diese wird über das `ObjectiveResource`-Framework als HTTP GET-Anfrage an den entfernten Kontextdienst gestellt. Die Rückgabe des Dienstes als XML-Repräsentation wird durch `ObjectiveResource` in `Context`-Objekte umgewandelt, die schlussendlich als Liste an die aufrufende mobile Applikation zurück gegeben werden.

6.5.2. Ablauf des entfernten Kontextdienstes

Dieser Abschnitt beschreibt die Funktionsweise aus der Perspektive des entfernten Kontextdienstes. Die Abbildung 6.8 zeigt den Ablauf zwischen den Komponenten als Sequenzdiagramm. Die einzelnen Schritte sind wieder hervorgehoben und mit den Buchstaben **A** bis **C** bezeichnet.

Schritt A Der Ablauf wird mit einer REST-Anfrage eines Dienstnutzers angestoßen[18]. Das Hauptprogramm (`ContextServer`) leitet die Anfrage an die Komponente `SparqlFactory` weiter. Diese erzeugt eine SPARQL-Abfrage, um alle Prädikate des angegebenen Typs vom Triple-Store anzufordern. Damit wird eine Vorauswahl aller im Kontextmodell vorhandenen Prädikate getroffen, welche zur Modellierung von Kontexten verwendet wurden. Die Abfrage wird über den `SesameAdapter` per HTTP GET an den Sesame Triple-Store gestellt. Dieser liefert die Resultate im JSON[19]-Format zurück. Die JSON-Repräsentation der Prädikate wird schließlich durch den Parser verarbeitet.

Schritt B Nun wird eine zweite SPARQL-Abfrage zur Ermittlung der Kontexte generiert, welche neben Benutzer, Kontexttyp und den Attributen die vorher ermittelten Prädikate enthält. Diese wird wieder per HTTP GET an den Sesame Triple-Store gestellt. Die zurück erhaltenen Kontexte werden wie zuvor durch den Parser verarbeitet.

Schritt C Der finale Schritt besteht in der Filterung der Kontexte nach den in der Ontologie modellierten Regeln. Diese Filterung wird durch den `Helper` durchgeführt. Nachdem dieser Schritt abgeschlossen ist, werden die zutreffenden Kontexte zurück an die aufrufenden Methoden gereicht und schließlich durch das Template `contexts.haml` als XML-Repräsentation an den Dienstnutzer zurückgegeben[20].

[18] Ein Beispiel einer Anfrage zeigt Quellcode 6.5
[19] *JavaScript Object Notation* (`http://www.json.org/`)
[20] Ein Beispiel einer Rückgabe zeigt Quellcode 6.6

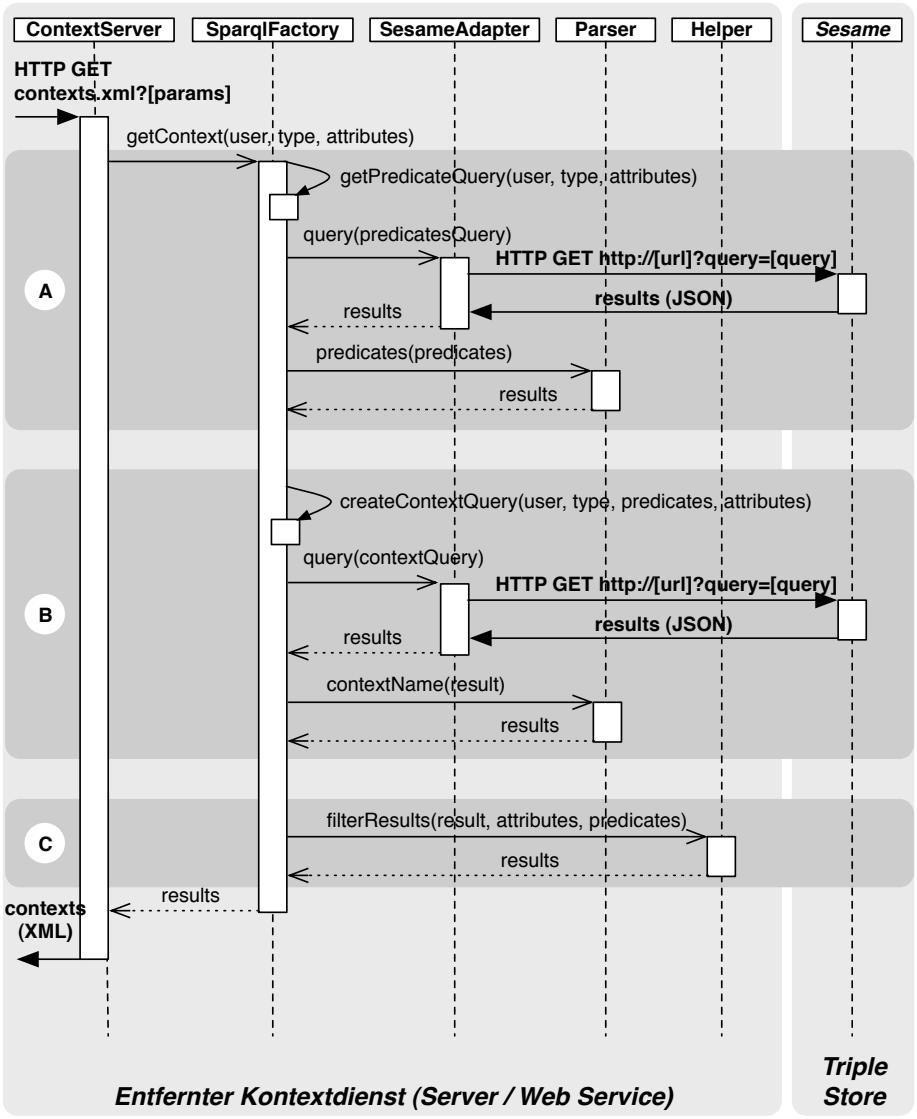

Abbildung 6.8.: Sequenzdiagramm des entfernten Kontextdienstes

6.6. Kritische Betrachtung

Zum Abschluss des Kapitels wird eine kritische Betrachtung des umgesetzten Kontext-Rahmenwerks vorgenommen.

Die Referenzimplementation veranschaulicht, dass sich die auf abstrakter Ebene festgelegten Strukturen des Konzepts unter Berücksichtigung der Rahmenbedingungen, Regeln und Zielvorgaben mit begrenztem Aufwand in ein lauffähiges Rahmenwerk umsetzen lassen. Das System wurde so weit implementiert, dass sich die Kernfunktionen unter bestimmten Bedingungen in der Praxis einsetzen lassen.

Die Verwendung des iPhone OS als mobile Plattform hat zu einigen Einschränkungen geführt, weshalb das Konzept nicht direkt umgesetzt werden konnte. Diese Beschränkungen wurden durch den Einsatz einer statischen Bibliothek teilweise umgangen. Dennoch wäre die Möglichkeit eines unabhängigen Dienstes zur Kontext-

verwaltung auf dem mobilen Endgerät wünschenswert, anstatt die Programmlogik dieses Dienstes in jede mobile Applikation einbinden und einkompilieren zu müssen.

Die Programmierschnittstelle für mobile Applikationen wurde zur Abfrage von Kontextinformationen implementiert. Das Einfügen von Kontexten durch Applikationen ist zwar vorgesehen, jedoch nicht umgesetzt worden.

Wichtige Herausforderungen sind die Themen Sicherheit und Datenschutz, die das System nicht berücksichtigt. Weiterhin gibt es keine Möglichkeit, Kontextdaten auf dem mobilen Gerät zwischenzuspeichern, um bei fehlenden Netzwerkverbindungen dennoch Kontextinformationen bereitstellen zu können und den Akku des Geräts zu schonen.

Ebenso fehlen Attribute zur Beschreibung der Qualität der Kontextinformationen, die in Kapitel 2.4 erörtert wurden. Das umgesetzte System verfügt zwar über erste Vorbereitungen im Bereich der Datenstrukturen zur Speicherung von Kontexten und Attributen, diese sind jedoch nicht bis zur Programmierschnittstelle hindurch implementiert worden.

Eine Zusammenfassung der gewonnenen Erkenntnisse und der daraus resultierenden Schlussfolgerungen sowie ein Ausblick erfolgen im nachfolgenden und letzten Kapitel dieser Masterarbeit.

7. Zusammenfassung und Ausblick

Diese Masterarbeit hat sich sich mit der Fragestellung beschäftigt, ob und inwieweit sich ein generisches, anwendungsunabhängiges und erweiterbares Rahmenwerk entwickeln lässt, welches Kontextinformationen repräsentiert und verwaltet, um mobilen Applikationen eine möglichst einfache Kontextadaption zu ermöglichen.

Die Ergebnisse der Arbeit sollen in diesem letzten Kapitel diskutiert werden. Zunächst werden die behandelten Themengebiete zusammengefasst. Der darauf folgende Abschnitt befasst sich mit den Fragestellungen, ob und inwiefern die in Kapitel 1.2 definierten Ziele erreicht wurden und welche Schlussfolgerungen aus den Ergebnissen dieser Arbeit gezogen werden können. Schlussendlich wird ein Ausblick darauf gegeben, an welchen Stellen eine Anknüpfung für themenverwandte Arbeiten möglich ist, um den Gegenstand dieser Masterarbeit auf Basis der bisherigen Ergebnisse weiter zu führen.

7.1. Zusammenfassung

Um sich den Antworten auf die Forschungsfrage anzunähern, erfolgte die Auseinandersetzung mit der Thematik in mehreren Phasen.

Nach einer Darstellung relevanter Grundlagen wurden zunächst die Möglichkeiten zur Modellierung und Repräsentation des Kontextes mobiler Nutzungsszenarien betrachtet. Es ging also um die Art und Weise, wie der mobile Kontext seitens der Software durch ein geeignetes Modell abgebildet werden kann. Nach Untersuchung und bewertendem Vergleich zahlreicher Ansätze aus der Literatur wurde der ontologiebasierte, semantische Ansatz für am geeignetsten befunden, um ein generisches und flexibles Rahmenwerk zur Kontextverwaltung zu entwickeln. Die semantische Kontextmodellierung wurde deshalb im Anschluss daran genauer betrachtet. Es stellte sich heraus, dass ontologiebasierte Ansätze sowohl automatisches als auch manuelles *Context Reasoning* ermöglichen, ihr Einsatz im mobilen Kontext jedoch Beschränkungen unterlegen ist. Um semantische Technologien trotz Limitationen portabler Hardware in mobilen Umgebungen einsetzen zu können, wurde zur Verwaltung des Kontextmodells eine Architektur in Erwägung gezogen, welche über einen entfernten Kontextserver verfügt, der das Kontextmodell vorhält.

Im nächsten Schritt wurden jene Systeme und Rahmenwerke zum Thema gemacht, die mobilen Applikationen Kontextinformationen bereitstellen, um ihnen eine Adaption des Kontextes zu erleichtern. Die Anforderungen an solche Systeme

betreffen hauptsächlich die Verwaltung und den Austausch von Kontextinformationen, eine Trennung von Management- und Modellkomponenten, Erweiterbarkeit und Skalierbarkeit, Kompaktheit und Portierbarkeit, Sicherheit und Privatsphäre sowie die Berücksichtigung von Kriterien der Kontextqualität. Aus den gewonnenen Erkenntnissen wurde im Anschluss das Konzept eines generischen Rahmenwerks ausgearbeitet. Dieses beschreibt zunächst die Architektur des Systems, welche die Konzepte der Middleware-Infrastrukturen und der entfernten Kontextserver kombiniert. Weiterhin wurden die Charakteristika der Schnittstellen zwischen den Systemkomponenten bestimmt, welche alle Funktionen beschreiben, die die Komponenten nach außen bereitstellen sollten.

Das Konzept diente als Grundlage für die im finalen Schritt durchgeführte prototypische Realisierung in Form einer Referenzimplementation. Sie besaß das Ziel, zu veranschaulichen, ob sich die auf abstrakter Ebene festgelegten Strukturen des Konzepts unter Berücksichtigung der Rahmenbedingungen, Regeln und Zielvorgaben mit begrenztem Aufwand in ein lauffähiges System umsetzen lassen. Bei der Wahl der Technologien wurde Wert auf den Einsatz aktueller Plattformen und Werkzeuge gelegt. Die Implementierung sowie die Funktionsweise der einzelnen Komponenten des Systems wurden im praktischen Teil der Arbeit eingehend dokumentiert.

7.2. Ergebnisse der Arbeit

Dieser Abschnitt befasst sich nun mit der Darstellung der Ergebnisse, die aus der Anfertigung dieser Masterarbeit resultieren.

Die vorliegende Arbeit macht deutlich, dass die Bereiche *Mobile Computing* und *Context-aware Computing* große Herausforderungen an die Softwareentwicklung stellen und daher aktive Forschungsarbeit erfordern. Im Kontext dieser Arbeit wurden einige dieser Herausforderungen tangiert. So wurde die Modellierung des Kontextes als eine erste Aufgabe thematisiert. Erste Erkenntnisse zur Realisierbarkeit der Modellierung von Kontextdaten wurden aus den Ergebnissen zahlreicher Arbeiten gewonnen, die sich diesem Thema mit sehr unterschiedlichen Ansätzen genähert haben. Eine weitere Herausforderung stellt die Abbildung der Modellierungskonzepte auf konkrete Systeme dar. Aus diesem Grund existieren diverse Ansätze für Rahmenwerke zur Kontextrepräsentation, die unterschiedliche Techniken anwenden, um eine Lösung dieses Problems herbeizuführen.

Um die zuvor definierten, eigenen Ziele zu erreichen, wurde mit dieser Arbeit ein Versuch unternommen, ein generisches, anwendungsabhängiges und erweiterbares Rahmenwerk zur Repräsentation des Kontextes für mobile Applikationen zu erarbeiten und praktisch umzusetzen. Der generische Charakter wurde durch eine bewusste Abstraktion von Kontextinformationen bewerkstelligt. Der Fokus wurde anstelle auf konkrete Kontexttypen auf die gemeinsamen Eigenschaften dieser Informationstypen gelegt.

Die Erweiterbarkeit des Systems im Hinblick auf diverse Quellen für Kontextinformationen wurde realisiert, indem in Anlehnung an die Ansätze verschiedener Autoren das Konzept der Kontextquellen eingeführt wurde. Diese kapseln die Erhebung der Kontextinformationen und ermöglichen eine Aggregation von Informationen aus diversen Quellen in einem einheitlichen Format. Diese Vereinheitlichung wurde über definierte Schnittstellen verwirklicht, die diesen Austausch erst ermöglichen.

Das umgesetzte Rahmenwerk ist in der Lage, verschiedene Konstellationen von Kontexten abzubilden. Dazu gehören *Low Level* sowie *High Level*-Kontexttypen, die sich mithilfe von Regeln aus anderen Kontexten ableiten lassen. Durch die Wahl eines ontologiebasierten Ansatzes wurde der größte Teil der Logik in das semantische Kontextmodell verlagert. Durch die Möglichkeiten der semantischen Auszeichnungen wird eine große Flexibilität erreicht. Eine Definition eigener Operatoren, etwa zur Erstellung von Regeln, wird durch einfache Ausdrücke in der Ontologie möglich.

Ein großer Vorteil des Einsatzes eines semantischen Kontextmodells gegenüber anderen Ansätzen ist die Unterstützung von *Context Reasoning*, wie in Kapitel 4.4.1 erläutert wurde. Mit Hilfe von Ontologie-Reasonern können automatisch implizite Fakten aus explizit modellierten Fakten abgeleitet werden. Von dieser Möglichkeit wurde im Rahmen dieser Arbeit jedoch aus Gründen der Komplexitätsreduktion in der Implementation kein Gebrauch gemacht. Statt dessen wurde zur Veranschaulichung auf benutzerdefiniertes Reasoning zurück gegriffen, welches abgeleitete Kontexte explizit durch Regeln definiert.

Die Referenzimplementation ermöglicht ausschließlich lesenden Zugriff auf die Informationen des Kontextmodells. Das Konzept sieht jedoch ebenfalls einen schreibenden Zugriff vor, der die Konfiguration ermöglichen soll. Wie eingangs beschrieben, ist das Thema Sicherheit und Datenschutz im Rahmen der Ausarbeitung nicht behandelt worden. Eine nachträgliche Implementierung der Sicherheitskonzepte erscheint jedoch durch den Einsatz standardisierter Plattformen und Protokolle durchaus möglich.

Auswirkungen auf den Kern des Systems wird jedoch die Berücksichtigung der in Kapitel 2.4 beschriebenen Qualitätskriterien haben. Techniken des *Context Reasoning* müssen diese Informationen nämlich explizit verarbeiten. Ein Beispiel dafür ist die Ableitung eines höherwertigen Kontextes aus untergeordneten Kontexten. So ergibt sich die Zuverlässigkeit des abgeleiteten Kontextes aus der Zuverlässigkeit der untergeordneten Kontexte.

Das Ziel der Realisierbarkeit mithilfe aktueller Technologien und Standards im Bereich mobiler Geräte sowie Infrastrukturen und der Modellierung wurde mit dieser Arbeit erreicht. Sowohl die iPhone-Plattform als auch die eingesetzten Rahmenwerke, Protokolle und Programmiersprachen entsprechen dem aktuellen Stand der Entwicklung.

7.3. Ausblick und weitere Schritte

Abschließend wird mit diesem Abschnitt ein Ausblick darauf gegeben, an welchen Stellen eine Anknüpfung für themenverwandte Arbeiten und damit die Fortführung relevanter Forschung möglich ist.

Die Bereiche des *Mobile Computing* und des *Context-aware Computing* sind vergleichsweise junge Forschungsgebiete. Mobile Geräte entwickeln sich erst jetzt in die Richtung leistungsfähiger Computer, immer neue Sensortechnologien sind in der Entstehung. Das Potenzial der Anwendungsszenarien auf Basis dieser Techniken wird erst langsam deutlich. Aus diesen Gründen ist eine weiterführende Bearbeitung des Themas dieser Masterarbeit durchaus empfehlenswert. Dabei sind mehrere Anknüpfungspunkte denkbar:

Zum einen ist die Erarbeitung von theoretischen Konzepten zur Gewährleistung von Sicherheit und Datenschutz anzustreben. Ein Beispiel ist die Untersuchung der Möglichkeiten zur Benutzerauthentifizierung. Zum anderen sind Modifikationen der Architektur in eine Richtung vorzuschlagen, in der externe Kontextquellen direkt an den Kontextserver angebunden werden können. Dadurch würden Kontexte nicht mehr ausschließlich durch die von mobilen Geräten erfassten Attribute entstehen, sondern auch aus der Umgebung heraus. Ein Netzwerk aus mehreren Kontextservern, die ihr Kontextwissen in Form von Ontologien untereinander Teilen würde der Idee des *Semantic Web* noch stärker Rechnung tragen.

In praktischer Hinsicht bietet sich zunächst die Erweiterung des Prototyps um den schreibenden Zugriff auf das Kontextmodell an. Weiterhin gibt es für die mobile Seite Optimierungspotenzial, um den effizienten Praxiseinsatz zu ermöglichen. Ein intelligenter Caching-Mechanismus etwa, könnte die Probleme beseitigen, dass der Kontextserver bei schlechten Netzwerkverbindungen nicht erreichbar sein kann und dass ständiges Abfragen des Servers die Batterielaufzeit der mobilen Geräte reduziert.

Zu Evaluationszwecken des Systems könnte der Bereich der Kontextquellen ausgeweitet werden, indem weitere dieser Quellen unterschiedlicher Ausprägung implementiert werden. Dadurch würden umfangreichere Testszenarien geschaffen, auf Basis derer weitere Untersuchungen durchgeführt werden können.

Danksagung

Mein ganz besonderer Dank richtet sich an alle Menschen, die mich bei der Anfertigung dieser Masterarbeit unterstützt haben:

- Für seine Tipps, Hinweise und Hilfestellungen in Fragen der praktischen Anwendung der semantischen Modellierung danke ich meinem Kommilitonen *Stephan Pavlovic*.
- Für die Betreuung dieser Arbeit danke ich meinen Professoren an der Fachhochschule Köln, *Prof. Dr. Kristian Fischer* und *Prof. Dr. Gerhard Plaßmann*.
- Weiterhin bedanke ich mich bei allen *Freunden*, *Familienmitgliedern* und *Kommilitonen*, die mich bei der Fehlersuche in dieser Arbeit unterstützt haben.

Literaturverzeichnis

[Ay07] Ay F.: *Context Modeling and Reasoning using Ontologies*, University of Technology Berlin, 2007. `http://www.ponnuki.de/cmaruo/cmaruo.pdf` (Abruf 10.01.2010)

[Bac97] Bacon J., Bates J., Halls D.: *Location-oriented multimedia*, IEEE Personal Communications, Vol. 4 (5), S. 48-57, 1997.

[Bald06] Baldauf M., Dustdar S., Rosenberg F.: *A Survey on Context-Aware Systems*, International Journal of Ad Hoc and Ubiquitous Computing, Forthcoming, 2006.

[Ball07] Ballard B.: *Designing the Mobile User Experience*, Wiley Blackwel, 2007.

[Bard05] Bardram J. E.: *Design, Implementation, and Evaluation of the Java Context Awareness Framework (JCAF)*, Technical report, Centre for Pervasive Healthcare, 2005.

[Bau03] Bauer J.: *Identification and Modeling of Contexts for Different Information Scenarios in Air Traffic*, Diplomarbeit, Technische Universität Berlin, 2003.

[Bau06] Bauer H. H., Reichardt T., Schüle A.: *Was will der mobile Nutzer? Forschungsergebnisse zu den Anforderungen von Nutzern an kontextsensitive Dienste*, Haasis K., Heinzl A., Klumpp D. (Hrsg., 2006): Aktuelle Trends in der Softwareforschung, Heidelberg, 2006.

[Baus02] Baus J., Krüger A., Wahlster W.: *A resource-adaptive mobile navigation system*, In: Proceedings of International Conference on Intelligent User Interfaces, San Francisco, ACM Press, 2002.

[Bea04] Beale R., Lonsdale P.: *Mobile Context-aware Systems: The Intelligence to Support Tasks and Effectively Utillise Resources*, MobileHCI, 2004.

[Beigl99] Beigl M., Schmidt A., Gellersen H. W.: *There is more to context than location*, Computers and Graphics, 1999.

[Bis04] Bisgaard J. J., Heise M., Steffensen C.: *How is Context and Context-awareness Defined and Applied? A Survey of Context-awareness*, Aalborg University, 2004.

[Bou97] Bouzy B., Cazenave T.: *Using the Object Oriented Paradigm to Model Context in Computer Go*, In: Proceedings of Context '97, Rio, Brasilien, 1997.

[Bro96] Brown, P. J.: *The stick-e document: a framework for creating context-aware applications*, In: Proceedings of Electronic Publishing, S. 259-272, 1996. `http://cajun.cs.nott.ac.uk/compsci/epo/papers/volume8/issue2/2point1.pdf` (Abruf: 10.01.2010)

[Bro97] Brown P. J., Bovey J. D., Chen X.: *Context-aware Applications: From the Laboratory to the Marketplace*, In: IEEE Personal Communications, 1997.

[Buch03] Buchholz T., Küpper A., Schiffers M.: *Quality of Context Information: What it is and why we need it*, In: Proceedings of the 10th International Workshop of the HP OpenView University Association, Vol. 2003, Genf, Schweiz, 2003.

[Chen00] Chen G., Kotz D.: *A survey of context-aware mobile computing research*, Technical Report, 2000.

[Chen04] Chen H.: *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*, University of Maryland, Baltimore County, 2004.

[CheFi04] Chen H., Finin T., Joshi, A.: *Semantic Web in the Context Broker Architecture*, In: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications, 2004.

[Chev99] Cheverst K., Davies N., Mitchell K., Friday A.: *Design of an Object Model for a Context-Sensitive Tourist Guide*, Elsevier Science Ltd., Computers & Graphics, 1999.

[Corr07] Corradi A., Montanari R., Toninelli A.: *Adaptive Semantic Middleware for Mobile Environments*, Journal of Networks, Vol. 2, University of Bologna, Italien, 2007.

[David05] David P., Ledoux T.: *WildCAT: a generic framework for context-aware applications*, In: Proceedings of the 3rd international Workshop on Middleware For Pervasive and Ad-Hoc Computing, Grenoble, Frankreich, 2005.

[Dey99] Dey A. K., Abowd G. D., Wood A.: *CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services*, Knowledge-Based Systems, 1999.

[Dey00] Dey A. K., Abowd G. D.: *Towards a Better Understanding of Context and Context-Awareness*, Workshop on The What, Who, Where,

When, and How of Context-Awareness, Conference on Human Factors in Computer Systems, 2000.

[Dey01] Dey A. K., Abowd G. D., Salber D.: *A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications*, In: Human-Computer Interaction, 2001.

[Du08] Du W., Wang L.: *Context-Aware Application Programming for Mobile Devices*, In: Proceedings of the 2008 C3S2E conference, Kanada, 2008.

[Dud97] Duden: *Das Fremdwörterbuch*, 6. überarbeitete und erweiterte Auflage, 1997.

[Gray01] Gray P., Salber D.: *Modelling and Using Sensed Context Information in the design of Interactive Applications*, In: Proceedings of 8th IFIP International Conference on Engineering for Human-Computer Interaction, Toronto, Kanada, 2001.

[Gu05] Gu T., Pung H. K., Zhang D. Q.: *A Service-Oriented Middleware for Building Context-Aware Services*, In: Journal of Network and Computer Applications, Vol. 28 (1), S. 1-18, 2005.

[Gu07] Gu T., Kwok Z., Koh K. K., Pung H. K.: *A mobile framework supporting ontology processing and reasoning*, In: 2nd Workshop on Requirements and Solutions for Pervasive Software Infrastructure, Österreich, 2007.

[Haek06] Häkkilä J., Mäntyjärvi J.: *Developing Design Guidelines for Context-Aware Mobile Applications*, In: Proceedings of the 3rd international conference on Mobile technology, applications & systems, Bangkok, Thailand, 2006.

[Han03] Hansmann U., Merk L., Nicklous M. S.: *Pervasive Computing*, Springer Verlag, 2003.

[Held02] Held A., Buchholz S., Schill A.: *Modeling of context information for pervasive computing applications*, In: Proceedings of SCI 2002 / ISAS 2002, 2002.

[Hess03] Hess, C. K., Campbell, R. H.: *An Application of a Context-Aware File System*, Personal and Ubiquitous Computing, 2003.

[Hof03] Hofer T., Schwinger W., Pichler M., Leonhartsberger G., Altmann J., Retschitzegger W.: *Context-Awareness on Mobile Devices - the Hydrogen Approach*, In: Proceedings of the 36th Annual Hawaii International Conference on System Sciences, S. 292.1, Washington DC, USA, 2003.

[Hua99] Huang A. C., Ling B. C., Ponnekanti S., Fox A.: *Pervasive computing: What is it good for?*, In: Proceedings of the ACM International Workshop on Data Engineering for Wireless and Mobile Access, S. 84-91, Seattle, 1999.

[Hull97] Hull R., Neaves P., Bedford-Roberts J.: *Towards situated computing*, In: Proceedings of the First International Symposium on Wearable Computers, Cambridge, Massachusetts, 1997.

[Indu03] Indulska J., Robinsona R., Rakotonirainy A., Henricksen K.: *Experiences in using CC/PP in context-aware systems*, In: Proceedings of the 4th International Conference on Mobile Data Management, 2003.

[ISO99] International Organization for Standardization: *ISO 13407: Human-centred design processes for interactive systems*, 1999.

[Kapp03] Kappel G., Pröll B., Retschitzegger W., Schwinger W.: *Customisation for Ubiquitous Web Applications - A Comparison of Approaches*, Int. Journal of Web Engineering and Technology, Vol. 1 (1), S. 79–111, 2003.

[Korp03] Korpipää P., Mäntyjärvi J., Kela J., Keränen H., Malm E. J.: *Managing Context Information in Mobile Devices*, In: Pervasive Computing, Vol. 2 (3), S. 42-51, 2003.

[Korp05] Korpipää P.: *Blackboard-based software framework and tool for mobile device context awareness*, Dissertation, VTT Publications, Vol. 579, VTT Electronics, Espoo, Finnland, 2005.

[Laer01] Van Laerhoven K., Schmidt A.: *How to Build Smart Appliances?*, IEEE Personal Communications, Vol. 8 (4), S. 66-71, 2001.

[Lieb00] Lieberman H., Selker T.: *Out of Context: Computer Systems That Adapt To, and Learn From, Context*, IBM Systems Journal, Vol. 39, 2000.

[McCa97] McCarthy J., Buvac S.: *Formalizing context (expanded notes)*, In: Working Papers of the AAAI Fall Symposium on Context in Knowledge Representation and Natural Language, American Association for Artificial Intelligence, S. 99–135, 1997.

[Oez97] Öztürk P., Aamodt A.: *Towards a Model of Context for Case-Based Diagnostic Problem Solving*, In: Proceedings of the interdisciplinary conference on modeling and using context, 1997.

[Pasc98] Pascoe J.: *Adding generic contextual capabilities to wearable computers*, In: Proceedings of the Second International Symposium on Wearable Computers, Pittsburgh, Pennsylvania, 1998.

[Rah01] Rahlff O., Rolfsen R. K., Herstad J.: *Using Personal Traces in Context Space: Towards Context Trace Technology*, Personal and Ubiquitous Computing, 2001.

[Rang03] Ranganathan A., Campbell R. H.: *A middleware for context-aware agents in ubiquitous computing environments*, In: Proceedings of the ACM/IFIP/USENIX International Conference on Middleware, Rio de Janeiro, Brasilien, 2003.

[Rot03] Rothermel K., Bauer M., Becker C.: *Digitale Weltmodelle - Grundlage kontextbezogener Systeme*, In: Friedemann Mattern: Total vernetzt - Szenarien einer informatisierten Welt, Springer-Verlag, 2003.

[Rot08] Rothermel K.: *Kontextbezogene Systeme – die Welt im Computer modelliert*, Digitale Visionen, Springer-Verlag, 2008.

[Ryan97] Ryan N., Pascoe J., Morse D. *Enhanced Reality Fieldwork: the Context-Aware Archaeological Assistant*, University of Kent, Canterbury, 1997.

[Ryan99] Ryan N.: *ConteXtML: Exchanging Contextual Information between a Mobile Client and the FieldNote Server*, University of Kent, 1999. `http://www.cs.kent.ac.uk/projects/mobicomp/fnc/ConteXtML.html` (Abruf: 08.02.2010)

[Salb99] Salber D., Dey A. K., Abowd G. D.: *The Context Toolkit: Aiding the Development of Context-Enabled Applications*, In: Proceedings of the ACM CHI, Pittsburgh, Pennsylvania, S. 434–441, 1999.

[Schi93] Schilit B., Theimer M., Welch B.: *Customizing mobile applications*, In: Proceedings of USENIX Mobile & Location-Independent Computing Symposium, S. 129-138, Cambridge, Massachusetts, 1993.

[Schi94] Schilit B., Adams N., Want R.: *Context-Aware Computing Applications* In: Proceedings of the 1st International Workshop on Mobile Computing Systems and Applications, 1994.

[Schmi99] Schmidt A., Aidoo K. A., Takaluoma A., Tuomela U., Van Laerhoven K., Van de Velde W.: *Advanced Interaction in Context*, First International Symposium, HUC '99, Springer Verlag, 1999.

[Sin06] Van Sinderen M. J., Van Halteren A. T., Wegdam M., Meeuwissen H. B., Eertink E. H.: *Supporting context-aware mobile applications: an infrastructure approach*, IEEE Communications Magazine, Vol. 44 (9), S. 96-104, 2006.

[Sing06] Singh A., Conway M.: *Survey of Context aware Frameworks - Analysis and Criticism*, Online Article, Information Technology Services, University of North Carolina, Chapel Hill, 2006. `http://its.unc.edu/teap/tap/core/caf_review.pdf` (Abruf 17.12.2009)

[Stra03] Strang T., Linnhoff-Popien C., Frank K.: *CoOL: A Context Ontology Language to enable Contextual Interoperability*, In: Proceedings of 4th IFIP International Conference on Distributed Applications and Interoperable Systems, 2003.

[Stra04] Strang T., Linnhoff-Popien C.: *A Context Modeling Survey*, In: Workshop Proceedings: First International Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004, Nottingham, England, 2004.

[The94] Theimer M., Schilit B.: *Disseminating Active Map Information to mobile Hosts*, In: IEEE Network, Vol. 8 (5), S. 22-32, 1994.

[Tur06] Turjalei M.: *Integration von Context-Awareness in eine Middleware für mobile Systeme*, Diplomarbeit, Universität Hamburg, 2006.

[W3C07] World Wide Web Consortium: *Composite Capabilities / Preferences Profiles (CC/PP)* `http://www.w3.org/Mobile/CCPP` (Abruf: 08.02.2010)

[Wang04] Wang X. H., Gu T., Pung H. K., Zhang D. Q.: *Ontology Based Context Modeling and Reasoning using OWL*, In: Proceedings of the 2004 Communication Networks and Distributed Systems Modeling and Simulation Conference, 2004.

[Web05] Merriam-Webster Online Dictionary: *Context.* `http://www.merriam-webster.com/dictionary/context` (Abruf: 10.01.2010)

[Wei91] Weiser M.: *The Computer for the 21st Century*, Scientific American, 1991. `http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html` (Abruf: 10.01.2010)

[Wei93] Weiser M.: *Some Computer Science Issues in Ubiquitous Computing*, Scientific American, 1993. `http://www.ubiq.com/hypertext/weiser/UbiCACM.html` (Abruf: 10.01.2010)

[Weiss07] Weissensteiner A.: *Ubiquitäre Web-Anwendungen - Modellierung und Implementierung von Kontextinformation*, Magisterarbeit, TU Wien, 2007.

Anhang

In diesem Anhang sind einige Informationen zur praktischen Umsetzung des Rahmenwerks zur Kontextrepräsentation zusammengestellt, die für eine intensivere Befassung mit der Thematik herangezogen werden können.

Anhang A stellt eine Übersicht aller Ressourcen und Projekte zusammen, die im Rahmen der Referenzimplementation erstellt, und im Internet veröffentlicht wurden.

Anhang B führt alle im Verlauf der Referenzimplementation modellierten Kontexte und deren Regeln auf, die in der Beispiel-Ontologie enthalten sind.

A. Internet-Ressourcen

Nachfolgend werden die im Internet veröffentlichten Ressourcen aller Projekte, die im Laufe der praktischen Umsetzung angelegt wurden, aufgeführt.

A.1. Bibliothek für iPhone-Applikationen

- GitHub-Projektseite:
  `http://github.com/flxmllr/mobile-context-iphone-lib`
  - Binärdateien der statischen Bibliothek
  - Quellcode der statischen Bibliothek
  - Kurzbeschreibung
  - Anleitung zur Einbindung in iPhone-Projekte
  - API-Dokumentation

A.2. Demo-Applikation

- GitHub-Projektseite:
  `http://github.com/flxmllr/mobile-context-iphone-demo`
  - Quellcode der Demo-Applikation
  - Kurzbeschreibung
  - Anleitung zur Inbetriebnahme
- YouTube-Demonstrationsvideos:
  `http://www.youtube.com/flxmllr`
  - Modellierung von Kontexten
  - Live-Demonstration der Demo-Applikation

A.3. Kontextserver Web-Applikation

- GitHub-Projektseite:
  `http://github.com/flxmllr/mobile-context-server`
  - Quellcode der Web-Applikation
  - Quellcode der Kontext-Ontologie
  - Kurzbeschreibung
  - Anleitung zur Inbetriebnahme
  - API-Dokumentation
- URI der Kontext-Ontologie:
  `http://contextserver.felixmueller.name/context#`

B. Kontexte der Beispiel-Ontologie

Die nachfolgende Tabelle A.1 führt alle im Rahmen der Referenzimplementation in der Beispiel-Ontologie modellierten Kontexte und deren Regeln auf.

Tabelle A.1.: Kontexte der Beispiel-Ontologie

Kontext	**Regeln**
`#AfternoonFelixMueller` Label: `"Afternoon"` Type: `#TimeContext`	`(#belongsToUser:#FelixMueller)` ∧ `(#hasDateMin:"14:00:00"^^time)` ∧ `(#hasDateMax:"17:00:00"^^time)`
`#AmericanInternetFelixMueller` Label: `"AmericanInternet"` Type: `#NetworkContext`	`(#belongsToUser:#FelixMueller)` ∧ `(#hasCountryCode:"US")`
`#BritishInternetFelixMueller` Label: `"BritishInternet"` Type: `#NetworkContext`	`(#belongsToUser:#FelixMueller)` ∧ `(#hasCountryCode:"GB")`
`#ChargingFelixMueller` Label: `"Charging"` Type: `#DerivedContext`	`(#belongsToUser:#FelixMueller)` ∧ `(#hasSubContext:#InDockFelixMueller)` ∧ `(#hasSubContext:#AfternoonFelixMueller)`
`#DockedPositionFelixMueller` Label: `"DockedPosition"` Type: `#OrientationContext`	`(#belongsToUser:#FelixMueller)` ∧ `(#hasOrientationXMin:-0.06)` ∧ `(#hasOrientationXMax:0)` ∧ `(#hasOrientationYMin:-1.035)` ∧ `(#hasOrientationYMax:-1.01)` ∧ `(#hasOrientationZMin:-0.26)` ∧ `(#hasOrientationZMax:-0.18)`
`#EveningFelixMueller` Label: `"Evening"` Type: `#TimeContext`	`(#belongsToUser:#FelixMueller)` ∧ `(#hasDateMin:"17:00:00"^^time)` ∧ `(#hasDateMax:"23:00:00"^^time)`

#GermanInternetFelixMueller Label: "GermanInternet" Type: #NetworkContext	(#belongsToUser:#FelixMueller) ∧ (#hasCountryCode:"DE")
#GermanTimeFelixMueller Label: "GermanTime" Type: #TimeContext	(#belongsToUser:#FelixMueller) ∧ (#hasTimezone:"Europe/Berlin")
#HomeLocationFelixMueller Label: "HomeLocation" Type: #LocationContext	(#belongsToUser:#FelixMueller) ∧ (#hasLatitudeMin:50.91) ∧ (#hasLatitudeMax:50.93) ∧ (#hasLongitudeMin:6.94) ∧ (#hasLongitudeMax:6.96) ∧ (#hasWlanName:"Felix Netzwerk 2")
#InDockFelixMueller Label: "InDock" Type: #DerivedContext	(#belongsToUser:#FelixMueller) ∧ (#hasSubContext:#DockedPosition FelixMueller) ∧ (#hasSubContext:#HomeLocationFelixMueller)
#LandscapeLeftFelixMueller Label: "LandscapeLeft" Type: #OrientationContext	(#belongsToUser:#FelixMueller) ∧ (#hasOrientationXMin:-1.1) ∧ (#hasOrientationXMax:-0.99) ∧ (#hasOrientationZMin:-0.3) ∧ (#hasOrientationZMax:0.3)
#LoudFelixMueller Label: "Loud" Type: #AudioContext	(#belongsToUser:#FelixMueller) ∧ (#hasAverageLevelMin:0.1)
#ModerateFelixMueller Label: "Moderate" Type: #AudioContext	(#belongsToUser:#FelixMueller) ∧ (#hasAverageLevelMin:0.009) ∧ (#hasAveraveLevelMax:0.1)
#MorningFelixMueller Label: "Morning" Type: #TimeContext	(#belongsToUser:#FelixMueller) ∧ (#hasDateMin:"00:00:00"∧∧time) ∧ (#hasDateMax:"11:00:00"∧∧time)
#MorningStephanPavlovic Label: "Morning" Type: #TimeContext	(#belongsToUser:#StephanPavlovic) ∧ (#hasDateMin:"07:00:00"∧∧time) ∧ (#hasDateMax:"11:00:00"∧∧time)
#NightFelixMueller Label: "Night" Type: #TimeContext	(#belongsToUser:#FelixMueller) ∧ (#hasDateMin:"23:00:00"∧∧time) ∧ (#hasDateMax:"23:59:00"∧∧time)
#NoonFelixMueller Label: "Noon" Type: #TimeContext	(#belongsToUser:#FelixMueller) ∧ (#hasDateMin:"11:00:00"∧∧time) ∧ (#hasDateMax:"14:00:00"∧∧time)
#OfficeLocationFelixMueller Label: "OfficeLocation" Type: #LocationContext	(#belongsToUser:#FelixMueller) ∧ (#hasLatitudeMin:50.94) ∧ (#hasLatitudeMax:50.95) ∧ (#hasLongitudeMin:6.88) ∧ (#hasLongitudeMax:6.9)

#OfficeTimeFelixMueller Label: "OfficeTime" Type: #TimeContext	(#belongsToUser:#FelixMueller) ∧ (#hasDateMin:"08:30:00"∧∧time) ∧ (#hasDateMax:"17:00:00"∧∧time)
#OnSiteFelixMueller Label: "OnSite" Type: #LocationContext	(#belongsToUser:#FelixMueller) ∧ (#hasSpeedMax:0)
#PlainFelixMueller Label: "Plain" Type: #OrientationContext	(#belongsToUser:#FelixMueller) ∧ (#hasOrientationXMin:-0.1) ∧ (#hasOrientationXMax:0.04) ∧ (#hasOrientationYMin:-0.1) ∧ (#hasOrientationYMax:0.01)
#PortraitFelixMueller Label: "Portrait" Type: #OrientationContext	(#belongsToUser:#FelixMueller) ∧ (#hasOrientationXMin:-0.2) ∧ (#hasOrientationXMax:0.2) ∧ (#hasOrientationZMin:-0.3) ∧ (#hasOrientationZMax:0.3)
#RelaxingFelixMueller Label: "Relaxing" Type: #DerivedContext	(#belongsToUser:#FelixMueller) ∧ (#hasSubContext:#HomeLocation FelixMueller) ∧ (#hasSubContext:#OnSiteFelixMueller) ∧ (#hasSubContext:#WeekendFelixMueller)
#SilenceFelixMueller Label: "Silence" Type: #AudioContext	(#belongsToUser:#FelixMueller) ∧ (#hasAverageLevelMax:0.009)
#TravelingFelixMueller Label: "Traveling" Type: #LocationContext	(#belongsToUser:#FelixMueller) ∧ (#hasSpeedMin:0)
#WeekendFelixMueller Label: "Weekend" Type: #TimeContext	(#belongsToUser:#FelixMueller) ∧ (#hasWeekday:"7"∧∧integer) ∧ (#hasWeekday:"1"∧∧integer)
#WorkingDayFelixMueller Label: "WorkingDay" Type: #TimeContext	(#belongsToUser:#FelixMueller) ∧ (#hasWeekdayMin:"1"∧∧integer) ∧ (#hasWeekdayMax:"7"∧∧integer)
#WorkingFelixMueller Label: "Working" Type: #DerivedContext	(#belongsToUser:#FelixMueller) ∧ (#hasSubContext:#OfficeLocation FelixMueller) ∧ (#hasSubContext:#OnSiteFelixMueller) ∧ (#hasSubContext:#WorkingDay FelixMueller) ∧ (#hasSubContext:#OfficeTimeFelixMueller)

Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Köln, den 25. Februar 2010

Felix Müller

Erstellt auf einem Apple Mac.

Textsatz mit LaTeX 2$_\varepsilon$.