Universität Rostock

Traditio et Innovatio

Intelligente Integration und Dissemination von Diensten in Smart Environments

Crowdsourcing von Diensten für die pervasive Hochschule

Der Fakultät für Informatik und Elektrotechnik
der Universität Rostock zur Erlangung des
akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

vorgelegte Dissertation von
Philipp Lehsten
geboren am 22.06.1985 in Güstrow

Datum der Einreichung: 16.02.2018
Datum der Verteidigung: 06.11.2018

1. Gutachter: Herr Prof. Dr. Gero Mühl, Universität Rostock
2. Gutachter: Herr Prof. Dr. Djamshid Tavangarian, Universität Rostock
3. Gutachter: Herr Prof. Dr. Bernd Krämer, Fernuniversität Hagen

https://doi.org/10.18453/rosdok_id00002379

Dissertation
Fakultät für Informatik und Elektrotechnik

Kurzfassung

Mit der starken Verbreitung mobiler, internetfähiger Geräte kann heute an beinahe jedem Ort auf unterstützende Dienste und Informationen zugegriffen werden. Unterstützen diese Dienste den Nutzer in seiner Umgebung und beziehen sie nutzerspezifische Informationen bei der Assistenz mit ein, so kann man hier von einer intelligenten Umgebung sprechen. In räumlich begrenzten Bereichen, wie Universitäten oder Großunternehmen, lassen sich so großflächige, intelligente Umgebungen bilden.

Die generelle Erreichbarkeit der Dienste in den Umgebungen ist durch drahtlose Netzwerke und mobile Endgeräte gegeben. Eine effiziente Nutzung der Dienste setzt jedoch voraus, dass dem Nutzer passend zu seiner Situation jene Dienste angeboten werden, die einen Mehrwert bieten können. Gleichzeitig sollten sich diese Dienste in seine Situation so integrieren, dass sie eine unaufdringliche Assistenz ermöglichen.

Diese Dissertation analysiert die Herausforderungen, die sich bei der Erfassung und Nutzung dieser Dienste in großflächigen, intelligenten Umgebungen im Allgemeinen und an der pervasiven Hochschule im Speziellen ergeben. Dazu werden potentielle Dienste sowie deren Nutzer und Anbieter an Hochschulen klassifiziert. Anschließend wird mit einem Crowdsourcing-Ansatz eine Methode vorgestellt, die sich sowohl für die Identifikation und Klassifikation der Dienste eignet, als auch für die Unterstützung der Dienstintegration in unterschiedliche Anwendungen.

Die Ansätze zur Adressierung der identifizierten Herausforderungen werden in dem CASA-System (Context-Aware Service Access) gebündelt, das den Kern dieser Dissertation darstellt. Dabei handelt es sich um ein neuartiges, verteiltes System, das eine domänenübergreifende Dissemination und Integration von Diensten unterstützt. Besonderer Wert wurde auf Methoden zum Schutz der Privatsphäre der Nutzer gelegt und auf die Eignung für großflächige, intelligente Umgebungen. Neben mehreren prototypischen Implementierungen zur Evaluation einzelner Aspekte des CASA-Ansatzes erfolgte auch eine umfangreiche Umsetzung zur Nutzung im produktiven Lernmanagementsystem Stud.IP der Universität Rostock.

Abstract

The widespread availability of mobile internet-enabled devices allows accessing assisting services and information nearly everywhere. If these services support the user in his/her environment and incorporate user-specific information into the assistance the environment can be considered as an ïntelligent environment". In spatial restricted environments, like universities or big enterprises, such intelligent environments can be formed at a large-scale.

The general accessibility of these services is recently provided by wireless networks and mobile devices. An efficient use of these services however, requires to offer services to the user which have an added value regarding to his/her current situation. At the same time, such services should integrate itself into the situation by providing an unobtrusive assistance.

This thesis analyses the challenges in large-scale intelligent environments in general using universities as a suitable case study given the frequent usage of mobile services. To reach this aim, potential services and their consumers and providers are classified and a crowdsourcing method is presented. This method is suitable for the identification and classification of services, as well as for the support of the integration of services into different applications.

A focal point of this thesis is a CASA (Context Aware Service Access) system which bundles several approaches to solve these challenges. The CASA is a novel distributed system, which supports the dissemination and integration of services across domains. Special emphasis is placed on methods to protect privacy and on the suitability for large-scale smart environments. Besides prototypic implementations for the evaluation of single aspects, an implementation was conducted for the productive learning management system Stud.IP system at the University of Rostock.

Danksagung

Die dieser Arbeit zu Grunde liegende Forschung fand im Rahmen eines Stipendiums des Graduiertenkollegs Multimodal Smart Appliance Ensambles for Mobile Applications (MuSAMA) am Lehrstuhl für Rechnerarchitektur an der Universität Rostock statt. Die Förderung des Graduiertenkollegs durch die Deutsche Forschungsgemeinschaft (DFG), die sowohl fachlich, organisatorisch als auch finanziell erfolgte, ermöglichte ein fokussiertes Arbeiten in einem innovativen und inspirierenden Umfeld.

Der größte Dank gebührt meinem Mentor und Betreuer Prof. Djamshid Tavangarian für seine ununterbrochene Unterstützung und Förderung während des Studiums, Stipendiums und auch in der Zeit zwischen dem Ende der Finanzierung und dem Abschluss dieser Arbeit. Ich erlebte das Arbeiten in flachen Hierarchien, mit offenen, bereichernden Diskussionen und lernte neben dem wissenschaftlichen Arbeiten auch viel über die Leitung von Gruppen und das Motivieren in einem familiären Umfeld. Werte, die ich heute in meiner beruflichen Tätigkeit als Projektleiter sehr zu schätzen weiß. Weiterhin gilt mein besonderer Dank Prof. Dirk Timmermann und Prof. Bernd Krämer für die Diskussionen und ihr Interesse an meiner Arbeit.

Auch wenn der Lehrstuhl für Rechnerarchitektur an der Universität Rostock heute nicht mehr existiert, so war es doch dieser Ort mit Freunden und Kollegen, an dem die vorliegende Arbeit erst möglich wurde. Besonderer Dank gilt dabei Prof. Ulrike Lucke, die mich sowohl als Student als auch als Stipendiat stets gefördert und unterstützt hat. Dr. Daniel Versick, der über die Jahre stets ein unverzichtbarer Bestandteil des Lehrstuhls war, bin ich besonders für die vielen kleinen Diskussionen und Gespräche dankbar. Aber auch allen anderen Mitgliedern des Lehrstuhls, vom Sekretariat über die Technik bis zu den Studenten möchte ich meinen Dank aussprechen für die Unterstützung und gemeinsame Zeit in der familiären Atmosphäre.

Mein Dank gilt auch Dr. Sebastian Bader, dessen unkonventioneller Blick auf die Probleme dieser Arbeit mich stets weiter gebracht hat und durch den meine Forschung und diese Arbeit weiter an Qualität gewonnen haben.

Nicht weniger gilt mein Dank Dr. Raphael Zender und Dr. Enrico Dressler, die mich schon als Student in das Graduiertenkolleg eingebunden und mich für die Themen begeistert haben. Ebenso gilt mein Dank und Respekt den Leitern und Professoren des Graduierenkollegs, die für uns Stipendiaten ein Umfeld geschaffen haben, in dem wir forschen und uns entwickeln konnten. Die dabei erlebte Freiheit und Kreativität war maßgeblich am Erfolg meiner und aller anderen entstandenen Arbeiten beteiligt. Ebenso danke ich allen Mitgliedern des Graduiertenkollegs, allen voran Alexander Gladisch, Till Wollenberg, David Gassmann und allen anderen, die diese Zeit geprägt und bereichert haben.

Mein abschließender und tiefempfundener Dank gilt meiner Familie, die nie aufgehört hat an mich zu glauben und die mir nach dem Stipendium stets mit Ermahnungen, Ermutigung und Motivation den Abschluss der Dissertation zwischen beruflicher Arbeit und eigener Familie ermöglicht hat.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation: Wie sich unsere IT-Landschaft zur pervasiven Umgebung wandelt

Im Folgenden wird die vorliegende Arbeit motiviert und erläutert, in welchem Bereich sie angesiedelt ist. Dazu wird neben der fachlichen Einordnung auch an einem Fallbeispiel dargestellt, wie eine weitreichende Umsetzung der vorgestellten Ansätze aussehen könnte. Des Weiteren wird das Graduiertenkolleg MuSAMA vorgestellt, in dessen Rahmen diese Arbeit maßgeblich entstand. Nach einer groben Skizzierung der Herausforderungen im Forschungsumfeld der Arbeit werden die Zielstellung und der Aufbau der Arbeit dargestellt.

1.1.1 Heterogenität als Hürde zur spontanen Interaktion zwischen Geräten

> „Computing is not about computers any more. It is about living."
>
> (Nicolas Negroponte [Negroponte 96])

Die fortschreitende Entwicklung in der Informationstechnologie und die Verbreitung drahtloser Netze hat die Art und Weise, wie Computer wahrgenommen und genutzt werden, verändert. Die Sensoren und Prozessoren in unserer Umgebung sind in den Hintergrund getreten und wir nehmen in erster Linie die Funktionalitäten wahr, die durch sie ermöglicht werden. Diese Vielfalt der Geräte, die auf die Situation des Nutzers reagieren, reicht dabei von einem Smartphone über ein Auto bis hin zum Wecker auf dem Nachttisch. Marc Weiser beschrieb 1991 eine Zukunft, in der Computer in den Hintergrund getreten und damit ubiquitär geworden sind [Weiser 91]. Er prägte mit seiner Vision den Begriff des „Ubiquitous Computing", das heute auch als Forschungsbereich in der Informatik etabliert ist.

Seit der Vision von Marc Weiser beinhaltet die Idee einer intelligenten Umgebung einen Bereich, in dem der Nutzer durch Computer unterstützt wird ohne diese selbst wahrzunehmen. Aktuellere Definitionen betrachten das differenzierter und erkennen darin eine Umgebung,

die fähig ist Wissen über die Nutzer und deren Umwelt zu erfassen und so anzuwenden, dass dies die Art verbessert, wie der Nutzer die Umgebung erfährt [Cook 07].

Während bei der letztgenannten Definition der Fokus auf der Optimierung der Erfahrung der Umgebung liegt, ist aus Sicht dieser Arbeit die Unterstützung des Nutzers bei seinen Aufgaben das Ziel der intelligenten Umgebungen. Wie der Nutzer diese lösen möchte, obliegt dabei ihm. Eine intelligente Umgebung kann einen Nutzer sowohl reaktiv unterstützen, indem es ihm auf Nachfrage Informationen und Funktionalitäten anbietet, die dem Nutzer in seiner Situation helfen können, als auch proaktiv agieren, indem es ohne Aufforderung und nur auf Grundlage der Informationen über die Umgebung aktiv wird und Aktionen durchführt. Für diese Arbeit wird eine intelligente Umgebung wie folgt definiert:

Definition 1 (Intelligente Umgebung)

> Eine intelligente Umgebung ist ein räumlicher Bereich, in dem einem Nutzer durch ein digitales System eine Unterstützung zuteil wird, die sowohl reaktiv als auch proaktiv sein kann und bestimmt wird durch die Informationen, die das System über Nutzer und Umgebung gesammelt hat.

Um dies zu erreichen, müssen sowohl Anforderungen an die Technik als auch an ihre Nutzung erfüllt werden. Zum einen sollten die Geräte sich unkompliziert mit den in der Umgebung zur Verfügung stehenden Funktionalitäten vernetzen, ohne auf eine Konfiguration durch den Nutzer angewiesen zu sein. Zum anderen müssen die Geräte, die der Nutzer noch bei sich trägt und die als Schnittstelle zu den Funktionalitäten der Umgebung dienen, einfach und intuitiv zu bedienen sein.

Ein kleines Beispiel für die Minimalkonfiguration einer intelligenten Umgebung sind die heute verfügbaren Sprachassistenten wie Siri[1] von Apple oder Alexa[2] von Amazon. Sie unterstützen durch ihre Spracherkennung die Abfrage von Informationen in einem räumlichen definierten Bereich. Neben der Allgegenwärtigkeit der Computer beschrieb Weiser auch einige Situationen, in denen die Computersysteme sich proaktiv miteinander austauschen, um den Nutzer bei seinen Intentionen zu unterstützen. Während die Durchdringung des Alltags durch Computer bereits zu einem gewissen Grad erreicht ist, ist der Bereich der proaktiven Assistenz durch heterogene Systeme noch ein sehr aktives Forschungsfeld. Unter Proaktivität wird hier verstanden, dass die Funktionen nicht durch den Nutzer selbst angefordert werden, sondern dass das Assistenzsystem den Bedarf an der Funktion selbstständig ermittelt und sie dem Nutzer ohne sein Zutun zur Verfügung stellt. Ein einfaches, heute bereits realisierbares Szenario wäre eine aktivitätsabhängige Lichtsteuerung in einem Raum. Diese würde das Licht löschen, sobald ein Bewegungssensor keine Aktivität mehr registriert und so durch ein Steuerungssystem geschlussfolgert wird, dass der Nutzer den Raum verlassen hat. Jedoch sind diese Umsetzungen statische Verknüpfungen der Sensoren mit den Aktoren und kein „Erkennen" der Abwesenheit eines Nutzers, der Licht benötigt.

Um dieses „Erkennen" zumindest teilweise zu realisieren, wäre es notwendig, eine Vielzahl von Sensoren miteinander zu verknüpfen, denn gerade diese statischen Verknüpfungen sind sehr

[1] `https://www.apple.com/de/ios/siri/` [Online letzter Zugriff 10.02.2018]

[2] `https://developer.amazon.com/de/alexa` [Online letzter Zugriff 10.02.2018]

fehleranfällig. Wenn der Bewegungsmelder keine Aktivität mehr verzeichnet, kann das auch bedeuten, dass der Nutzer schläft oder ein Buch liest. Nur wenn man eine Vielzahl an Sensoren, z.B. in diesem Fall Druckmatten unter dem Teppich, Positionssensoren im Mobiltelefon, Türsensoren usw., durch smarte Regeln miteinander verknüpft, kann man zu Systemen kommen, die eine situationsbedingte Entscheidung optimal ermöglichen. Jedoch wäre bei einer Nutzung von derart vielen Sensoren eine statische Verknüpfung ungeeignet, da sie bei Abwesenheit eines Sensors nicht mehr funktionieren würde. Es ist also notwendig, eine dynamische Erkennung und Verknüpfung zu realisieren und diese bedürfen dann einer spontanen Vernetzung untereinander. Die gilt in besonderem Maße, wenn sich die Nutzer in ihrer Umgebung bewegen.

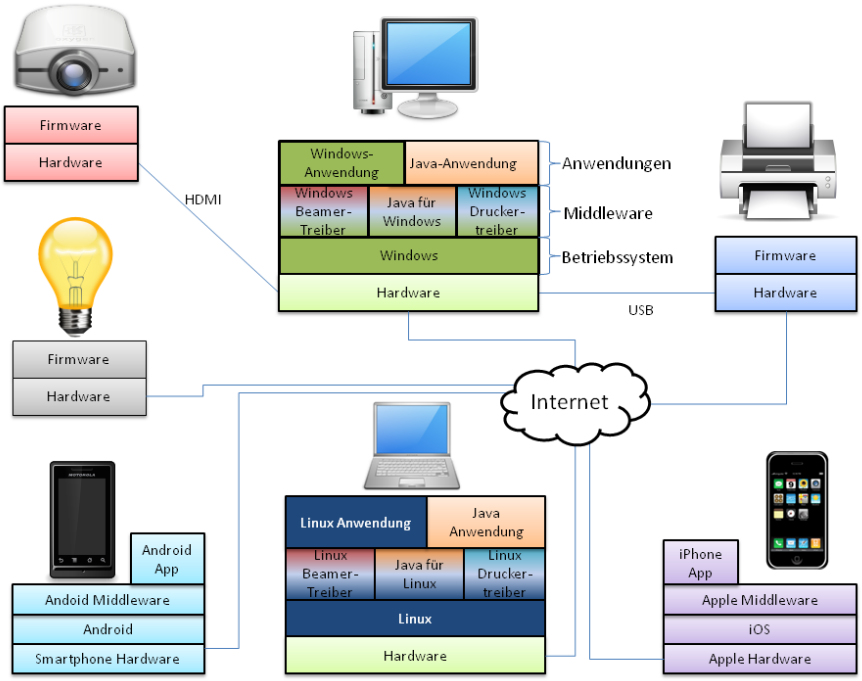

Abbildung 1.1: Beispiel für ein einfaches Netzwerk mit mehreren Geräten, die nur bedingt miteinander interagieren können

Diese spontane Vernetzung und die Interaktion zwischen den einzelnen Komponenten erfordert Kenntnisse über die umgebenden Systeme, Wissen über die genutzten Netzwerke sowie die systemintern verwendeten Sprachen und Protokolle. Dem entgegen stehen die Heterogenität und Inkompatibilität der Komponenten untereinander, da sie gegebenenfalls von verschiedenen Herstellern entwickelt wurden und so keine übergreifende Technologie existiert, die alle ohne Protokollumsetzung und Emulation miteinander verbinden würde. Abbildung 1.1 zeigt beispielhaft ein Ensemble aus verschiedenen miteinander vernetzten Geräten. Dabei wird zwischen der Hardware, den darauf aufsetzenden Betriebssystemen, den Middleware-Komponenten und den Anwendungen unterschieden. Die Middleware stellt dabei eine Vermittlungsschicht dar, über die Schnittstellen zur Software-Kommunikation bereitgestellt werden. So kann über die Java-Middleware die gleiche Java-Anwendung sowohl auf Windows, als auch auf Linux ausgeführt werden. Die Java-Middleware auf die aufgesetzt wird, ist je-

doch betriebssystemspezifisch. Für den dargestellten Windows-PC ist es dank der auf das Betriebssystem abgestimmten Middleware möglich den Projektor und den Drucker zu nutzen. Auch der Linux-Laptop kann durch den Treiber und die Verbindung mit dem Drucker dessen Funktionen nutzen. Jedoch wäre es dem Linux-Laptop nicht möglcih den Projektor zu nutzen, obwohl über das Internet und den Windows-PC ein Verbindung bestehen würde. Die Telefone können ohne die passende Middelware nicht auf den Drucker oder die Lampe zugreifen obwohl die Verbindung direkt möglich wäre. Und selbst wenn ein Ensemble-Mitglied die passende Middleware und Kommunikationsverbindungen zu allen anderen Teilnehmern hätte, so wären doch die anderen Teilnehmer untereinander noch immer unverbundene Inselsysteme. Neben den Problemen bei der Kommunikation und der Middleware sind mit der Marktführerschaft von wenigen zueinander inkompatiblen Betriebssystemen weitere Barrieren errichtet worden. So ist die Installation von Applikationen teilweise nur über die Plattform des Herstellers (z.B. AppStore von Apple) möglich oder sie erfordert tiefe Eingriffe in das System selbst.

Parallel zu den Diensten, die sich direkt auf einem Gerät installieren lassen, gibt es auch die Möglichkeit Dienste zu nutzen, die sich als Webapplikationen auf Webseiten auf nahezu jedem Gerät darstellen lassen. Über diesen Weg ist es möglich, geräte- und plattformunabhängig mit Diensten zu interagieren. Jedoch wird durch die Vielzahl und die Dynamik der Dienste ein komplexes Problem für pervasive Umgebungen eingeführt, das im nächsten Abschnitt erläutert wird.

1.1.2 Pervasive Universitäten als Large-Scale-Pervasive-Environment

Um das Forschungsthema dieser Dissertation zu motivieren, wird im Folgenden ein Szenario beschrieben, das zeigt, wie Integration und Dissemination von Diensten im Hochschulumfeld aussehen könnten. Dazu wird ein sogenannter CASA-Knoten verwendet. Dieser steht für ein kontextsensitives System, wie es in dieser Arbeit konzipiert und umgesetzt wird. An dieser Stelle ist es ausreichend diesen Knoten als ein Programm zu sehen, das verschiedene Informationsquellen nutzt, um die Situation des Nutzers zu erfassen. Basierend auf den Voreinstellungen des Nutzers werden dann verschiedene Funktionen und Dienste angeboten.

Fallbeispiel

Albert ist Wissenschaftler an der Pervasive University. Typischerweise steht er um 6:30 Uhr auf, damit er um 8:00 Uhr im Institut sein kann. Da er dies regelmäßig so handhabt, hat er in seinem CASA-Knoten dem Wetterdienst in der Zeit von 6 bis 7 Uhr eine hohe Priorität eingeräumt. Diese gilt analog für den Abfahrtsplan von seiner Haltestelle in der Zeit von 7 bis 8 Uhr. Für die Zeit von 8 bis 11 Uhr liegt die hohe Priorität bei den Diensten, die ihm das Campus-Management-System anbietet. Das sind im Wesentlichen die Webseiten für seine aktuelle und die jeweils nächste Veranstaltung. Zwischen 11 und 13 Uhr geht er gewöhnlich in die Mensa und hat sich für diesen Zeitraum den Speiseplan als Dienst eingetragen. Am frühen Nachmittag befasst er sich meist eine Zeit lang mit bürokratischen Aufgaben wie Reisekostenanträgen oder Anmeldungen für studentische Arbeiten. Diese werden durch ein

Prozessmanagementsystem der Universität online verwaltet und können so direkt als Dienste in Form von Webseiten angeboten werden. Ab 14 Uhr kehrt Ruhe im Institut ein und er kann sich seiner Forschung widmen. Dazu verwendet er meist die Dienste der sozialen Netzwerke für Forscher, um stets auf dem aktuellen Stand über neue Entwicklungen, Konferenzen und Ausschreibungen zu bleiben. Des Weiteren nutzt er die Dienste seines Instituts, um zusammen mit anderen Forschern Artikel zu schreiben oder Zugriff auf aktuelle Publikationen zu haben. Um 17 Uhr beginnt der Zeitabschnitt, in der die Abfahrtspläne von nahen Haltestellen relevant werden. Für die Abendgestaltung nutzt er Dienste, die vom lokalen Theater und dem Kino angeboten werden und ihn über aktuelle Veranstaltungen informieren. Für die Nacht hat er keine zeitlich gesteuerten Dienste eingestellt.

Heute ist alles jedoch etwas anders als sonst. Sein Institutsleiter hat sich gestern Abend entschieden, die Morgenbesprechung ausfallen zu lassen und den Termin im Gruppenkalender der Arbeitsgruppe gelöscht. Albert hat festgelegt, dass der CASA-Gruppen-Knoten seinen privaten CASA-Knoten auf seinem Smartphone informieren soll, wenn der Kalender sich ändert und dadurch vor 10 Uhr keine Termine im Kalender stehen. Dieses weckt ihn dann mit einem anderen Ton, so dass Albert schauen kann, was sich geändert hat.

Und heute ist so ein Tag. Da er nun um 8:00 Uhr noch zu Hause ist, bleiben die Wetter und Haltestellendienste relevant. Nach dem Frühstück genügt ein Blick auf sein Smartphone, um die Abfahrtszeit zu erfahren.

Bei seiner Ankunft im Institut loggt sich sein Smartphone in das Uni-WLAN ein und registriert ihn. Anschließend startet sein Smartphone seinen Bürocomputer, da dies vorher so von ihm spezifiziert wurde. Nachdem er für mehrere Veranstaltungen die Übungen ausgearbeitet hat, geht Albert in die Mensa. Die Mensa der Pervasive University bietet einen eigenen Dienst an, der Gästen auf Basis verschiedener Bedürfnisse Menüs zusammenstellt. Da Albert Vegetarier ist, kann er so bereits auf dem Weg zur Mensa schauen, welche fleischfreien Gerichte heute angeboten werden.

Zurück in seinem Büro erfährt er, dass seine Vorlesung heute Nachmittag auf Grund eines Stromausfalls in einem anderen Institut stattfinden muss. Er ändert den Raum im Campus-Management-System und sendet direkt eine Benachrichtigung an alle eingetragenen Teilnehmer. Da sich nun sein nächster Termin auf einem anderen Campus befindet, informiert ihn sein Smartphone über die nächste Straßenbahnverbindung. Dort angekommen, bekommt er einen Plan angezeigt, auf dem sein Standort und der entsprechende Raum markiert sind.

Im Raum öffnet er lediglich seinen Laptop und kann direkt über die Seite des Campus-Management-Systems auf die dortigen Beamer zugreifen. Er wählt aus dem Dienst für diesen Raum eine für Vorlesungen und diese Tageszeit empfohlene Konfiguration. Da er auch Studenten hat, die nicht vor Ort sein können, hat er den Aufzeichnungsdienst des Raumes aktiviert und ermöglicht ihnen so, die Veranstaltung per Videoübertragung zu verfolgen.

Im Anschluss an die Veranstaltung bleibt Albert auf dem Gelände und sucht sich einen freien Platz in der geisteswissenschaftlichen Bibliothek. Neben den Diensten zu den sozialen Netzwerken und denen seines Instituts, die sein Knoten ihm nun normalerweise anbietet, erscheinen nun auch die Dienste, die mit der Bibliothek auf diesem Campus verknüpft sind. So ist es ihm hier ebenfalls möglich, schnell seine Dokumente auszudrucken und er kann sich auch über die fachfremden, jedoch ebenfalls interessanten Neuanschaffungen der Bibliothek

informieren. Ab 17 Uhr wird ihm wie üblich die nächstgelegene Haltestelle und auch die schnellste Verbindung nach Hause angezeigt.

Die an dieser Hochschule verfügbaren Dienste sind feingranular und gekapselt. Dies ermöglicht eine Integration in die verschiedenen Arbeitsabläufe und eine Interaktion mit eigenen Diensten, wie dem Weckdienst auf dem Smartphone. Die Evaluation von Situationen und Empfehlung von Diensten und Aktionen erfolgt stets durch die Geräte, die im Besitz des Nutzers sind. Damit bleiben schützenswerte Informationen über den Nutzer und seine Handlungen in seinem Einflusskreis. Aktions- und Dienstempfehlungen, die unabhängig von der Person des Nutzers sind, wie zum Beispiel die Menüempfehlung, werden öffentlich angeboten. Die Empfehlung des in diesem Beispiel vegetarischen Menüs erfolgt durch das private Gerät, das auf diese Vorliebe konfiguriert ist, sie jedoch nicht nach außen weitergibt.

Um diese Vision zu realisieren, sind neben dem erwähnten CASA-Knoten noch viele andere Komponenten notwendig. Nicht zuletzt ist auch die Akzeptanz in der Nutzergemeinschaft sowie eine stete Pflege und Weiterentwicklung der Systembestandteile durch Nutzer und Entwickler für diese Vision essentiell. Das in dieser Arbeit vorgestellte CASA-Konzept erhebt nicht den Anspruch alle erwähnten Probleme zu lösen, jedoch beinhaltet es eine Vielzahl an Ansätzen und Lösungsvorschlägen, auf die in Zukunft aufgebaut werden kann.

1.1.3 Multimodal Smart Appliance Ensembles for Mobile Applications (MuSAMA)

Diese Arbeit entstand im Rahmen des Graduiertenkollegs Multimodal Smart Appliance Ensembles for Mobile Applications (MuSAMA). MuSAMA liegt die These zugrunde, dass die ubiquitäre Intelligenz unserer zukünftigen Umwelt von dynamischen Ensembles gebildet wird – lokale Ansammlungen intelligenter Alltagsgegenstände, deren Zusammensetzung sich unvorhersehbar ändern kann. Die Mitglieder eines solchen Ensembles müssen in der Lage sein, spontan und ohne menschliche Anleitung sinnvoll miteinander zu kooperieren, um den Nutzer zielgerichtet zu unterstützen – zum Beispiel als Smart Home oder als Smart Office. Die Forschungsthemen werden dabei in vier Forschungsschwerpunkte unterteilt:

1. Ubiquitäre Kontexterfassung mit dem Ziel der physischen Fähigkeit zur selbstständigen Umgebungswahrnehmung, zum Beispiel mittels verteilter Indoor-Mikrolokalisierung mit RFID.
2. Multimodale Interaktion und Visualisierung für Fragestellungen der situationsadaptiven Interaktion mit der pervasiven Geräteinfrastruktur und der dynamischen Visualisierung in Multi-Display-Environments.
3. Intentionserkennung und Strategiesynthese zur Ableitung von Nutzerzielen und zur Entwicklung von Strategien zu deren proaktiver Umsetzung.
4. Datenhaltung, Ressourcen und Infrastrukturmanagement zur Entwicklung und Optimierung von Kommunikationsparadigmen für Ensembles, um die verfügbaren Ressourcen und Informationen situationsspezifisch nutzen zu können.

In diesem Forschungsschwerpunkt ist auch die vorliegende Arbeit angesiedelt. Dabei lag der Fokus auf dem Infrastrukturmanagement und der Entwicklung eines Systems zur situationsspezifischen Nutzung der vorhandenen Informationen und Ressourcen.

Das Evaluationsszenario „Pervasive Universität"[Tavangarian 09a], das im Abschnitt 1.1.2 bereits kurz illustriert wurde, umfasst dabei raumübergreifende und großflächige pervasive Umgebungen, in denen universitäre Abläufe aus den Bereichen Forschung, Lehre und Verwaltung stattfinden. Als Evaluationsumgebung stehen dem Graduiertenkolleg zwei hochtechnisierte Spezialräume zur Verfügung. Neben einem Speziallabor mit diversen Multi-Display-Umgebungen existiert auch ein Smart Room, der sich durch verschiedene Sensorik und Aktorik insbesondere für die Wissensexploration und Meeting-Szenarien eignet.

1.2 Herausforderungen für die Empfehlung von Diensten in großflächigen, pervasiven Umgebungen

Nach Cook ist das Ziel von „Pervasive Computing" die Erschaffung einer Umgebung, in der netzwerkfähige Geräte sich in die Umgebung einbetten und sowohl eine verlässliche Konnektivität als auch zusätzliche Dienste anbieten. Im Resultat soll sich die Lebensqualität verbessern, ohne dass die Kommunikation und Computer bewusst wahrgenommen werden [Cook 07].

Damit wird eine Stadt, in der per Mobilfunktechnologie überall eine Netzwerkverbindung möglich ist, ebenso zu einer potentiell pervasiven Umgebung wie ein Universitäts-Campus, auf dem neben dem stadtweiten Mobilfunknetzwerk auch lokale, drahtlose und drahtgebundene Netzwerke existieren. Zusätzlich zu der Konnektivität kommen auf dem Campus weitere spezielle Dienste hinzu, die sich exklusiv an die Nutzer der Universität richten. Diese Umgebungen werden im Folgenden als großflächige, pervasive Umgebungen bezeichnet, um sie von den bereits angesprochenen räumlich begrenzten, pervasiven Umgebungen in Speziallaboren abzugrenzen.

Bei der Betrachtung von Diensten außerhalb eng umrissener Installationen wird die Position sowohl eines Dienstnutzers als auch die Verknüpfung eines Dienstes mit einem Ort relevant. Für die Verknüpfung von Diensten mit geographischen Positionen gab es bereits in den 1990er Jahren die Technik der Location-based Services, zu deutsch etwa Dienste mit Positionsbezug. Jedoch stellt der Ort im Umfeld von heterogenen Umgebungen mit einer Vielzahl an Institutionen innerhalb eines Gebäudes wiederum nur ein begrenzt nutzbares Filterkriterium dar. So ist die Technik, um Nutzer auf den Raum genau in einem Gebäude zu orten, heute kaum verbreitet und andererseits ist die Position ein Datum, das nutzerbezogen und damit schützenswert ist.

Die Empfehlung von relevanten Diensten, also Diensten, die einen Mehrwert für die Lebensqualität in der aktuellen Situation des Nutzers bieten, ist ein Teilziel des Pervasive Computings. Die dazu notwendige Verknüpfung von Nutzersituationen mit Diensten und ihre Verbreitung werden in dieser Arbeit als Dissemination bezeichnet und stellen die erste Herausforderung dar, der sich diese Arbeit widmet.

Die Dissemination kann grundsätzlich auf verschiedene Arten erfolgen. Zum einen kann der Nutzer Vorwissen über die verfügbaren Dienste haben und daher seine individuelle Situation als eine wahrnehmen, in der er einen (bestimmten) Dienst nutzen möchte. In diesem Fall könnten dem Nutzer grundsätzlich alle an einem Ort verfügbaren Dienste angeboten werden und er würde auf Basis seines Vorwissens aus den angebotenen Diensten auswählen. Neben dem Wissen über deren Existenz wäre auch Wissen zu ihrer Konfiguration und Verwendung durch den Nutzer notwendig. Dieser Weg stellt einen hohen Anspruch an den Nutzer und ist eng begrenzt auf die ihm bekannten Umgebungen. Daher ist diese Methode nicht geeignet, um die Vision des Pervasive Computings im Bereich der Dienstempfehlung zu realisieren, auch wenn die Realisierung der Methode technisch einfach umzusetzen wäre.

Der Gegenentwurf zu dieser Methode erfordert, dass die Umgebung über das notwendige Vorwissen zum Nutzer und seiner aktuellen Situation verfügt. Das Ziel eines Systems mit diesem Wissen wäre es das Nutzerverhalten und seine Intentionen vorherzusagen, um zu bestimmen, welche Dienste und Aktionen einen Mehrwert bieten könnten. Die Probleme für diesen Ansatz liegen in verschiedenen Bereichen. So besteht einerseits ein großes Missbrauchspotential für ein derartiges System, da sehr große Mengen von nutzerspezifischem Wissen gespeichert und verarbeitet werden müssten. Andererseits ist es technisch eine große Herausforderung dieses Wissen so aufzubereiten, dass auf nahezu alle beliebigen Situationen und Dienste reagiert werden kann.

Die Disseminiationsmethode, die in dieser Arbeit vorgeschlagen wird, versucht dem Nutzer einiges an umgebungsbezogenem Vorwissen abzunehmen und dieses für ihn vorzuhalten. Gleichzeitig soll die Erhebung und Verarbeitung der zur Dienstempfehlung notwendigen Daten nur auf seinen eigenen Geräten erfolgen und nur in dem Umfang, den er für sich akzeptiert.

Ist ein Dienst oder eine Aktion als relevant identifiziert, muss sich die Ausführung in die Umgebung und speziell in den Arbeitsablauf des Nutzers integrieren lassen. Diese Integration stellt die zweite Herausforderung dar, der sich diese Arbeit widmet. Sie kann durch eine Textpräsentation erreicht werden, die sich auf dem Bildschirm des Nutzers öffnet und ihn auf einen Dienst oder eine Aktion hinweist. Dies kann sehr schnell störend auf den Nutzer wirken. Alternativ kann die Aktion oder der Dienstaufruf ohne die Information des Nutzers erfolgen, was jedoch vermutlich zu einer Irritation des Nutzers führen würde.

In dieser Arbeit wird daher eine Methode vorgeschlagen, die eine Integration in die Anwendungen des Nutzers erlaubt. So kann passend zur jeweiligen Arbeitssituation eine in die Anwendung integrierte Präsentation von möglichen Funktionen erfolgen. Der Nutzer nimmt dabei keine zusätzliche Komplexität wahr, da er sich in einem für ihn gewohnten Software-Umfeld bewegt.

1.3 Zielstellung dieser Arbeit

Im Fokus dieser Arbeit steht die Konzeption und prototypische Implementierung einer Infrastruktur für die nutzergetriebene Dienstvermittlung in pervasiven Umgebungen unter Einbeziehung der Aspekte Nutzung, Pflege und Erweiterbarkeit durch den Nutzer selbst.

Die Vermittlung orientiert sich dabei an der Kontextdefinition von Anid K. Dey [Dey 00]. Entsprechend dieser wird gezeigt werden, wie sich Dienste anhand eines Ortes, der Identität des Nutzers, seiner Aktivität oder der Zeit verteilen, finden und nutzen lassen. Diese Herausforderung wurde in zwei Teilziele unterteilt.

Zum einen musste untersucht werden, welche Anforderungen durch den nutzergetriebenen Ansatz gestellt werden und welche Bedingungen sich dadurch für die Architektur des Systems ergeben. Zum anderen musste das System zumindest prototypisch umgesetzt werden, um neben der Demonstration der Anwendbarkeit auch evaluieren zu können, ob dieser Weg der Nutzereinbindung durch den Anwender auch akzeptiert und genutzt werden würde.

Daher wurden im ersten Teil der Arbeit die Anforderungen an eine Infrastruktur herausgearbeitet, die system- und plattformunabhängig in der Lage ist, einen Nutzer darin zu unterstützen Dienste und Informationen zur Verfügung zu stellen. Daraus wurde ein Konzept, genannt CASA, entwickelt, das von der generellen Dienstnutzung abstrahiert und Funktionen und Informationen dort zur Verfügung stellt, wo sie nach Ansicht der Nutzer sinnvoll sind. CASA steht dabei für den kontextbewussten Dienstzugriff (Context-Aware Service Access) und bildet den Prozess von der Beschreibung eines relevanten Dienstes durch den Nutzer bis zur dynamischen Integration dieses Dienstes in den Arbeitsablauf anderer Nutzer ab. Neben den technischen Herausforderungen, die sich durch heterogene Diensttechnologien, Anwendungen und Endgeräte stellen, wurden sowohl organisatorische Herausforderungen, wie der Schutz der Nutzerdaten, als auch soziale Herausforderungen, wie die Motivation zur Wartung und Weiterentwicklung des Systems, betrachtet. Das entstandene Konzept ist dabei über das hier fokussierte gruppenöffentliche Campus-Umfeld hinaus einsetzbar und kann angewendet werden, um den Nutzer sowohl im privaten, wie auch im öffentlichen Raum bei der Dienstsuche und -nutzung zu unterstützen.

Im zweiten Teil dieser Arbeit wird eine prototypische Umsetzung des CASA-Konzeptes beschrieben. Dabei wird neben der Beschreibung der Umsetzung und der Demonstration der Machbarkeit auch die Architektur vorgestellt, mit der CASA im zentralen Lernmanagementsystemen der Universität Rostock sowie im dortigen Juniorstudium regulär eingesetzt wird. Diese Ergebnisse werden sowohl aus der technische Perspektive evaluiert als auch durch Nutzerbefragungen bewertet.

1.4 Aufbau der Arbeit

Die weitere Arbeit ist wie folgt gegliedert: In Kapitel 2 werden zuerst die Komplexität und die Herausforderung der Kontextorientierung bei der Dienstvermittlung dargestellt. Darauf folgend wird die Dienstorientierung als Paradigma für eine Universität der Zukunft motiviert und gezeigt, warum sich gerade hier eine Brücke zur Kontextorientierung schlagen lässt. Anschließend wird der Aspekt der Einbindung des Nutzers untersucht. Das hier genutzte Crowdsourcing wird erläutert und es wird gezeigt, welche weiteren Randbedingungen durch dieses Konzept beachtet werden müssen. Abschließend werden verwandte Arbeiten untersucht, um eine klare Abgrenzung zu existierenden Ansätzen darzustellen und den in dieser Arbeit geschaffenen Fortschritt einzuordnen.

Kapitel 3 behandelt das CASA-Konzept, das als Antwort auf die Anforderungen, die sich durch das Szenario, die Kontextorientierung und die Nutzereinbindung ergeben, entwickelt wurde. Dabei wird von der Systemarchitektur ausgehend zuerst gezeigt, warum ein Ansatz mit verteilten, autonomen Knoten präferiert wurde und einem Konzept mit einem zentralen Kontextserver vorgezogen wurde. Anschließend wird auf die verschiedenen Knotenarten sowie auf die Organisation und Kommunikation zwischen diesen eingegangen. Im Folgenden werden alle Knotenarten mit ihren Komponenten sowie die Modularisierung erläutert. Die letzten beiden Abschnitte befassen sich mit einer Auswahl von Konzepten zur Nutzereinbindung sowie einer Betrachtung der rechtlichen Implikationen, die sich durch das CASA-Konzept und dessen Umsetzung ergeben.

Die prototypische Umsetzung des Systems wird in Kapitel 4 behandelt. Dabei werden die Softwarearchitektur, die einzelnen Komponenten und das verwendete Kontextmodell erläutert. Mit der Beschreibung der Umsetzung und Anpassung für das Lernmangementsystem der Universität Rostock wird hier auch gezeigt, wie die Implementierung auf einen realen Anwendungsfall übertragen werden kann und wie die erstellten Software-Module eingesetzt und erweitert werden können. Abschließend wird auf weitere Realisierungen im Bezug auf die Motivation der Nutzer in gruppenöffentlichen Knoten und die Integration weiterer Nutzerschnittstellen bei öffentlichen Knoten eingegangen.

In Kapitel 5 wird auf die Evaluation von CASA genauer eingegangen. Dabei wird zuerst eine Evaluation ausgehend von der in Kapitel 2 erstellten Anforderungsanalyse durchgeführt. Anschließend wird am Beispiel der durchgeführten Realisierungen die Anwendbarkeit evaluiert.

Die Arbeit wird durch eine Zusammenfassung mit Ergebnisdiskussion sowie durch einen Ausblick auf die noch offenen sowie die aus der Arbeit resultierenden Forschungsfragen beendet.

Kapitel 2

Kontext-basierte Verteilung und Integration von Diensten

„Man is still the most extraordinary computer of all."

(John F. Kennedy [Kennedy 63])

Um einem Nutzer in seinem Alltag den Grad an Assistenz zu bieten, den er als erwünscht ansieht, ist es unerlässlich den Nutzer selbst zu beteiligen. Die Charakterisierung einer Situation kann zwar automatisiert durch einen Computer anhand von Messungen erfolgen, jedoch ist die Bewertung der Eignung von Diensten als Assistenz zu dieser Situation nutzerabhängig und subjektiv. Sowohl die Definition von relevanten Situationen als auch die Klassifikation der anwendbaren Assistenz bedürfen des „außergewöhlichsten aller Computer", des Menschen.

Ein Ziel dieser Arbeit ist es, dem Nutzer in Abhängigkeit von seiner aktuellen Situation die Dienste anzubieten, die einen Mehrwert bieten. Um diese Funktionalitäten in Abhängigkeit von der Situation und damit in Abhängigkeit der vorhandenen Kontextinformationen zu erfassen, zu verteilen und anzubieten, wird in diesem Kapitel zunächst erläutert, was in der vorliegenden Arbeit unter dem Begriff Kontext verstanden werden soll. Anschließend wird dargestellt, warum sich Hochschulen besonders gut für die Anwendung des Dienste-Paradigmas eignen und gleichzeitig Vorteile unter dem Aspekt der Kontextorientierung haben. Um das in der Einleitung bereits angeschnittene Problem der sich weiterentwickelnden und anpassenden Systeme zu adressieren, wird im Folgenden das Konzept des Crowdsourcing erläutert. Diese Methode integriert den Nutzer in einen Prozess der Erstellung von Inhalten und in die Weiterentwicklung des Systems als solches und verringert somit den redaktionellen Aufwand für die Betreiber des Systems. Gleichzeitig kann ein Teil der Weiterentwicklung durch die Nutzer betrieben oder zumindest der Bedarf für Weiterentwicklungen artikuliert werden. Durch Vergleiche mit der Evolutionstheorie wird dargestellt, welches Potential in der Anwendung für nutzergetriebene Systeme liegt. Mit den sich daraus ergebenden Bedingungen werden die Anforderungen für eine intelligente Umgebung mit den beschriebenen Eigenschaften formuliert und die existierenden Systemarchitekturen auf ihre Eignung überprüft. Abschließend wird eine Systematik verwandter Arbeiten erstellt und die hier entwickelte Idee eingeordnet.

Dabei wird erläutert, welche Lücken in der Entwicklung durch das in der Arbeit vorgestellte Konzept adressiert werden.

2.1 Kontext und Semantik als Herausforderung

Die Herausforderungen in der Forschungsrichtung, eine ubiquitäre Umgebung für die Unterstützung eines Nutzers bereitzustellen, bestehen auf mehreren Ebenen. Auf der unteren Ebene müssen Geräte entwickelt werden, die eine Vielzahl an Funktionalitäten möglichst kompakt bieten können. Die mittlere Ebene ist geprägt durch die Erfassung und Verarbeitung von Kontextinformationen und schließlich durch die Unterstützung des Nutzers bei seinen Aufgaben. In der oberen Ebene stehen der Nutzer selbst und die Frage, wie eine Interaktion mit diesen sichtbaren und unsichtbaren Geräten ablaufen kann, im Mittelpunkt. Da die mittlere Ebene im Fokus der vorliegenden Arbeit steht, soll sie mit zugehörigen Fragestellungen genauer betrachtet werden.

Im folgenden Abschnitt wird zuerst auf die unterschiedlichen Definitionen von Kontext eingegangen, um in den folgenden Unterabschnitten die Erfassung, Modellierung, Verwaltung und Nutzung von Kontextinformationen erläutern zu können.

2.1.1 Definitionen von Kontext

Um sich den Herausforderungen der Kontexterfassung und -verarbeitung zu widmen, ist es zuerst notwendig den Begriff Kontext genauer zu betrachten. Im Wesentlichen wird Kontext zur Beschreibung von Situationen genutzt, jedoch ist diese Beschreibung stark geprägt durch die Sichtweise des Beobachtenden. Die Situation einer Entität lässt sich auch als Zustand dieser verstehen, wobei der Kontext dann die verschiedenen Merkmale und (Mess-)Größen umfasst, die zu einer Beschreibung genutzt werden können.

So ist aus Sicht eines Dozenten eine Vorlesung dadurch zu beschreiben, dass er sich in einem Raum mit Zuhörern befindet und diesen Wissen vermittelt. Aus Sicht eines Studienbüros würde hier im Vordergrund stehen, dass diese Personen eingeschrieben sind und dass der Raum in diesen Zeiträumen nicht anders verwendet werden kann. Aus Sicht eines Administrators ist während einer Vorlesung mit erhöhter Netzwerkaktivität aus dem Vorlesungsraum zu rechnen, auf die hier reagiert werden muss. Entsprechend der verschiedenen Beobachter ergeben sich verschiedene Dimensionen und Metriken für die Klassifikation von Situationen, die für die Beobachter jeweils ein anderes Gewicht und damit eine andere Bedeutung haben. Zu beachten ist hier auch, dass gerade die Semantik eine große Rolle spielt. So kann die Beschreibung „Die Vorlesung ist gut“ für den Dozenten eine Aussage über das Interesse der Studenten sein, für das Studienbüro eine über die Auslastung des Raumes und für den Administrator eine über die Auslastung des Netzwerkes sein.

Für den Informatiker wird der Bezugsrahmen dabei durch den Nutzer und seine Interaktion mit einer Anwendung umrissen. Dies zeigt sich auch in der häufig in diesem Zusammenhang zitierten Definition von Abowd und Dey:

> Kontext ist jede Information, die verwendet werden kann um die Situation einer Entität zu charakterisieren. Eine Entität ist eine Person, ein Ort oder ein Objekt, das für die Interaktion zwischen einem Nutzer und einer Anwendung als relevant betrachtet wird, einschließlich dem Nutzer und der Anwendung selbst.
>
> (übersetzt nach Anind K. Dey und Gregory D. Abowd [Dey 00])

Während in dieser Definition die Entitäten Person, Ort und Objekt die Beschreibung eingrenzen, erlaubt die Definition von Chen und Kotz, die aus dem Bereich Mobile Computing kommt, eine abstraktere Betrachtung:

> Kontext ist die Menge an Umweltzuständen und -vorgaben, die entweder das Verhalten einer Anwendung bestimmen oder in der ein Anwendungsereignis auftritt und interessant für den Nutzer ist.
>
> (übersetzt nach Guanling Chen und David Kotz [Chen 00])

Diese Definition legt Wert darauf, dass Kontext einerseits für die Aktionen oder Reaktionen einer Anwendung bestimmend sein kann, ebenso wie er eine Beschreibung für Situationen geben kann, in denen die Aktionen der Anwendung als interessant für den Nutzer empfunden werden.

In dieser Arbeit ist die durch den Nutzer empfundene Relevanz eines Dienstes entscheidend für seine Motivation und Bereitschaft, diesen mit anderen zu teilen oder, vielmehr ihn zu nutzen. Daher wird für die Definition von Kontext folgende Formulierung genutzt:

Definition 2 (Kontext)

> Der Kontext umfasst jede darstellbare Information zur Beschreibung einer Situation, in der eine Anwendung oder das Ergebnis eines Aufrufs der Anwendung für einen Nutzer einen Mehrwert bringt.

Es wird damit bewusst auf Kontexteigenschaften verzichtet, die durch einen Nutzer wahrgenommen werden können, deren Darstellung jedoch für ein technisches System kaum möglich ist. Als Beispiel seien hier Gerüche oder Töne genannt, die zwar detektiert aber nur bedingt interpretiert werden können.

2.1.2 Erfassung von Kontextinformationen

Um Daten mit technischen Mitteln zu erfassen, werden Sensoren eingesetzt. Der Begriff Sensor wird in dieser Arbeit als ein System zur Erfassung von Daten verstanden. Abstrakt betrachtet können Sensoren in drei Klassen eingeteilt werden [Indulska 03]. Es wird dabei nach physischen, virtuellen und logischen Sensoren unterschieden. Ein physischer Sensor ist dabei ein real existierendes System beziehungsweise eine Komponente eines Systems, das zur quantitativen oder qualitativen Erfassung von Daten aus der Umwelt dient. Ein solcher physischer Sensor kann zum Beispiel ein Thermometer oder auch ein Bewegungsmelder sein. Neben den Umweltgrößen und Ereignissen der realen Welt können auch Messungen auf der Software-Seite eines Computers durchgeführt werden. Diese virtuellen Sensoren sind Softwareprozesse, die zum Beispiel die Speicherauslastung eines Systems messen oder Ereignisse

wie das Eintreffen einer E-Mail. Die letzte Gruppe von Sensoren sind die logischen Sensoren. Sie kombinieren die Werte sowohl von physischen als auch von virtuellen Sensoren mit Informationen aus anderen Quellen und können so komplexe Eigenschaften und Ereignisse beschreiben. Ein logischer Bewegungssensor kann zum Beispiel die Verbindungsdaten eines WLAN Access Points nutzen, um in Verbindung mit vorhandenen Ortsinformationen zu den Access Points die Bewegungen von Nutzern zu registrieren.

Um die Daten eines Sensors in einem kontextsensitiven System nutzen zu können, ist es notwendig, diese für das System zugänglich zu machen. Dazu werden Treiber für die physischen Sensoren und die Application Programming Interfaces (APIs) für virtuelle und logische Sensoren als Schnittstellen angesprochen. Diese Schnittstellen sind hersteller- und systemabhängig und daher für jeden Sensor spezifisch.

Das Lösen des daraus resultierenden Problems der Heterogenität und des damit verbundenen Aufwands beim Verknüpfen von Anwendungen und Sensoren ist bereits seit einigen Jahren Ziel wissenschaftlicher Arbeiten [Henricksen 06, Compton 09].

2.1.3 Möglichkeiten der Modellierung von Kontextinformationen

Um Kontextinformationen verarbeiten zu können, muss ein Modell existieren, auf das sie abgebildet werden können. Es existiert eine Vielzahl an Modellen, die sich in drei große Modellierungsmethoden gliedern lassen. Diese sind die Modellierung von Objekten und deren Rollen (ORM, Object Role Modeling), räumliche Modellierung (engl. Spacial Modeling) und die ontologiebasierte Modellierung [Bettini 10]. Neben der Methode ist auch eine Einteilung auf Basis der Datenstrukturen möglich, dabei wird unterschieden nach Key-Value-, Markup Scheme, graphischen, objektorientierten, logikbasierten und ontologiebasierten Modellen [Strang 04]. Im Folgenden werden verschiedene Ansätze diskutiert und ihre Eignung für die Modellierung von Kontext im Umfeld einer pervasiven Hochschule untersucht.

Frühe Ansätze mit Key-Value-Modellen und Markup Schemes

Die ersten Ansätze zur Modellierung von Kontextinformationen basierten auf Key-Value-Pairs, also Schlüssel-Wert-Paaren, in denen das Wissen einer Domäne abgebildet wurde. Sie eignen sich zur einfachen Verwaltung von wenig komplexen Daten und statisch bekannten Eigenschaften, jedoch haben sie Schwächen bei der Modellierung von komplexen Strukturen und der Abfrage von Wissen (engl. Retrieval). Eine Anwendung wurde in [Schilit 94] beschrieben, um den Kontext von pervasiven Anwendungen zu modellieren.

Darauf folgten Ansätze, die Auszeichnungssprachen (engl. markup schemes) nutzen. Sie verwenden eine hierarchische Datenstruktur mit Attributen und Parametern, die sich häufig rekursiv aus anderen ergaben. Meist basieren diese Ansätze auf der Standard Generic Markup Language (SGML), die die Oberklasse von Auszeichnungssprachen wie XML ist [Strang 04]. Durch ihre hierarchische Struktur eignen sich diese Ansätze gut zur Modellierung von komplexen Anwendungsfällen und der Abfrage von Wissen aus diesen, jedoch ist durch die textuell häufig sehr umfangreichen Beschreibungen ihre Nutzung auf ressourcenbeschränkten Geräten begrenzt.

Graphische Modelle und die Modellierung der Rollen von Objekten

Graphische Modelle verfügen über eine Diagrammnotation, die eine strukturierte Darstellung des Schemas und der Instanzen eines Kontextmodells erlaubt. Ein Vertreter dieser Gruppe ist Object-Role Modeling (ORM)[Henricksen 03]. Es gehört zu den faktenbasierten Kontextmodellierungsmethoden und versucht durch die Zuordnung von Entitäten zu Rollen und den Beziehungen zwischen Entitäten das Wissen innerhalb einer Domäne darzustellen.

Ihren Ursprung haben diese Methoden in der strukturierten Modellierung von Wissen in Datenbanken, z.B. für Abfragen und als Vorlage für die Software-Umsetzungen. Daher eignen sich diese Modelle besonders gut, um aus ihnen Datenbankschemata für spätere Softwarerealisierungen abzuleiten. Ein bekannter Vertreter ist hier das Entity-Relationship Model (ER Model) von Chen [Chen 76].

Zu den graphischen Modellen lassen sich auch Notationen zur Modellierung von Geschäftsprozessen, wie die Business Process Modelling Notation (BPMN) [OMG 11] zählen. Ebenso lassen sich Graphen-basierte Methoden wie Petri-Netze oder auch Zustandsgraphen zu dieser Kategorie zählen. Die Modellierung mit letzteren Methoden ist besonders praktikabel, wenn es diskrete Zustände mit definierten Übergängen gibt.

Der Vorteil dieser graphischen Modelle im Allgemeinen ist die leichtere Verständlichkeit für Nutzer, die kompetent in der Domäne, aber weniger kompetent im Verständnis von abstrakteren Kontextmodellen sind. Des Weiteren vereinfachen sie durch ihre Nähe zur Modellierung von Datenbanken eine Umsetzung in selbigen oder Objekt-orientierten Systemen. Ihre Schwäche ist die, bei komplexen Sachverhalten notwendige, Wahl zwischen einer hohen Abstraktion oder einer unübersichtlichen Komplexität, die sich schon durch die graphische Darstellung ergibt.

Räumliche Modellierung

Bei der räumlichen Modellierung wird der physische Ort als Bezugssystem genutzt. Diese Modelle sind ebenfalls meist faktenbasiert und unterscheiden zwischen zwei Koordinatensystemen, den geometrischen und den symbolischen Koordinatensystemen. Bei ersteren wird ein geodätisches Referenzsystem, wie zum Beispiel WGS84 bei GPS, genutzt, um Entitäten mit Orten zu verknüpfen. Im Gegensatz dazu können auch symbolische Koordinaten verwendet werden, um zum Beispiel eine Entität mit einem virtuellen Raum oder einer Bezeichnung zu verknüpfen. Dies kann die ID eines nahen Zugriffspunktes, eines angeschlossenen Netzwerkes oder jede andere Form von abstrakter Position sein.

Der Vorteil dieser Modelle ist, dass sie sich besonders gut eignen, um Anfragen zu begrenzen, da der Ort häufig eine zentrale Rolle für die Situation eines Nutzers einnimmt. Daher können zum Beispiel Dienste, die sich nicht in einem gewissen Umkreis befinden, eventuell schon im Vorfeld ausgeschlossen werden. Des Weiteren erlaubt der Ort durch die Entfernung zum Nutzer eine Relevanzbewertung, da z.B. entferntere Orte, die die gleichen Dienste anbieten meist weniger relevant wären. Der Nachteil liegt in der Notwendigkeit von Ortsinformationen für Entitäten, da diese teilweise dynamisch sind und der Aufwand zur Erhebung hoch ist.

So müssen die Entitäten einerseits ihren eigenen Ort kennen, was je nach Umgebung unterschiedliche Technologien erfordert (z.B. GPS, WLAN, Bluetooth). Andererseits müssen auch die Dienste mit Ortsinformationen verknüpft sein, was manuelle Eingaben erfordert und sich nur bedingt automatisieren lässt.

Logik- und ontologiebasierte Modellierung

Logikbasierte Systeme bilden den Kontext als eine Menge von Fakten, Regeln und Folgen ab. Dazu wird eine formale Sprache genutzt, mit der festgelegt wird, wie sich die Regeln auf die Fakten anwenden lassen und welche Schlüsse daraus zu ziehen sind. Vertreter dieser Art der Modellierung sind Regel-basierte Systeme. Dabei kann zwischen prozeduralen und deklarativen Systemen unterschieden werden. Bei ersteren werden Regeln definiert, die mögliche Schlussfolgerungen aus vorhandenen Fakten zulassen. Eine solche Regel könnte sein: WENN eine Person 18 Jahre oder älter ist, DANN gilt sie als erwachsen. Analog dazu wäre die Formulierung in deklarativem Wissen: Die Person ist 18 Jahre alt. Mit 18 Jahren gilt eine Person als erwachsen.

Ontologien gehen dabei einen Schritt weiter. Der Begriff bezeichnet die Wissenschaft vom Seienden als solches, welche sich zurückführen lässt bis zur Metaphysik des Aristoteles. Das Wesen des Ontologie-Begriffs heute, wie er in der Informatik verwendet wird, ist die Darstellung der Klassifizierung von Entitäten, der zwischen ihnen existierenden Beziehungen und der Konzepte, die sich mit ihnen abbilden lassen. Um Ontologien zu realisieren, wird häufig die Web-Ontology-Language (OWL) verwendet [McGuinness 04]. Es handelt sich dabei um eine Spezifikation des World-Wide-Web-Consortiums (W3C), mit der sich Ontologien formulieren lassen. Eine Umsetzung davon entstand zum Beispiel bei COBRA [Chen 04b]. Der folgende Quelltext ist ein Auszug aus der COBRA-Ont[1] und stellt die Eigenschaften eines speziellen Mobiltelefons dar.

```
<owl:Class rdf:ID="SonyEricssonT68i">
  <rdfs:label>SonyEricssonT68i</rdfs:label>

  <rdfs:subClassOf rdf:resource="#CellPhone"/>
  <rdfs:subClassOf rdf:resource="&dev;DeviceSupportsBluetooth"/>
  <rdfs:subClassOf rdf:resource="&dev;DeviceSupportsIrDA"/>
</owl:Class>
```

Quelltext 2.1: Ontologie-Eintrag für ein Sony Ericsson T68i Mobiltelefon aus der COBRA-Ont

Diese maschinenlesbaren Formalisierungen lassen sich nutzen, um automatisiert Wissen abzuleiten. Im Beispiel zeigt die Zuordnung des T68i zu den Klassen der Mobiltelefone und der Bluetoothgeräte an, dass es zusätzlich zu den Eigenschaften, die Mobiltelefone generell haben, auch die Eigenschaften eines Bluetoothgerätes hat.

Die Nutzung von Ontologien erfordert ein komplexes Wissen über die zu beschreibende Domäne. Allerdings erlaubt gerade diese Formalisierung auch eine Austauschbarkeit zwischen

[1] http://daml.umbc.edu/ontologies/cobra/0.4/personal-device [Online letzter Zugriff 15.02.2018]

verschiedenen Domänen, wenn die gleichen Strukturen benötigt werden. So könnte das Wissen über ein T68i, das einmal für eine Domäne definiert wurde, unter Einbeziehung aller Überklassen auch in einer anderen Domäne genutzt werden. Doch genau hier zeigt sich ein Problem der starken Formalisierung mittels Ontologien und RDF [Manola 04], denn sie können schnell sehr komplex werden. Ihre Erweiterung unter Gewährleistung von Konsistenz und Widerspruchsfreiheit ist keine triviale Aufgabe und erfordert sowohl ein komplexes Wissen über die zu modellierende Domäne als auch über die möglichen Probleme innerhalb der Modellierung mit Ontologien wie z.B. Redundanzen, Zirkelschlüsse oder semantische Fehler.

Diskussion der verschiedenen Modellierungsmethoden

Es muss beachtet werden, dass jedes Modell einen anderen Anspruch an das Verständnis des Modellierers stellt. Nicht alle Modelle sind geeignet, um auch von Nicht-Informatikern soweit verstanden zu werden, dass diese genutzt und erweitert werden können. Betrachtet man dies aus der klassischen Sicht, bei der der Nutzer ein reiner Konsument der Dienste und der Situationsbeschreibungen ist, so stellt dies kein Problem dar. Es setzt jedoch voraus, dass jede Änderung an Dienst- und Situationsbeschreibungen durch eine Fachkraft erfolgen muss und der Nutzer nicht selbst aktiv werden kann. Da dies der Komplexität und Vielzahl an Diensten und Situationsbeschreibungen entgegensteht, ist die Verständlichkeit für Laien eine Anforderung an die Modellierungsmethode.

Es ist festzustellen, dass Modelle, die sich verständlich graphisch darstellen lassen, wie zum Beispiel die objektbasierten Modelle, Vorteile für nicht versierte Anwender haben. Eine Einarbeitung in die Strukturen von RDF oder anderen komplexen Werkzeugen zur Modellierung kann von einem Standardnutzer nicht erwartet werden. Für eine Nutzung wären hier mächtige Werkzeuge notwendig, die es dem Nutzer ermöglichen, die Modelle und Beziehungen zu verstehen und nach einer Bearbeitung die Prüfung und Korrekturen der Eingaben durchzuführen. Dennoch haben gerade die komplexen Methoden den Vorteil, dass sie es erlauben, Daten strukturiert und wiederverwendbar aufzubereiten. Daher sollte ein Ansatz gewählt werden, der es erlaubt, komplexe Strukturen abzubilden, zum Beispiel die logikbasierten Systeme und regelbasierten Systeme, und der auch geeignet ist, um dem Nutzer verständlich die Zusammenhänge von Kontextinformationen darzustellen. So erhält dieser neben der Möglichkeit zur Anpassung existierender Beschreibungen auch die Fähigkeit zur Kontrolle, auf Basis welcher eventuell schützenswerter Daten agiert wird.

2.2 Dienstorientierung als Paradigma für eine Universität

Da diese Arbeit eine Anwendung im Hochschulumfeld adressiert, ist es notwendig, einige Organisationsaspekte von Hochschulen auszuführen. Betrachtet man eine Hochschule auf Basis ihrer Strukturen und Abläufe, so ergibt sich ein komplexes und vielschichtiges System mit einer großen Anzahl unterschiedlicher Akteure mit verschiedenen Zielstellungen. Gemein ist allen Akteuren, dass sich ihre Arbeitsabläufe in Prozesse zerlegen lassen, die unterschiedlich viele Akteure einbinden.

Gablers Wirtschaftslexikon definiert den Begriff Prozess wie folgt:

> Unter Prozess versteht man die Gesamtheit aufeinander einwirkender Vorgänge innerhalb eines Systems. So werden mittels Prozessen Materialien, Energien oder auch Informationen zu neuen Formen transformiert, gespeichert oder aber allererst transportiert.
>
> (Gabler-Wirtschaftslexikon [Berwanger 18])

Für die vorliegende Arbeit wird der Begriff des Prozesses weiter gefasst und wie folgt definiert:

Definition 3 (Prozess)

> Ein Prozess besteht aus einer Abfolge von Aktivitäten, den Prozesschritten, die unter Beteiligung unterschiedlicher Akteure der Erreichung eines Ziels dienen. Die Prozessschritte sind miteinander logisch verknüpft und können als Gruppe zu einem Untersprozess zusammengefasst werden. Teilweise kann ein einzelner Prozessschritt auch in mehrere kleinere Prozessschritte zerlegt werden. Die Abbildung eines Prozesses als Graph mit den Übergängen und Bedingungen zwischen den einzelnen Prozessschritte wird als Prozessplan verstanden. Komplexe Prozesse können mehere unabhängige Prozesspläne haben, die jeweils bestimmte Akteure oder Aspekte fokussieren.

So lässt sich das Ablegen einer Prüfung als Prozess betrachten, der neben dem zuständigen Prüfungsamt auch den prüfenden Dozenten als Akteur involviert. Dieser Prozess lässt sich weiter unterteilen und so ergeben sich die sequenziell ablaufenden Prozessschritte der Prüfungsanmeldung und -zulassung, der Prüfungsdurchführung und der Zertifizierung. Diese Prozessschritte lassen sich als Dienste betrachten, die von den jeweiligen Akteuren angeboten werden. Der Vorteil dieser Sichtweise besteht darin, dass sich durch einheitliche Prozessstrukturen und Dienste eine Normierung und Standardisierung ergibt. So können über unterschiedliche Fakultäten hinweg gleiche Qualitätskriterien angesetzt werden, um zum Beispiel die Rechtssicherheit für Prüfungsabläufe zu erhöhen. Gleichzeitig erlaubt eine Dienstorientierung auch Synergieeffekte und Skalierung, da sich so Akteure identifizieren lassen, die einen Dienst besonders effizient abwickeln. Die Prozessstrukturen können so angepasst werden, dass diese Akteure ihre Aufgaben exklusiv übernehmen und damit andere Akteure, die diesen Dienst früher parallel angeboten haben, entlasten. Als Beispiel wäre vorstellbar, dass alle Prüfungsanmeldungen über das Prüfungsamt der Fakultät laufen, da hier auch die Planung der Prüfungsräume und -zeiten erfolgt. Dies kann Dozenten entlasten, die nun die Anmeldung von Studenten nicht mehr weiterleiten müssen, sondern stattdessen direkt durch das Prüfungsamt über die zugelassenen Kandidaten informiert werden und zeitgleich Vorschläge für Prüfungsraum und -zeit erhalten. Neben Skalierungseffekten, wie dem der Entlastung der Dozenten, ergeben sich hier auch Synergieeffekte, wie die höhere Rechtssicherheit durch die kürzeren Wege und die Konzentration der Anmeldungen bei den zuständigen Mitarbeitern.

Dieses Paradigma zur Standardisierung und Effizienzsteigerung ist auch in der Informatik verbreitet. Unter einer Architektur mit Dienstorientierung (SOA, engl. Service-oriented Architecture) wird hier ein spezielles Architekturparadigma verstanden, das sich besonders für

heterogene und verteilte Strukturen eignet. Es zeichnet sich durch eine lose Kopplung von Software-Diensten aus, die durch einen Verzeichnisdienst oder einen Vermittler offen auffindbar sind. Die Dienste kapseln dabei modulare Funktionalitäten, die auch mit anderen Diensten kombiniert werden können. Dies wird durch die in maschinenlesbarer Form offen verfügbaren Schnittstellendefinitionen ermöglicht, die auch zur Laufzeit ein dynamisches Einbinden der Dienste erlauben. In der Literatur wird eine SOA auch wie folgt definiert:

> Unter einer SOA versteht man eine Systemarchitektur, die vielfältige, verschiedene und eventuell inkompatible Methoden und Applikationen als wiederverwendbare und offen zugreifbare Dienste repräsentiert und dadurch eine plattform- und sprachenunabhängige Nutzung und Wiederverwendung ermöglicht.
>
> (Ingo Melzer [Melzer 07, Seite 11])

Als Verknüpfung zwischen Hochschulen und Dienstorientierung wird im folgenden Abschnitt auf bereits durchgeführte Arbeiten zur Übertragung des SOA-Gedankens auf Hochschulen als Ganzes eingegangen. Anschließend werden Dienstbegriffe untersucht und auf die Hochschule angewandt. Abschließend werden die Rollen des Nutzers und schließlich die des Anbieters von Diensten betrachtet.

2.2.1 Dienstorientierung in Hochschulen in vorangegangenen Arbeiten

Aus bisherigen Forschungs- und Entwicklungsarbeiten ist der Einsatz von Service-orientierten Architekturen in Hochschulen bereits bekannt, z.B. in [Tavangarian 09b, Zender 10]. Grundlage ist dabei das Konzept der Pervasive University.

> Eine Pervasive University ist eine Bildungseinrichtung, die zielführend mit Mechanismen und Geräten des Pervasive Computings angereichert ist. Aus Sicht der Applikationen ist es eine Universität mit einer nahtlosen IT-Unterstützung in allen Aktivitätsfeldern: eLearning, eTeaching, eScience und eAdministration. Aus technischer Sicht ist es eine Pervasive-Computing-Umgebung, in der die Komponenten und Interaktionsmuster auf die Charakteristika einer Hochschule adaptiert wurden.
>
> (übersetzt nach [Tavangarian 09b])

Ein Kernkonzept umfasst dabei die Nutzung von SOA, da sich mit diesem Paradigma die gegenseitigen Abhängigkeiten der Systeme reduzieren lassen und sich die Systeme zur Laufzeit dynamisch erweitern und neu konfigurieren lassen. Des Weiteren wird Peer-to-Peer Computing, also die direkte Kommunikation zwischen verschiedenen, autonomen Knoten ohne zentrale Kontrolle, als Konzept identifiziert, mit dem sich Nutzer thematisch vernetzen lassen ohne ihre Privatsphäre zu verletzen. Abschließend wird die Selbstorganisation genannt, also die Verwaltung von Systemen auf Basis einfacher Regeln durch sich selbst, um spontan Lerngruppen zu formen oder die Infrastruktur an sich ändernde Situationen anzupassen, wie sie zum Beispiel bei einer auf dem Campus stattfindenden Konferenz vorkommen.

Darauf aufbauend wurde in den Dissertationsschriften von Zender und Dressler Dienstorientierung als Basis für Anwendungen im Bereich des Lehr- und Lernumfelds vorgeschla-

gen [Zender 10, Dressler 10]. Kernstück ist das Konzept des General Purpose Access Points (GPAP), der zwischen verschiedenen Diensttechnologien als zentraler Übersetzer dienen soll. Auf dessen Netzwerkebene wird eine Paketvermittlung zwischen verschiedenen drahtgebundenen und drahtlosen Technologien wie ZigBee, WLAN oder Bluetooth vorgestellt [Dressler 10]. Für die darüberliegende Service-Ebene wurde mit STiL eine Meta-Sprache entwickelt, die ein Entdecken und Verbreiten von Diensten über verschiedene Diensttechnologien wie zum Beispiel Web Services, UPnP und Jini hinweg ermöglichen soll [Zender 10]. Der Schwerpunkt dieser Arbeiten lag auf der IT einer Hochschule, der Vereinheitlichung der unterschiedlichen Technologien und der Nutzung in verschiedenen Lernszenarien. Es ging daher nicht um die Inhalte selbst, sondern um die Middleware-Dienste und Technologien, die Inhalte anbieten.

In dieser Arbeit wird von der technischen Ebene der Dienstinfrastruktur abstrahiert und der Nutzer der IT sowohl als Verbraucher als auch als Produzent und Vermittler von Diensten hinzugefügt. Daher muss das Konzept der Pervasiven Hochschule um eine soziale Sicht erweitert werden. Damit werden neben den Diensten, die Lerninhalte anbieten, die Lerninhalte selbst zu Diensten. Zusätzlich lassen sich so auch Elemente und Vorgänge aus den Forschungs- und Verwaltungsprozessen als Dienste einbinden. Die IT wird nicht länger losgelöst von ihrem Anwender betrachtet, sondern der Nutzer wird zu einem Kernpunkt in der Dienstvermittlung, da erst der Nutzer begründet entscheiden kann, ob ein Dienst für ihn relevant ist.

Dabei werden die Dienste in definierte Prozesse eingebunden. Diese Prozesse existieren bereits an den Hochschulen in Form von Verwaltungsabläufen und der Studienorganisation und sind ihren Nutzern daher als Strukturen bekannt.

2.2.2 Definition und Klassifikation von Diensten

Es gibt für den Begriff Dienst entsprechend seines Anwendungsbereichs unterschiedliche Definitionen. Bezugnehmend auf den vorliegenden Anwendungsfall der Informatik bietet sich folgende Definition an:

> Ein Dienst ist ein Programm oder auch eine Softwarekomponente, die lokal oder über ein Netzwerk von anderen genutzt werden kann.
>
> (Ingo Melzer [Melzer 07, Seite 12])

Diese Definition ist sehr abstrakt gehalten und bezieht sich somit sowohl auf systemnahe Dienste, wie die Druckerwarteschlange, als auch auf Programme, die vom Anwender direkt genutzt werden können, zum Beispiel zum Versenden von E-Mails. Orientiert man sich in Richtung der Dienste, die im Rahmen von dienstorientierten Architekturen verwendet werden, bietet sich folgende Definition an:

> Services are autonomous, platform-independent computational entities that can be used in a platform independent way. Services can be described, published, discovered, and dynamically assembled for developing massively distributed, interoperable, evolvable systems. Services perform functions that can range from answering simple requests to executing sophisticated business processes requiring peer-to-peer relationships between possibly multiple layers of service consumers and providers.

(Michael P. Papazoglou et al. [Papazoglou 06])

Die erwähnten Entwurfsmerkmale beinhalten, dass ein solcher Dienst über eine standardisierte Schnittstelle verfügt und eine gekapselte Funktionalität darstellt. Dies bedeutet, dass er keine Details über seine Funktionsweise kommuniziert, sondern nur spezifiziert, welche Informationen als Eingabe erwartet werden und wie die Ausgabe formatiert ist. Dies kann von einer einfachen Konvertierung zwischen zwei physikalischen Größen wie der Umrechnung von Celsius in Fahrenheit bis zu einem komplexen Prozess wie der Buchung eines Fluges oder der Belastung einer Kreditkarte reichen. Die Komplexität entsteht hier durch die Anzahl der intern notwendigen Aktionen und die Menge der eingebundenen und eventuell sogar externen Dienste. Somit kann ein einzelner Dienst intern wiederum einen Prozess abbilden, der aus weiteren Diensten besteht.

Diese Aufgaben werden im Bereich der dienstorientierten Architekturen aus sogenannten Geschäftsprozessen abgeleitet und stellen in diesen einen einzelnen Schritt dar. Sie stammen aus der Betriebswirtschaft, in der auch der naheliegende Begriff Dienstleistung definiert ist:

> Dienstleistung, nicht stoffliche, nicht körperliche Leistung (immaterielles → Gut). Dienstleistungen sind z.B. Bank-, Transport-, Versicherungsleistungen oder Unterhaltungsleistungen. Sie sind v.a. durch Gleichzeitigkeit von Produktion und Verbrauch und dadurch gekennzeichnet, dass sie nicht lager- und transportfähig sind. - Gegensatz: → Sachgut.

(Gabler Kompakt-Lexikon Wirtschaft [Gab 13, Seite 99])

Es ist zu beachten, dass es Parallelen zwischen dem Dienst und der Dienstleistung gibt. So ist beiden gemeinsam, dass das Ziel die Erfüllung einer konkreten Aufgabe ist, und dass der Verbrauch oder die Eingabe direkt in eine Produktion bzw. Ausgabe überführt wird. Dabei ist unerheblich, wie dies geschieht.

Da eine Universität neben bisher betrachteten IT-Diensten sowohl Dienstleistungen ohne Sachgüter (z.B. Studienberatung) als auch Dienstleistungen mit physischem Bezug anbietet (z.B. Mensa, Druckerei), muss bei der Betrachtung der Integration dieser Dienste in die Abläufe der Nutzer von einem allgemeineren Dienstbegriff ausgegangen werden.

Die folgende angepasste Definition versucht daher, die IT-Dienste und die Dienstleistungen in einem Begriff zu vereinen:

Definition 4 (Dienst)

> Ein Dienst ist eine Entität mit einer physischen oder virtuellen Schnittstelle, die eine Interaktion anbietet und dabei entweder eine Information über ein Objekt liefert oder eine Manipulation des Objektes durchführt. Dabei kann das Objekt auch selbst eine oder mehrere Dienste umfassen.

Um möglichst viele verschiedene Dienste im Konzept berücksichtigen zu können, wurde im Rahmen dieser Arbeit, eine Klassifikation der verschiedenen Dienste an einer Hochschule erstellt. Im Folgenden werden diese Klassifikationen mit ausgewählten Vertretern vorgestellt.

Klassifikation nach den Endstellen: dem Nutzer und dem Anbieter eines Dienstes

	Nutzer			
Anbieter		Mensch	Software	Gerät
	Mensch	Beratung im Studienbüro	Bibliotheksrecherche, Lernmanagement	Kopieren
	Software		Virenschutz, VPN-Zugang	Drucken
	Gerät	PC-Wartung, Laborgeräte-Bereitstellung	WLAN-Zugang, automatische Verbindungserstellung für mobile Geräte	Stromversorgung

Tabelle 2.1: Klassifikation von typisch universitären Diensten nach der Art ihrer Nutzer

Bei dem Begriff Endstelle handelt es sich um die Entität, die den Dienst nutzt, beziehungsweise den Dienst bereitstellt oder selbst der Anbieter ist. Dabei kann unterschieden werden zwischen einem Menschen, einer Software und einem technischen Gerät. Dargestellt sind in Tabelle 2.1 exemplarisch Dienste, die eine Endstelle von einem Anbieter anfragen könnte.

Es zeigt sich, dass diese Art der Klassifikation ungeeignet ist, da sie diverse offensichtliche Nachteile hat. Die Gruppierung erlaubt nur die drei Einteilungen nach Mensch, Gerät und Software, wobei hier zu hinterfragen ist, inwiefern sich Software und Gerät wirklich trennen lassen. Des Weiteren ist hier eine Verfeinerung nicht möglich ohne eine der anderen Klassifikationen einzubinden.

Relationsklassifikation

Wie sich bei der Diskussion des Begriffs Kontext gezeigt hat, sind die Beziehungen, die zwischen den Objekten bestehen gut geeignet, um ihre Situationen zu beschreiben. Daher wird dies genutzt, um hier im Folgenden eine Klassifikation entsprechend der Kontextdefinition von Dey vorzunehmen. Es wird dabei zwischen den Kategorien des Ortes, der Identität des Nutzers und der Aktivität des Nutzers bei Verwendung eines Dienstes unterschieden. In den Tabellen sind somit Beispiele für die jeweilige Relation, dazu passende durch Software realisierbare Dienste und die jeweilige Zielgruppe aufgeführt. Auch diese Art der Klassifikation ist nur als Vorschlag zu verstehen und ist speziell für den Anwendungsfall einer Hochschule strukturiert.

Ort	Dienst	Zielgruppe
Campus	Internetzugang	Alle Nutzer und ggf. Gäste
Vorlesungsraum	Raumsteuerung, -belegung/-reservierung	Studierende, Dozenten, techn. Mitarbeiter
Studienbüro	Veranstaltungsmanagement, Prüfungsverwaltung	Administrative Mitarbeiter
Mensa	Essenspläne	Alle
Bibliothek	Zugriff auf Forschungsdatenbanken, Literatursuche	Alle
	Bibliotheksverwaltungs-system, Mahnwesen	Mitarbeiter
Uniklinikum	Internetzugang, Raum-/Gebäudepläne	Alle
	Personalpläne, Patientenakten	Mitarbeiter

Tabelle 2.2: Klassifikation von typisch universitären Diensten nach den Orten ihrer Nutzung

Ortsrelation zur Klassifizierung von Diensten

Der Ort des Nutzers stellt einen besonderen Kontext dar, da sich ein Nutzer zwar physisch nur an einer Position befinden, diese Position jedoch je nach Person eine Menge an Orten umfassen kann. Während sich die Position mit technischen Mitteln bestimmen lässt, ist der Ort an dem der Nutzer ist nicht zwingend eindeutig. Ein Nutzer kann sich durch aus gleichzeitig auf den Campus, im Fakultätsgebäude und im Studienbüro befinden.

In Tabelle 2.2 ist exemplarisch dargestellt, an welchen physischen universitären Orten Dienste angeboten werden und wer hier als Zielgruppe zu sehen ist. Neben den dargestellten physischen Orten sind auch logische Orte, wie zum Beispiel eine medizinische Fakultät, sinnvoll zur Klassifikation von Diensten. Diese können dabei ganze Areale wie einen eigenen Campus und große Gebäude wie ein Universitätsklinikum umfassen und sind somit noch unspezifischer. Des Weiteren lassen sich mit der Ortsrelation auch unscharfe Bereiche beschreiben, wie der Bereich „In der Nähe der Haltestelle Mensa“.

Diese Klassifikation ist geeignet, um die Menge der nutzbaren Dienst aus der Gesamtmenge zu reduzieren, da, wie bereits erwähnt, ein Nutzer sich nur an einem physischen beziehungsweise in einer begrenzten Menge logischer Orte aufhalten kann. Jedoch zeigt sich auch, dass durch Orte allein die Zielgruppenspezifizierung schwer wird, da jene Dienste, die an Orte gebunden sind, häufig für alle oder viele Nutzergruppen zur Verfügung stehen. Dass ein Nutzer sich an einem bestimmten Ort befindet, erlaubt noch keinen gesicherten Rückschluss auf seine Rolle oder darauf, Teil welcher Zielgruppe er ist. Als Beispiel sei hier das Uniklinikum genannt. Die

Anwesenheit erlaubt keinen Rückschluss darüber, ob man Patient, Besucher, Pfleger, Arzt oder Techniker ist. Gleichzeitig gibt es Orte, wo der überwiegende Teil der Anwesenden auch Zielgruppe der dortigen Dienste ist. Ein Beispiel sein hier die Mensa genannt, mit dem Dienst Essensplan.

Rollenrelation

Bei der Klassifikation nach den Rollen eines Nutzers innerhalb eines Prozesses wird berücksichtigt, dass ein Nutzer diesen Dienst stets in einer von ihm ausgeübten Rolle nutzt. Dabei kann ein Nutzer gleichzeitig mehrere Rollen in sich vereinen.

Rolle	Dienste	Zielgruppe
Lehrender	Prüfungen abnehmen, Vorlesungen halten	Dozenten, wiss. Mitarbeiter
Forscher	Nutzung spezieller Datenbanken, Software	Dozenten, wiss. Mitarbeiter, stud./wiss. Hilfskräfte
Studierender bzw. Lernender	Vorlesungsteilnahme, Hausaufgabenabgabe, Prüfungsanmeldung	Studenten, Gasthörer
Verwaltender	Reisekostenabrechnung, Beschaffung von Verbrauchsmaterial, Prüfungsorganisation	Verwaltungsangestellte, wiss. Mitarbeiter
Technischer Unterstützer	Administration der IT-Dienste, Unterstützung der Mitarbeiter	Technische Mitarbeiter
Campus-Zugehöriger	Internetzugang, Veranstaltungsinformationen	Alle Nutzer und Gäste

Tabelle 2.3: Klassifikation von typisch universitären Diensten nach den Rollen ihrer Nutzer

Tabelle 2.3 zeigt eine exemplarische Auswahl von universitären Rollen, genutzten Diensten und den möglichen Besitzern der Rolle und somit der Zielgruppe.

Dabei zeigt sich, dass Verknüpfungen zwischen Rollen und Zielgruppen bestehen, was eine Klassifikation erleichtert. Jedoch ist die Rolle, die ein Nutzer hat, nicht zeitlich stabil. Somit eignet sich die Rolle gut zur Bestimmung von geeigneten Diensten, ist jedoch selbst nicht trivial von außen zu bestimmen. Legt ein Nutzer im Rahmen eines Profiles, z.B. in einem Lernmanagementsystem, fest, welche Rollen er hat, so ist klar welche Rolle er hat. Wollte

man dies nur aus seinen Aktivitäten ermitteln, so wäre dies mit einer großen Unsicherheit behaftet.

Aktivitätsrelation

Aktivität	Prozessplan	Dienst
Lehren	Vorlesungsplan	Raumsteuerung, StudIP
Zum Essen gehen	Tagesablauf, privater Kalender, Gruppenkalender	Mensaplan, Raumbelegung, Nahverkehrsinformationssysteme
An Vorlesung teilnehmen	Prüfungsplan, Vorlesungsplan, Veranstaltungsplan	Stud.IP, Hochschulinformationssystem (HIS), Vorlesungsverzeichnis (LSF)
Betreuer für eine Hausarbeit finden	Prüfungsordnung, Studienordnung	LSF, HIS, StudIP
Zu einer Vorlesung einschreiben	Anmeldung bei Uni-Infrastruktur, Studienordnung	Bibliotheksregistrierung, Rechenzentrum, LSF, HIS
Mit Kommilitonen treffen	Tagesablauf, privater Kalender	Soziale Netzwerke, Informations- und Unterhaltungsdienste

Tabelle 2.4: Klassifikation von typischen universitären Diensten nach den Aktivitäten und Prozessplänen die sie unterstützen

Der Aktivität eines Nutzers lassen sich Prozesse zuordnen. Dabei verfügt jeder Prozess über Prozesspläne, die die Prozesschritte (Aktivitäten) zur Erreichung des Prozessziels strukturieren und die vergangenen Arbeitsschritte mit den künftigen in eine Relation setzen. Nicht alle Prozesse lassen sich abschließend beschreiben. So gibt es zwar Prozesse, die durch Regelwerke beschrieben werden. Als Beispiel sei hier der Prozess der Lehre einer Hochschule mit dem Vorlesungsplan oder der Prozess der Einschreibung zu einem Kurses genannt. Neben diesen Prozessen mit standardisierten und verfassten Regelwerken, Akteuren und Strukturen gibt es jedoch auch Prozesse die stattfinden, ohne dass sie ausformuliert sind und von Prozessteilnehmer zu Prozessteilnehmer unterschiedlich sind. Als Beispiel sei hier das Studium allgemein genannt, das zwar einer grundlegenden Strukturierung unterliegt, jedoch als Prozess betrachtet von Student zu Student unterschiedlich und individualisiert ist. Man kann hier von einem

Metaprozess sprechen, der über andere Prozesse hinweg stattfindet ohne selbst ein im Voraus definiertes Ziel zu haben.

Des Weiteren können Prozesse sowohl statisch als auch dynamisch sein. Statisch in dem Sinne, dass eine Einschreibung zu einem Studium definierten Regeln und Abläufen folgt, während ein Studium diverse Wahlmöglichkeiten bietet und auf Basis von Anerkennungsanträgen auch die Möglichkeit existiert Aktivitäten anderer Studiengänge einzubinden. Ein Nutzer kann sich gleichzeitig in verschiedenen Prozessen befinden. Als Beispiel sei hier das Studieren von einem Haupt- und einem Nebenfach gegeben, oder auch das Absolvieren eines oder mehrerer Auslandssemester. Diese Prozesse können dabei sowohl einmalig, mehrmalig als auch zeitunabhängig aufgeführt werden.

Bei dieser Klassifikation, die beispielhaft in Tabelle 2.4 ausgeführt ist, zeigt sich, dass es für verschiedene vorhandene Dienste mehrere Möglichkeiten der Zuordnung zu einer Aktivität gibt. Dies ergibt sich daraus, dass es in manchen Diensten, wie zum Beispiel dem Stud.IP, mehrere Funktionen gibt, die innerhalb eines zugeordneten Prozessplans durch diesen Dienst realisiert werden können. Einige Prozesspläne sind reine Kalender, in denen die Dienste bei bestimmten Einträgen weitere Informationen geben. Andere Prozesspläne sind komplexer und beinhalten, wie am Beispiel der Studienordnungen, auch Verzweigungen (z.B. die Wahl einer Vertiefungsrichtung) und Regeln zu Sonderfällen, zum Beispiel wie eine krankheitsbedingte Abwesenheit bei einer Prüfung nachzuweisen und zu werten ist.

Aktivitäten und Prozesspläne eignen sich gut um Dienste zu gruppieren, da die Dienste einzelne Funktionalitäten kapseln, die in verschiedenen Plänen wiederverwendet werden. Da an Hochschulen die Prozesse eng umrissen, kaum individualisiert und einige durch Ordnungen fixiert sind, bietet die Gruppierung der Dienste nach diesen Prozessen und ihren Aktivitäten eine sinnvolle Grundlage für die Empfehlung von Diensten.

Diskussion der Klassifikation nach Relationen

An den bisherigen Beispielen wird deutlich, dass eine 1:1-Zuordnung zwischen Diensten und einzelnen Kontextinformationen nicht sinnvoll ist. Dienste können jeweils einem Vektor von Kontextinformationen zugeordnet werden. Der Dienst einer Mensa ist relevant für jene Nutzer, die in der Nähe sind (Ortsrelation) und jene, die regelmäßig dort essen (Aktivitätsrelation). Dieser Dienst hat jedoch auch noch Restriktionen, wie zum Beispiel die Öffnungszeiten.

Die Kombination verschiedener Relationen ermöglicht die Formulierung von Regeln, die es erlauben eine Dienstauswahl auf Daten durchzuführen, über die die Nutzer verfügen beziehungsweise die die Nutzer über sich anonym preisgeben könnten. Es ist daher einer Lösung vorzuziehen, bei der aus erhobenen Nutzerdaten Profile gebildet werden, die dann für eine Empfehlung genutzt werden würden.

2.2.3 Definition und Klassifikation der Nutzer von Diensten

Betrachtet man den Benutzer eines Systems genauer, so ergibt sich auch hier die Notwendigkeit einer Definition. Gablers Wirtschaftslexikon definiert den Benutzer wie folgt:

> Ungenauer, selten definierter Begriff aus der Informatik; häufig verwendet im Software Engineering. Allg. derjenige, der von einem Softwareprodukt oder auch nur von einer Softwarekomponente Gebrauch macht; muss nicht zwingend ein menschlicher Benutzer sein (menschliche Benutzer werden deshalb auch als Endbenutzer bezeichnet). Der Begriff wird auch auf andere Softwarekomponenten ausgedehnt, z.B. der Benutzer eines Moduls (i.Allg. ein anderes Modul).
>
> (Gabler-Wirtschaftslexikon [Lackes 17])

Für die vorliegende Arbeit wird der Begriff des Nutzers auf den Dienstbegriff hin angepasst.

Definition 5 (Nutzer)

> Ein Nutzer ist eine Person, die auf einen Dienst über seine Schnittstelle, egal ob physisch oder virtuell, zugreift und ihn persönlich verwendet.

Dies impliziert, dass der Nutzer eines Dienstes die Schnittstelle des Dienstes kennt und mit der Nutzung des Dienstes ein Ziel verfolgt. Diese Ziele lassen sich Nutzergruppen zuordnen, in die sich die Nutzer auf Grund ihrer Rolle einordnen lassen. Nutzergruppen lassen sich hierarchisch organisieren und können so Personen enthalten, die unterschiedliche Rollen einnehmen.

Nutzer	Vertreter	Aufgabenbereiche
Dozenten	Professoren, Lehrkräfte, technische Mitarbeiter, Laboringenieure	Lehre, Forschung und Entwicklung, Gremienarbeit, Koordination und Verwaltung
Wissenschaftliche Mitarbeiter	Projektmitarbeiter, stud./wiss. Hilfskräfte	Lehre, Forschung und Entwicklung, Gremienarbeit
Nicht-Wiss. Mitarbeiter	Administratoren, Verwaltungskräfte	Koordination und Verwaltung, Technische Unterstützung
Studenten	Haupthörer, Promotionsstudenten	Studium, Forschung und Entwicklung
Externe	Gäste, Dienstleister	Information und Kommunikation

Tabelle 2.5: Klassifikation von typischen, universitären Nutzern nach Vertretern und Aufgabenbereichen

Um den verschiedenen Nutzergruppen Dienste zuordnen zu können, müssen die Nutzer entsprechend ihrer Ansprüche klassifiziert werden.

Während sich durch die Gruppierung der Vertreter über die Nutzergruppen in Tabelle 2.5 im Vergleich zur Gruppierung nach den Rollen der Nutzer keine Verfeinerung bietet, lässt sich jedoch ein Mehrwert in der Gruppierung generieren, wenn man den Grad der Bekanntheit des

Nutzers selbst als Kriterium nimmt und somit eine Einteilung nach bekanntem Einzelnutzer, Nutzer innerhalb einer Gruppe und anonymen Nutzer vornimmt.

Grad an Anonymität	Vertreter	Dienste
bekannter Nutzer	namentlich identifizierte Personen	persönliche Weckfunktion, Übersicht der persönlich verbrauchten Reisemittel
Nutzer innerhalb einer bekannten Gruppe	Studenten einer Lehrveranstaltung, Dozenten einer Fakultät	Informationen und Materialien zu einerLehrveranstaltung, Dienstreiserichtlinien, etc.
unbekannte / anonyme Nutzer	Nutzer an öffentlichen Terminals, Nutzer mit Position auf dem Campus	Hinweise zu nahen Haltestellen des öffentlichen Nahverkehrs, Informationen zu Gebäuden und Geschichte

Tabelle 2.6: Klassifikation von typischen, universitären Diensten nach dem Grad der möglichen Anonymität

Es zeigt sich, dass sich hier die Möglichkeit bietet, auf die Unschärfe in der Situationsbestimmung des Nutzers zu reagieren und ein System somit robust für Erweiterungen zu machen. Tabelle 2.6 zeigt eine Klassifikation typischer universitärer Dienste zu drei Nutzergruppen, die sich durch ihren Grad an Anonymität unterscheiden. Wenn über einen Nutzer nicht bekannt ist, welchen Gruppen er angehört oder wenn er überhaupt keine Informationen über sich preisgeben möchte, so lässt er sich in die unterste Kategorie einordnen und kann Dienste angeboten bekommen, die sich nur auf die vorhandenen Kontextinformationen über Zeit und gegebenenfalls Ort der Diensterbringung stützen. Zielgruppe dieser Dienste ist die Allgemeinheit, zu der jeder Dienstnutzer gehört. Lässt sich ein Nutzer über seine bekannt gegebenen Kontextinformationen (z.B. über das Lernmanagementsystem) einer Gruppe zuordnen, so lassen sich hier, zusätzlich zu den Diensten für die Allgemeinheit, spezifisch an die Gruppe angepasste Dienste anbieten. Analog könnte er selbst auch Dienste für diese Gruppe anbieten. Dazu muss dem System, das die Dienste vermittelt, seine genauere Identität nicht bekannt sein. Ist ein Nutzer dem System genau bekannt und hat dieser eigene persönliche Dienste im System, die nur für ihn zugreifbar sein sollen, so lässt sich auch dies realisieren.

2.2.4 Definition und Klassifikation der Anbieter von Diensten

Im Bereich der Service-orientierten Architekturen ist der Anbieter eines Dienstes (Service-Provider) auch derjenige, der für Inhalt und Funktion die Verantwortung trägt. Er ist auch derjenige, der einen Dienst bei einem Vermittler (Broker) registriert. Die Rolle des Anbieters eines Dienstes kann im Zusammenhang mit der vorliegenden Arbeit zwei verschiedene Bedeutungen haben. Zum einen ist ein Anbieter jene Entität, die einen Dienst verfügbar macht.

Dienstanbieter	Dienstnutzer	Dienste
Bekannter Nutzer	Der Nutzer selbst	Persönliche Weckfunktion innerhalb eines Kalenders
	Mitglieder seiner Gruppen	Prüfungsergebnisse für eigene Veranstaltungen
	Anonyme Nutzer	Hinweise zu Sprechzeiten
Nutzer innerhalb einer Gruppe	Weitere Mitglieder der Gruppe	Mitschriften zu Lehrveranstaltungen der Gruppe
	Anonyme Nutzer	Allgemeine Hinweise ohne Gruppenbezug
Unbekannte / anonyme Nutzer	Andere anonyme Nutzer	Hinweise zu nahen Haltestellen des öffentlichen Nahverkehrs, Informationen zu Gebäuden und Geschichte

Tabelle 2.7: Klassifikation von typischen, universitären Diensten nach den Anbietern und Nutzern

Aus der Perspektive einer SOA ist dies der Service-Provider, der am Beispiel einer Webseite durch den Web-Server vertreten wird.

Zum anderen ist der Anbieter aber auch zum Teil jene Entität, die diesen Dienst gegenüber Dritten bekannt macht, diesen Dienst also zum Beispiel in ein (Dienst-)Verzeichnis einträgt (publish), damit er von anderen genutzt werden kann. In der SOA wird diese Rolle ebenfalls vom Service-Provider wahrgenommen, der das Publizieren beim Broker durchführt. Im Rahmen dieser Arbeit muss dieser Schritt nicht vom Autor eines Dienstes vorgenommen werden. Im weiteren Verlauf der Arbeit werden beide Rollen, der Autor und der Publizist, unter der Bezeichnung Anbieter vereinigt.

Basierend auf der in Abschnitt 2.2.3 verwendeten Klassifikation für Nutzer werden auch die Anbieter des Systems nach ihrem Bekanntheitsgrad innerhalb des Systems klassifiziert.

Tabelle 2.7 stellt dies beispielhaft dar.

Die Identität des Anbieters eines Dienstes muss, wie hier exemplarisch gezeigt, nur selten auf die Person zurückgeführt werden können. Wenn ein Dienst die Information anbietet, welche Ergebnisse in einer Prüfung erzielt wurden, sollte die Identität des Anbieters sich sicher bis zum Dozenten zurückverfolgen lassen, um glaubhaft zu sein. Wenn jemand Mitschriften zu einer Vorlesung anbietet oder ein zugehöriges Forum anbietet, so muss hier die Identität nicht zwingend bekannt sein, jedoch sollte gesichert sein, dass der Anbieter an dieser Vorlesung auch teilnimmt. Wenn jemand Informationen oder Funktionen anbietet, bei denen die Identität des Anbieters keinen Einfluss auf die Qualität oder Glaubhaftigkeit der Informationen oder Funktionen hat, so ist es nicht notwendig hier eine Identität anzugeben. Dies ist der Fall bei Diensten wie zum Beispiel einer Webseite mit Hinweisen zum öffentlichen Nahverkehr.

2.2.5 Zusammenfassung der Erkenntnisse aus den Klassifikationen

Es hat sich gezeigt, dass die Klassifikation der Dienste nach ihren Anbietern und Nutzern als Kriterium der Gruppierung von Diensten nicht geeignet ist, da hier die Semantik des Begriffs Dienst sehr unterschiedlich wahrgenommen werden kann und die Nutzer bzw. Anbieter somit nicht eindeutig zugeordnet werden. Diese Art der Klassifikation lässt jedoch einen Rückschluss auf die Art der Kommunikation zwischen dem Anbieter und dem Nutzer zu. So kann die Kommunikation ohne einen menschlichen Teilnehmer ausschließlich auf Netzwerk- oder Softwareebene ablaufen.

Bei den Relationsklassifikationen zeigt sich besonders die Ortsrelation als geeignet, um Dienste zu gruppieren, da der Ort eines Dienstes für einen Nutzer als Präferenzkriterium hinzugezogen werden kann und auch bei sonst gleichwertigen Diensten eine Priorisierung erlaubt. Des Weiteren zeigt sich die Identitätsrelation als geeignet, um innerhalb von Gruppen auf die verschiedenen Bedürfnisse von Rollen (z.B. Dozent und Student in der Gruppe der Teilnehmer einer Veranstaltung) einzugehen. Dieses Kriterium ist daher neben der Dienstauswahl auch für die Ermittlung des anzubietenden Dienstumfangs geeignet. Die Aktivitätsrelation zeigt, dass die Einordnung von Diensten in Prozesspläne und die Zusammenführung in komplexen Prozessen sinnvoll sein kann, um Dienste anzubieten, deren Nutzen und Relevanz zeitlich eng begrenzt sind.

Ein Kontextmodell für eine diensteorientierte Hochschule sollte also im besonderen die Eigenschaften des Ortes, der Identität und der Aktivitäten sowohl bei Nutzern als auch bei den angebotenen Diensten unterstützen.

Die Betrachtung der Nutzer und Anbieter zeigt, dass die persönliche Identität des Nutzers weniger relevant ist als das Wissen, welchen Gruppen der Nutzer angehört. So lassen sich, wie schon erwähnt, Bedürfnisse der Rolle des Nutzers besser zuordnen als der Identität. Die Gruppen lassen auch die Ermittlung der Nutzerkreise zu, für die der Nutzer Dienste anbieten könnte. Außerdem ist es mit Hilfe dieser Methode möglich, auch auf weitergehende Kontextinformationen zu verzichten und eine Dienstvermittlung ohne genaues Wissen über die Identität des Nutzers zu realisieren.

2.3 Crowdsourcing zur evolutionären Entwicklung kontextbewusster Systeme

Ein zentraler Aspekt, der zur Konzeption des CASA-Ansatzes führte, war die Überlegung, mit welchen Mechanismen sich (Weiter-)Entwicklung und Instandhaltung eines kontextbewussten Systems effektiv gestalten lassen. Systeme, die für einen speziellen Raum oder eine spezielle Installation entwickelt wurden, sind in ihrer Lebensfähigkeit an ihren Raum oder ihre Installation gebunden. Für ein langfristig nutzbares und nutzbringendes System bedarf es daher neben einer Trennung zwischen abstrakter Funktion und Realisierung auf der konzeptionellen Ebene auch einer breiten Basis an Komponenten und Installationen. Gleichzeitig unterliegen diese Komponenten den Entwicklungs- und Lebenszyklen der Umgebungen, für die sie entwickelt wurden.

Eine Methode, mit der die Anforderung einer breiten Entwicklung bei gleichzeitiger stetiger Aktualisierung umgesetzt werden kann, ist Crowdsourcing. Bei dieser Methode wird ein Problem an eine dezentrale Gemeinschaft gegeben, die sich aus eigenem Interesse mit der Lösung des Problems oder einzelner Teilprobleme befasst. Dabei können je nach Ausprägung die Ergebnisse unter den Mitgliedern der Gemeinschaft frei geteilt werden und eigene Weiterentwicklungen angestrebt werden. Auf das Potential von Crowdsouring für Pervasive Computing wird bereits in [Cook 12] hingewiesen.

> Crowd sourcing, or the act of outsourcing a task to the public, has the potential to revolutionize data collection and processing by enabling in-depth, large-scale, cost-effective information gathering as well as more accurate techniques for information extraction from data.
>
> (Diane J. Cook, Sajal K. Das [Cook 12])

Gleichzeitig werden die zu erwartenden Herausforderungen benannt. Diese umfassen hier die Schaffung eines Anreizes für die Nutzer, das Erkennen und Kompensieren von bösartigem Verhalten und die Sicherung der Privatsphäre. Alle drei hier genannten Aspekte werden in dieser Arbeit adressiert, wobei der Schwerpunkt auf dem letzten Aspekt liegt.

Als Analogie zur Problematik der kontinuierlichen Anpassung heterogener, verteilter Systeme an sich ändernde Umgebungen bieten sich die Erkenntnisse auf dem Gebiet der Evolutionstheorie an. Um diese Sicht näher zu beleuchten, werden in den folgenden Abschnitten zuerst die Charakteristika des Crowdsourcing beschrieben sowie Vor- und Nachteile der Modulentwicklung für kontextbewusste Systeme diskutiert. Um auch Aussagen über das Potential für kontinuierliche Entwicklungen treffen zu können, werden anschließend Analogien zwischen Crowdsourcing und der Evolutionstheorie dargestellt. Abschließend wird ausgeführt, welche Schlüsse sich daraus für diese Arbeit ableiten lassen.

2.3.1 Crowdsourcing

Der Begriff Crowdsourcing wurde erstmals im Juni 2006 von Jeff Howe in seinem Artikel „The Rise of Crowdsourcing“ benutzt [Howe 06]. Er beschreibt darin unter anderem die Arbeitsweise von iStockphoto[2], einer Webseite, auf der Nutzer die Nutzungsrechte an eigenen Bildern und Illustrationen verkaufen und damit hauptberuflichen Fotografen und Designern Konkurrenz machen können. Möglich wird dies durch die Verfügbarkeit preisgünstiger professioneller Kameras. Es handelt sich dabei nicht um reines Outsourcing, da die Aufgabe, hier die Erstellung von Illustrationen und Fotografien, nicht an ein spezifisches anderes Unternehmen oder einen Dienstleister weitergegeben wird. Howe definiert Crowdsourcing wie folgt:

> ”Crowdsourcing is the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call.”
>
> (Jeff Howe [Howe 10])

[2] https://www.istockphoto.com [Online letzter Zugriff 15.02.2018]

Weitere Beispiele dieser Methode sind Wikipedia[3] und Amazon Mechanical Turk[4]. Bei ersterer wird der Autor für seine Einträge in die freie Online-Enzyklopädie nicht finanziell entlohnt, sondern nur bei angemessener Qualität mit Reputation und Rechten bedacht. Mechanical Turk ist eine Internetplattform von Amazon, auf der Unternehmen kleinteilige Aufgaben, wie das Transkribieren von Gesprächsmitschnitten oder das Aussortieren von qualitativ minderwertigen Bildern aus Sammlungen ähnlicher Bilder, den Nutzern der Plattform offen anbieten. Die Entlohnung ist dabei ebenso kleinteilig und beginnt bei 0,01 $ pro Aufgabe.

Nach [Quinn 11] sind die verschiedenen Methoden, Nutzer zur Mitarbeit zu bewegen, folgende:

- Bezahlung, wie im Fall von Mechanical Turk,
- Altruismus, wie bei OpenStreetMap oder Wikipedia,
- Freude, z.B. durch Gamification,
- Reputation, wie im Fall von Wikipedia, oder
- Implizite oder auch versteckte Arbeit, wie bei reCAPTCHA [von Ahn 08]

In [Kaufmann 11] wurde bei einer Erhebung mit 431 Nutzern von Mechanical Turk festgestellt, dass die extrinsische Motivation der Bezahlung die Hauptmotivation darstellt. Darauf folgt jedoch die intrinsische Motivation des Auslebens der eigenen Kreativität oder des Anwendens der eigenen Fähigkeiten, die von den Befragten höher gewichtet werden als sonstige extrinsische Motivation, wie zum Beispiel das Aneignen neuer Kenntnisse.

Bei Nutzern, die Bezug zu den jeweiligen Domänen haben, lässt sich somit davon ausgehen, dass sie sich auch ohne eine Bezahlung motivieren lassen Aufgaben aus ihrer Domäne zu lösen. Als Beispiel sei hier auf die große Anzahl an Autoren der deutschsprachigen Wikipedia mit rechtswissenschaftlichem Hintergrund[5] verwiesen, die vornehmlich Themen ihres eigenen Fachgebiets bearbeiten.

Die Übertragung auf das Problem der Identifikation, Beschreibung und zuletzt auch auf die Implementierung von Diensten erlaubt daher den Schluss, dass der Nutzer einer Domäne sowohl geeignet als auch gewillt ist, eine eigene Domäne und für sie relevante Dienste zu beschreiben und gegebenenfalls auch zu implementieren. Die Motivation, die von Anfang an möglich wäre, ist einerseits Altruismus, denn die Dienste könnten auch anderen nutzen, und Eigeninteresse, denn der Nutzer schafft in erster Linie für sich selbst einen Mehrwert. Als weitere Motivation ist auch Gamification möglich, wie in der Evaluation in Kapitel 5 genauer beschrieben wird. Weitere Motivation, wie die Steigerung der eigenen Reputation innerhalb seiner sozialen Gruppe oder das Nutzen impliziter Arbeit, erfordern in erster Linie Erweiterungen auf der Präsentationsebene des Systems, und sind ebenso möglich. Als Beispiel für die implizite Arbeit sei hier die Möglichkeit genannt, dass Nutzer während des Logins in das System ein simples Tagging von neuen Diensten vornehmen könnten und dafür Punkte

[3] https://en.wikipedia.org [Online letzter Zugriff 15.02.2018]

[4] https://www.mturk.com [Online letzter Zugriff 15.02.2018]

[5] `https://de.wikipedia.org/wiki/Wikipedia:Wikipedianer/nach_Wissensgebieten/Rechtswissenschaft` [Online letzter Zugriff 15.02.2018]

bekommen könnten. Mit Bonuspunkten für das tägliche oder zumindest regelmäßige Erfüllen dieser Aufgabe könnte eine einfache Gamification-Methode umgesetzt werden.

Notwendig für die Nutzung des Crowdsourcing-Ansatzes sind offene Schnittstellen für das System sowie nutzerfreundliche Interaktionskonzepte. So sollte ein System geschaffen werden, in dem jeder Anwender die Beschreibungen und Systeme seiner Domäne verbessert und diese Ergebnisse nach Möglichkeit mit anderen teilt. Ein kritisches Problem an dieser Stelle ist, dass gerade in diesem Umfeld des Beschreibens und Identifizierens viele ähnliche Lösungen für gleiche Dienste denkbar sind. Die Gründe dafür liegen in den unterschiedlichen Perspektiven der Nutzer, ihrer Unabhängigkeit und Verteiltheit sowie der unterschiedlichen Aggregationen der Dienste in Domänen. Genau diese Gründe sind jedoch auch die Voraussetzung für eine erfolgreiche Entscheidungsfindung in Gruppen [Surowiecki 05].

2.3.2 Evolution als Vorlage zur Entwicklung von Systemen für sich ändernde Umgebungen

Bei der Betrachtung des Problems der parallelen, aber von einander unabhängigen Entwicklung ergab sich die Idee, die Anforderungen für diesen Aspekt des Systems in der Evolutionstheorie zu suchen. Die Theorie der Evolution durch natürliche Selektion wurde 1858 von Charles Darwin und Alfred Wallace erstmals dargestellt [Darwin 58]. In dem bekannten Hauptwerk von Darwin, „The Origin of Species“ von 1859, erfolgte die ausführliche Darstellung [Darwin 59]. Die Hauptidee der Theorie ist, dass jene Organismen sich schneller reproduzieren, die sich am besten an die Gegebenheiten ihrer Umwelt angepasst haben, als Organismen, die nicht so gut angepasst sind.

Die Analogie in der Software-Entwicklung ist die These, dass es eine Vielzahl von Programmiersprachen oder auch Softwaresysteme gibt, die sich in einem Entwicklungsprozess befinden, in dem sie der Umgebung, in der sie genutzt werden, angepasst werden. Diese Entwicklung ist im Gegensatz zur Evolution jedoch gesteuert und wird aktiv durch den Menschen vorangetrieben. Die primär wirkende Kraft ist dabei das Streben nach einem System, das die jeweils aktuellsten Anforderungen, die durch seine Nutzer gestellt werden, am besten erfüllt. Entwickler und Softwareunternehmen versuchen dabei möglichst effizient durch Weiterentwicklung existierender Systeme und Neuentwicklungen einen Markt zu befriedigen.

Im Rahmen dieser These muss jedoch betont werden, dass es nicht das Ziel dieses Ansatzes ist, Darwins Evolutionstheorie eins zu eins auf technologische Prozesse und Entwicklungen anzuwenden. Neben der bereits erwähnten, zielgerichteten Steuerung in der technischen Entwicklung gibt es hier auch Prozesse, die sich so nicht in der natürlichen Evolution wiederfinden. Dies ist zum einen die Rekombination existierender Komponenten verschiedener Systeme zu einem neuen System. Zum anderen ist dies die Innovation, die ohne die Ableitung von existierenden Systemen ein neues schaffen kann.

Jedoch gibt es durchaus Parallelen, die sich zwischen Technologie und Evolution zeigen. Der Fokus liegt darauf, dass innerhalb von biologischen Nischen sehr spezielle Organismen existieren, die hochgradig angepasst und dort vielen anderen Organismen überlegen sind. Vergleichbar ist dies mit technischen Systemen, die für einen sehr begrenzten Anwendungsfall konzipiert sind. Als Beispiel sei hier das Konzept des Vektorprozessors angeführt, der bei der

gleichzeitigen Verarbeitung gleichartiger Daten, wie sie bei Matrizenberechnungen vorkommen, einem klassischen x86-Prozessor überlegen ist, da letzterer seine Operationen sequentiell ausführt.

In diesem Fall sind die Spezialisten besser an die Umweltbedingungen angepasst als die Generalisten, welche zwar mit den Spezialisten konkurrieren, jedoch auch abseits der Nische des Spezialisten ihre Existenz sichern können. Ändern sich jedoch die Umweltbedingungen zu Ungunsten der Spezialisten, so wird es diesen schwerer fallen sich anzupassen und sie werden eher aussterben als die Generalisten [Ridley 03, Kapitel 23]. Als Beispiel seien hier die High-End-Grafik-Workstations von Silicon Graphics genannt, die bedingt durch die rasante Zunahme der Grafikfähigkeit von PCs in den 1990er Jahren „ausstarben“. Die Analogie ist die These, dass ein System, das für einen spezifischen Zweck in einem spezifischen Umfeld geschaffen wurde schwerer an grundlegend veränderte Anforderungen anzupassen ist, als ein System, das hier allgemeinere Funktionen bietet. Dabei ist zu beachten, dass sich aus Generalisten auch Spezialisten entwickeln können, wenn sich dadurch der Spezies ein ökologischer Vorteil bietet [Ridley 03, Kapitel 13].

Im Kontext dieser Arbeit lässt sich der Vergleich ziehen, dass durch die starke Heterogenität in intelligenten Umgebungen eine große Menge an Systemen (Spezialisten) konzipiert und entwickelt wurde, die speziell für ihre jeweiligen Domänen (Umgebungen) geschaffen und angepasst wurden, sich jedoch nur mit erheblichen Aufwand zur Verwendung in anderen Umgebungen umstellen lassen. Gleichzeitig gibt es einige Systeme, die in einer Vielzahl von Umgebungen existieren können, ohne an diese speziell angepasst zu sein. Aus dieser Parallele lässt sich der Schluss herleiten, dass die Bedingungen, die in der Natur Wandel und Anpassungen von Arten an sich verändernde Umgebungen begünstigen in angepasster Form auch im technischen Umfeld zur Definition von Systemeigenschaften dienen können. Diese sollten Wandel und Anpassung begünstigen und können so zur Entwicklung eines Systems beitragen, das auch langfristig in sich ändernden Umgebungen erfolgreich sein kann.

Es werden daher die folgenden Thesen abgeleitet, die ebenfalls als Anforderungen an das in dieser Arbeit entwickelte System dienen.

1. Das Nebeneinander von Systemen, die als Spezialisten und Generalisten innerhalb einer Umgebung angesehen werden können, ist ein normaler Zustand, denn es existiert durch den kontinuierlichen Wandel der technischen Umgebungen sowie der Nutzerbedürfnisse kein System, dass alle Anforderungen und Möglichkeiten alleine erfüllen und ausschöpfen kann.

   Daraus folgt, dass es sowohl als Nutzer als auch als Entwickler innerhalb einer pervasiven Umgebung sinnvoll ist, sowohl Spezialisten als auch Generalisten zu unterstützen, zu entwickeln und zu nutzen.

2. Je allgemeiner, offener und vielseitiger ein System ist, desto geringer ist der Aufwand es an eine neue Umgebung anzupassen oder die Umgebung in der es etabliert ist zu verändern.

   Um möglichst allgemein und offen zu sein, muss ein System in einer pervasiven Umgebung in der Lage sein flexibel durch verschiedene unabhängige Entwickler erweitert

zu werden und gleichzeitig eine Vielzahl an bereits existierenden Schnittstellen und Standards zu unterstützen.

3. Langfristig ist davon auszugehen, dass Änderungen und Weiterentwicklungen, die in einer Umgebung erfolgen, grundsätzlich auch in anderen Umgebungen erfolgen können.

   Während sich in der Natur mit der konvergenten Evolution bei verschiedenen Arten gleiche Merkmale herausbilden können, ist es im Kontext dieser Arbeit ein Ziel, eine Innovation, die in einer Umgebung entwickelt wurde, in andere Umgebungen zu übertragen. Dazu ist es notwendig, dass nicht nur die Beschreibungen und Dienste austauschbar und erweiterbar sind, sondern dass auch das Empfehlungssystem selbst modular und erweiterbar ist.

2.3.3 Anwendung der Erkenntnisse der Evolutionstheorie auf das Beschreiben von Diensten durch Crowdsourcing

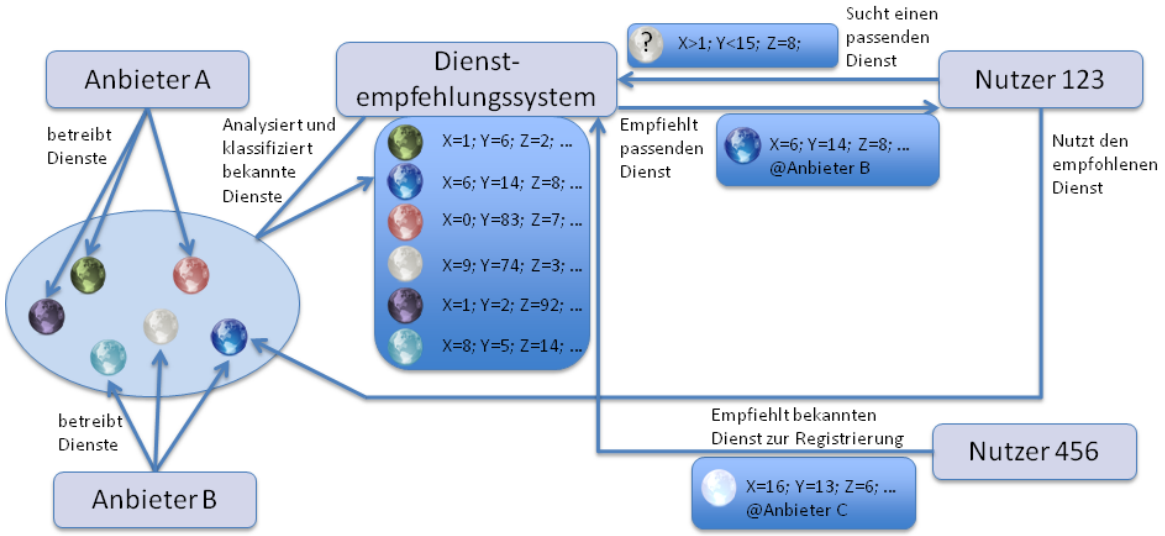

Abbildung 2.1: Konzeptuelle Architektur eines Dienstempfehlungssystems

Im folgenden Abschnitt werden die vorangegangenen Thesen auf den Anwendungsfall der Beschreibung von Diensten bezogen und es werden Schlussfolgerungen für das zu entwickelnde System gezogen. Zur besseren Verständlichkeit ist in Abb. 2.1 konzeptuell die Architektur eines Dienstempfehlungssystems dargestellt. Die Koordinaten stellen abstrakt die Eigenschaften der Dienste dar.

Das System verfügt dabei über eine bereits bekannte Menge von Diensten, die analysiert und klassifiziert wurden. Diese hier als Wertepaare dargestellte Beschreibung dient der Beantwortung von Empfehlungsanfragen von Nutzern. Dabei gibt der Nutzer bei einer Anfrage möglichst genau die Eigenschaften des Dienstes an, den er benötigt. In der Antwort des Systems sind dann die für den Nutzer notwendigen Informationen enthalten, um zu bewerten, ob der empfohlene Dienst den Anforderungen genügt und wo dieser angeboten wird. Will ein

Nutzer einen Dienst empfehlen, so muss er dem Dienstempfehlungssystem ebenfalls mitteilen wie der Dienst zu erreichen ist und welche Eigenschaften er hat.

Nebeneinander von Spezialisten und Generalisten

Für gleiche Domänen (Umgebungen) ist es sinnvoll, verschiedene Beschreibungen eines Dienstes sowie verschiedene Funktionsumfänge zu verwenden und zu unterstützen, da so je nach Nutzer und Situation eine für den Nutzer geeignetere Version des Dienstes angeboten werden kann, als wenn es nur genau eine Beschreibung mit einem Funktionsumfang gäbe.

Damit ein System eine Menge an Dienstbeschreibungen zur Verfügung stellen kann, die für verschiedene Nutzer und Situationen passend sind, ist es sinnvoll, Nutzer in den Prozess der Erstellung und Weiterentwicklung aktiv einzubeziehen. Somit können Beschreibungen aus unterschiedlichen Perspektiven erstellt werden, die wiederum denen der Nutzer entsprechen.

Ableitung der Spezialisten aus den Generalisten

Da sowohl Dienste als auch die Situationen, in denen sie sinnvoll verwendet werden können, stets noch weiter verfeinert beschrieben werden können, ist es von Vorteil, den Prozess der Beschreibung und das Format der Beschreibungen selbst offen und leicht zugänglich zu gestalten.

Mit der Beschreibung von Diensten und Situationen durch vordefinierte Eigenschaften ist es schneller möglich, bei einer Änderung eine Anpassung vorzunehmen. Wenn die Änderung hinreichend klein ist, kann sie auch durch einen Nutzer vorgenommen werden.

Weiterentwicklung durch Austausch

Auch wenn in verschiedenen Domänen eine Vielzahl von Beschreibungen entsteht, so sollte eine grundsätzliche Kompatibilität angestrebt werden, um langfristig die Austauschbarkeit und Erweiterbarkeit gewährleisten zu können.

Die Beschreibungen müssen in einem maschinenlesbaren Format, wie zum Beispiel XML, erstellt werden oder zumindest in ein solches überführt werden können. Gleichzeitig ist darauf zu achten, dass die Komponenten des Systems selbst möglichst offen und austauschbar erstellt werden, um eine Entwicklung nicht nur auf Ebene der Beschreibungen, sondern auch auf Ebene des Systems selbst zu gewährleisten.

2.3.4 Klassifikation der Entwicklungsprozesse und Typisierung der Entwickler

Betrachtet man diese Anforderungen an die Weiterentwicklung, so wird deutlich, dass die Weiterentwicklung auf verschiedenen Ebenen erfolgt. Um diese Entwicklung in einem späteren System sinnvoll zu unterstützen, muss zunächst überlegt werden, welche Entwicklungsprozesse sich wie unterstützen lassen und welche Nutzergruppen dabei angesprochen werden.

Konfiguration

Der einfachste Entwicklungsprozess lässt sich in dem Begriff der Konfiguration bündeln. Bei diesem Prozess wird eine existierende Beschreibung durch einen Nutzer oder Entwickler angepasst. Dies ist ein kleiner Schritt hin zu einer Spezialisierung und umfasst lediglich eine Änderung der Werte der Parameter, die bereits für den Dienst oder die Situation definiert wurden. Das Ergebnis kann dabei sowohl eine Ersetzung der alten durch die neue Beschreibung sein als auch eine alternative Beschreibung, die zusätzlich zu den existierenden zur Verfügung gestellt wird.

Eine Nutzerunterstützung lässt sich zum Beispiel durch Formularfelder in der Präsentation der Dienste oder Dienstempfehlungen realisieren und erfordert somit keine spezielle Fachkenntnis. Ein Nutzer würde so während der Nutzung des Dienstempfehlungssystems selbst die Möglichkeit haben das Ergebnis anzupassen und zu verbessern. Änderungen könnten hier zum Beispiel eine Aktualisierung einer Webadresse oder die Korrektur eines Fehlers in einem Text sein. Ob diese Änderungen direkt in das produktive System übernommen werden können oder zuerst geprüft werden müssen, hängt von dem Missbrauchspotential und dem Grad an Anonymität des Nutzers ab. Beim Missbrauchspotential ist dabei abzuwägen, wie schwerwiegend eine bewusst fehlerhafte Änderung im jeweiligen System ist und wie viele Nutzer davon betroffen wären. Eine Klassifikation der Dienste nach ihrer Kritikalität könnte sich hier neben der Anzahl der Nutzer, ihrer Anonymität und dem ihnen zugefügten Schaden auch an dem Schaden für das System selbst bemessen.

So ist eine Änderung des Dienstes, der Studenten über die Prüfungstermine informiert und ihnen relevante Hinweise geben soll, kritischer zu betrachten als eine Änderung an einem Dienst, der in einem Speziallabor eine Videoeinführung in die Benutzung der Klimaanlage gibt. Die missbräuchliche Änderung am Prüfungsinformationsdienst kann falsche Informationen an eine potentiell große Menge von nicht identifizierbaren Personen geben. Eine Richtigstellung erreicht eventuell nicht mehr alle Nutzer rechtzeitig und kann zu rechtlicher Unsicherheit bei der Gültigkeit der Prüfungsergebnisse führen. Bei dem Einführungsdienst in das Speziallabor wäre davon auszugehen, dass zum die Empfänger einer Falschinformation leicht zu identifizieren sind, und dass zum anderen der potentieller Schaden durch falsche Benutzung geringer ist. Gleichzeitig muss bedacht werden, dass ein System, das durch zu häufigen Missbrauch auffällt, aktive Nutzer verliert und damit auch jene Nutzer, die Fehler korrigieren würden.

Weiterentwicklung

Unter dem Begriff Weiterentwicklung lässt sich der zweite Entwicklungsprozess zusammenfassen. Dabei werden neben dem Inhalt der Beschreibungen auch die Strukturen und die Zuordnungen von Diensten und Situationen verändert. Die Änderungen umfassen damit alles, was sich ohne einen Eingriff in die Software des Dienstempfehlungssystems realisieren lässt. Das Ergebnis kann ebenso die Definition neuer Dienste und Situationen sein wie vollständig neue Verknüpfungen zwischen diesen.

Um diese Möglichkeit der Weiterentwicklungen einer Software zu realisieren, muss sie streng modular aufgebaut und gut konfigurierbar sein. Für diesen Prozess ist Fachkenntnis zwingend

nötig, wobei hier der Fokus auf dem Wissen über die Umgebung und die Ziele der Nutzer liegt und nicht auf einer bestimmten Programmiersprache oder -umgebung. Ein potentieller Entwickler in diesem Prozess ist der Administrator einer Umgebung, der durch das Installieren von Plugins und Erweiterungen sowie durch das Strukturieren und Konfigurieren der Dienste und Dienstempfehlungen agiert.

Innovation

Die Innovation setzt in der Software des Dienstempfehlungssystems selbst an. Dies umfasst neben der Software zur Darstellung der Dienstempfehlungen und der Verarbeitung der Kontextinformationen auch die Systeme zur Erhebung der Kontextinformationen und das Kontextmodell selbst. So ist es in jedem System entscheidend wie die einzelnen Fakten gespeichert und aufbereitet werden, um bei einer Anfrage ein Ergebnis mit einer möglichst hohen Empfehlungsgüte zu erzeugen. Wird der Empfehlungsalgorithmus angepasst, so kann dies meist nicht ohne eine Änderung der Sensoren und des Kontextmodells erfolgen. Dies gilt auch für eine Erweiterung der Anfrageparameter, die genutzt werden, um aus der Menge der Dienste möglichst jene auszuwählen, die einem Nutzer eine Mehrwert bieten.

Neben der Innovation in einem einzelnen Teil der Software kann eine Innovation auch die Ableitung eines neuen System für einen neuen Anwendungszweck darstellen, zum Beispiel die Entwicklung eines Dienstempfehlungssystems für ein pervasives Krankenhaus. Dabei könnten Teile aus dem in dieser Arbeit entwickelten System für eine pervasive Universität übernommen werden. Für das Krankenhaus müssten jedoch gleichzeitig diverse Strukturen und Module vollständig neu geschaffen werden, um zum Beispiel höheren Anforderungen an Datenschutz oder Ausfallsicherheit gerecht zu werden. Neben der Neukonzeption und Neuimplementierung bedeutet dies ggf. auch, dass viele der bereits existierenden Strukturen und Module angepasst werden müssten, um diese Querschnittsthemen zu berücksichtigen. Auch müssten neue Kontexte berücksichtig werden. Die Ergebnisse dieser Innovation sollten soweit wie möglich kompatibel mit dem ursprünglichen System gehalten werden, um sowohl durch die Veröffentlichung von Plugins die Entwicklung der ursprünglichen Umgebungen voranzubringen als auch um von anderen Innovationen zu profitieren, die auf Basis der Ursysteme entstehen.

Die Unterstützung dieser Form der Innovation innerhalb eines Systems benötigt eine grundlegende Offenheit der Quellen des Systems ebenso wie eine aktuelle Dokumentation und rege Kommunikation zwischen den Entwicklern. Fachwissen in der Softwareentwicklung ist hier zwingend notwendig, da Änderungen auf dieser Ebene alle anderen Ebenen nachhaltig beeinflussen und zu Störungen führen können. Potentielle Entwickler in diesem Bereich wären Softwareentwickler oder auch Teams, die Mitglieder in den verschiedenen Rollen der Softwareentwicklung enthalten(z.B. Entwickler, Tester, Architekten, usw.).

2.4 Pervasivität von Daten und Computern für intelligente Umgebungen

2.4.1 Systemarchitekturen für verteilte Umgebungen

Im Folgenden werden existierende Architekturstile für Anwendungen und Systemarchitekturen für verteilte Umgebungen vorgestellt und unter dem Aspekt der Eignung für intelligente Umgebungen diskutiert. Anschließend werden die Anforderungen für eine intelligente Umgebung innerhalb einer Hochschule hergeleitet und mit den existierenden Architekturen verglichen.

Software-Architektur

Vereinfacht kann unter einer Software-Architektur die logische Anordnung der Software-Komponenten eines Systems verstanden werden [Tanenbaum 08]. Folgende Definition leitet sich dabei aus [Bass 03] ab.

> The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components and the relationships among them.
>
> (Len Bass, Paul Clements, Rick Kazman, Linda Northrop [Bass 03])

Die „von außen sichtbaren“ Eigenschaften beziehen sich dabei auf jene Charakteristika, die externe Komponenten über das System annehmen können. Die Software-Architektur umfasst damit die Aufteilung der fachlichen Funktionalität auf Komponenten. Diese Komponenten stellen dabei die austauschbaren und wiederverwendbaren Einheiten dar, die mit ihren definierten Schnittstellen nach außen eine Software prägen.

Systemarchitektur

Je nach Blickwinkel kann unter der Systemarchitektur die Abbildung einer konkreten Software-Architektur auf eine bestimmte Hardware verstanden werden oder aber auch die Anordnung unterschiedlicher Komponenten eines Software-Systems und deren Organisation. Da diese Arbeit sich nicht auf eine bestimmte Hardware bezieht, kann dieser Aspekt hier unbetrachtet bleiben und es kann sich auf die konkrete Instanzierung der Software-Architektur fokussiert werde. Systemarchitekturen lassen sich nach [Tanenbaum 08] in drei Hauptarten unterteilen: die zentralisierten, die dezentralisierten und die Hybridarchitekturen. Bei den zentralisierten Architekturen existiert eine Komponente, die als zentraler Anlaufpunkt für viele andere Komponenten agiert und ohne die die anderen Komponenten ihren Aufgaben nicht nachkommen können. Bei den verteilten Architekturen existieren mehrere Komponenten (Peers), die jeweils über einen Teil der Datenmenge verfügen. Sind sie dabei untereinander gleichwertig und gleichberechtigt, ermöglichen sie es, dass die Ressourcen, die den einzelnen Peers zur Verfügung stehen, über das gesamte System hinweg gleichmäßig ausgelastet werden

und diese Systeme somit gut für die Verbreitung von Informationen geeignet sind (z.B. durch File Sharing). Hybridarchitekturen vereinen dabei Eigenschaften aus beiden vorangegangenen Architekturen, indem sie sie vermengen und unter den gleichwertigen Komponenten einige hervorheben und diesen so eine zentrale Rolle zuteilen (Superpeers).

Für verteilte Umgebungen gibt es im Wesentlichen vier Realisierungen der Systemarchitektur, die bereits in [Lehsten 08] vorgestellt wurden und hier mit ihren Vor- und Nachteilen wiedergegeben werden. :

Client-Server-Architektur
: Diese zentralisierte Architektur besteht zum einen aus dem Client, der seine Anfragen an einen ihm bekannten Server stellt, und zum anderen aus dem Server, der für die Bearbeitung von Anfragen verschiedener Clients zuständig ist. Der Vorteil dieser Variante ist, dass die Last zum großen Teil auf der Seite des Servers liegt und somit die Clients deutlich kleiner dimensioniert werden können. Dagegen existieren mehrere Nachteile. So ist es für den Client zwingend notwendig zu wissen, wie der Server zu erreichen ist. Dies schließt die Adresse, aber auch das Nachrichtenformat und das Kommunikationsprotokoll mit ein. Des Weiteren skaliert dieser Ansatz nicht, da bei einer wachsenden Zahl an Anfragen die Kommunikation und Last beim Server schnell steigt und dieser so zum Flaschenhals in der Kommunikation wird. Dies ist auch bei der Ausfalltoleranz ein Problem, da bei einem Ausfall des Servers oder seiner Kommunikationsverbindung alle Clients auf diese Funktionalität verzichten müssen. Abschließend geht die Kommunikation hier stets vom Client aus, was für einzelne Anfragen (Pulling) vorteilhaft ist, jedoch bei der regelmäßigen Abfrage eines Zustandes (Polling) zu unnötiger Kommunikation führt. Diese Architektur eignet sich daher nur sehr bedingt für dynamische Umgebungen mit sich ändernden Kommunikationspartnern oder Kommunikationsparametern.

Publish-Subscriber-Architektur
: Auch bei dieser ebenfalls zentralisierten Architektur existieren zwei Rollen, wobei sich die einzelnen Subscriber beim Publisher registrieren, um informiert zu werden, wenn neue Informationen verfügbar sind. Ein Publisher kann dabei viele Subscriber bedienen, da er neue Informationen nur einmal ermitteln und dann zum Beispiel über Multicast veröffentlichen kann. Nachteilig ist jedoch, dass auch hier der Subscriber über das Wissen verfügen muss, wie und wo der Publisher zu erreichen ist. Der Publisher stellt in dieser Architektur den Flaschenhals dar und ist außerdem für die Ausfalltoleranz eine kritische Komponente, da er gleichzeitig mehrere Subscriber versorgen soll. Für dynamische Umgebungen kann sich dieses Modell eignen, wenn das Problem des Auffindens des Publishers dynamisch gelöst wird und die Informationen für mehrere Verbraucher von Relevanz sind.

Peer-to-Peer-Architektur
: Durch die Vereinigung von Client und Server in einem Knoten und die gleichberechtigte Kommunikation unter mehreren verschiedenen Knoten wird das Peer-to-Peer-Konzept als Vertreter der verteilten Architekturen umgesetzt. Dabei kann jeder Knoten Anfragen an seine Nachbarknoten stellen, die diese an ihre Nachbarknoten weiterleiten. Die Antworten gelangen dabei auf dem gleichen Weg zurück an den Ausgangspunkt. Auf diese Weise müssen Knoten nur Wissen über ihre Nachbarknoten vorhalten und können von einem viel größeren Netzwerk profitieren, während der Bedarf an notwendigem Wissen

zur Kommunikation in den einzelnen Knoten geringer ist als in anderen Architekturen. Durch Redundanz der Kommunikationswege und Knotenfunktionalitäten kann auch eine höhere Ausfallsicherheit und Skalierbarkeit gewährleistet werden. Nachteilig wirkt sich die fehlende zentrale Steuerung und Überwachung aus, die den Wartungsaufwand erhöht. Des Weiteren erfordert diese Architektur eine intensive Vernetzung der jeweiligen Knoten, die nicht unbedingt gegeben ist, wenn sie in verschiedenen Netzen sind. Für intelligente Umgebungen eignet sich diese Architektur, sobald eine Vielzahl von Geräten vorliegt, die sich im gleichen Netz austauschen und gegenseitige Funktionalitäten nutzen können.

Broker-Architektur
: Bei diesem Ansatz wird zwischen dem Anbieter (Provider) und den Konsumenten (Consumer) einer Information oder eines Dienstes ein Vermittler (Broker) geschaltet. Dieser nimmt die Dienstbeschreibung sowie die Kontaktinformationen eines Providers entgegen (publish) und stellt sie auf Anfrage (find) dem Consumer zur Verfügung. Abschließend nimmt der Consumer selbstständig Kontakt (bind) mit dem Dienst des Providers auf, um diesen zu nutzen. Wie beim vorhergehenden Konzept muss hier nur eine Instanz, der Broker, bekannt sein. Dies gilt sowohl für den Consumer als auch für den Provider. Bei entsprechender Redundanz kann ein Consumer aus mehreren verfügbaren Dienst-Providern auswählen und so die Ausfallsicherheit erhöhen. Neue Dienstanbieter und Dienstnutzer brauchen ebenfalls nur mit dem Broker zu kommunizieren und können so auch zur Laufzeit die verfügbaren Funktionalitäten des Netzwerks erweitern. Für intelligente Umgebungen eignet sich dieser Ansatz besonders, da er die Dynamik des Systems unterstützt und eine gute Ausfalltoleranz besitzt. Des Weiteren lassen sich auch komplexere Dienste aus Verkettungen von einfacheren Diensten erstellen und transparent als neue Dienste anbieten.

Neben den vorgestellten Architekturen sind auch Hybride möglich, die die jeweiligen Vor- und Nachteile vereinen.

2.4.2 Architekturstile für Anwendungen in verteilten Umgebungen

Architekturstil

Getrennt von der Software-Architektur lässt sich der Architekturstil betrachten. Während die Architektur die Ordnung und Struktur des Systems und seiner Komponenten beschreibt, umfasst der Architekturstil die Leitlinie für die Gestaltung der Architektur. Dies umfasst z.B. wie die Komponenten miteinander verbunden sind, wie sie Daten austauschen und wie sie zu einem System konfiguriert werden [Tanenbaum 08]. Zur Verdeutlichung lässt sich hier eine Parallele zur baulichen Architektur ziehen. Die Architektur einer Kirche beschreibt dabei ihren Aufbau mit Elementen wie dem Mittelschiff und den Seitenschiffen, während der Architekturstil festlegt, ob romanische, gotische oder barocke Elemente verwendet werden.

Auch in der Informatik existieren für Anwendungen verschiedene Architekturstile, wobei die wichtigsten vier nach [Tanenbaum 08] hier im Folgenden erläutert und auf ihre Vor- und Nachteile für Anwendungen in intelligenten Umgebungen beurteilt werden.

Geschichtete Architekturen
: Bei diesem Stil werden die einzelnen Softwarekomponenten in Schichten angeordnet und eine Kommunikation erfolgt stets zwischen den angrenzenden Schichten. Der Vorteil dabei ist, dass die Kommunikationswege statisch sind und die Komponenten somit zur Laufzeit nicht auf Veränderungen dynamisch reagieren müssen. Dies stellt zwar für die Geschwindigkeit einen Vorteil dar, schließt jedoch das Einfügen von Komponenten und Schichten zur Laufzeit aus, was den Stil für komplexe Anwendungen in intelligenten Umgebungen ungeeignet macht.

Objektbasierte Architekturen
: Durch die Objektorientierung kann auf Schichten verzichtet werden und mittels Prozedurenaufrufen direkt zwischen den Objekten kommuniziert werden. Dies erlaubt eine höhere Dynamik, da hier ein Nebeneinander von dynamischen und statischen Prozedurenaufrufen möglich ist. Bei den statischen Aufrufen müssen die Schnittstellen für die Objekte bereits zur Entwurfzeit definiert sein, um zur Laufzeit genutzt werden zu können. Die dynamischen Aufrufe erlauben auch das Aufrufen von Prozeduren in Objekten, die zur Entwurfszeit unbekannt waren.

Datenzentrierte Architekturen
: Kommunizieren die Prozesse über einen gemeinsamen Datenraum (Repository), wird von datenzentrierten Architekturen gesprochen. Der Vorteil hier ist, dass nicht mehr die Schnittstellen zu den Prozessen standardisiert sein müssen, sondern lediglich die Austauschformate für die Kommunikation. Da dem Repository hier eine zentrale Rolle beim Austausch zwischen den Komponenten zukommt, ist die Datenhaltung ein wichtiger Aspekt. So müssen hier bei einer Vielzahl gleichzeitiger Zugriffe in intelligenten Umgebunden geeignete Konsistenzstrategien Anwendung finden.

Ereignisbasierte Architekturen
: Werden die Komponenten über einen gemeinsamen Kommunikationskanal (engl. Bus) verbunden und kommunizieren darüber in Form von Nachrichten, kann von einer ereignisbasierten Architektur gesprochen werden. Dieser Stil findet Anwendung in allen Systemen, die eine nachrichtenorientierte Middleware nutzen und so eine entkoppelte dynamische Kommunikation realisieren. Für komplexe Anwendungen ist dieser Stil gut geeignet, jedoch ist die Auswahl der richtigen Middleware hier ein entscheidendes Kriterium, da es eine Vielzahl an zueinander inkompatiblen Komponenten und Bus-Systemen gibt.

2.4.3 Kernanforderungen für intelligente Umgebungen an Hochschulen

Betrachtet man intelligente Umgebungen genauer, so stellt man fest, dass es Unterschiede in den Anforderungen entsprechend des Einsatzszenarios gibt. Das im Rahmen dieser Arbeit gewählte Szenario einer Hochschule stellt entsprechend seiner Nutzerstruktur sowie deren Nutzungsverhalten keine Ausnahme dar.

Unterstützung heterogener Geräte, Anwendungen und Netzinfrastrukturen

Gerade an Hochschulen zeigt sich durch die hohe Zahl an jungen und technikaffinen Nutzern eine hohe Fluktuation an Geräten. Die Netzinfrastrukturen stehen häufig nicht unter einer zentralen Verwaltung, sondern obliegen teilweise den Instituten, den Fakultäten oder den Rechenzentren. Des Weiteren existieren im Bereich der Verwaltung auch viele Geräte und Anwendungen, die als Altsysteme betrachtet werden können und somit ihrerseits zur Heterogenität beitragen. Daher sollte eine intelligente Umgebung hier plattformunabhängig konzipiert sein und sich möglichst dynamisch auch in Altsysteme integrieren lassen.

Skalierbarkeit

Da so ein System einer steten Entwicklung unterworfen ist, sind die Erweiterbarkeit und die Skalierbarkeit ebenfalls entscheidende Faktoren. In den verschiedenen Systemen einer Hochschule gibt es entsprechende Hochzeiten der Belastung. So sind z.B. Systeme zur Einschreibung in Veranstaltungen gerade am Semesteranfang besonders frequentiert. Wird hier eine Assistenzfunktion durch eine intelligente Umgebung eingebunden, so muss das berücksichtigt werden.

Nutzerfreundlichkeit

Intelligente Systeme, die ihren Nutzern Assistenz anbieten, müssen so konfiguriert sein, dass die Nutzer sie auch bedienen und nutzen können. Daher ist die Nutzerfreundlichkeit eine der wichtigsten Anforderungen an ein solches System. Nur wenn der Zugang zum System als leicht und die Qualität der Assistenz als hochwertig empfunden wird, ist davon auszugehen, dass das System aktiv genutzt wird. Hier ist zu beachten, dass der Zweck ja gerade die Vereinfachung bereits existierender Prozesse und Arbeitsabläufe ist und nicht die Einführung von neuen. Ebenfalls muss darauf geachtet werden, die Hürden für die Nutzer so gering wie möglich zu halten und einen Zugriff von möglichst vielen Endgeräten aus zu ermöglichen. Abschließend sollte auch der Dialog mit dem Nutzer gefordert werden, da ein solches System darauf angewiesen ist, sich stets den Entwicklungen der Nutzer anzupassen.

2.5 Systematik verwandter Arbeiten

Durch die Vielzahl an relevanten Forschungsgebieten ergibt sich für Teilbereiche dieser Arbeit ein großes Feld verwandter Arbeiten und Projekte, jedoch nur eine geringe Anzahl an Projekten und Publikationen, die sich mit dieser Arbeit als ganzes in Relation setzen lässt. Um nicht eine weitere Übersicht zu (Middleware-)Systemen für den kontextbewussten Zugriff auf Dienste zu geben, sei dies hier ausgespart und auf die Übersichten [Baldauf 07, Kjær 07, Knappmeyer 13, Bellavista 13] verwiesen. Im weiteren Verlauf des Abschnitts werden nur die wichtigsten Kernaspekte dargestellt.

Strukturell besteht ein kontextbewusstes System aus drei Abschnitten: der Erfassung und Aggregation von (Kontext-)Daten, der Verarbeitung dieser Daten mit der Erstellung eines Kontextmodells, das die Situation abbildet, und der abschließenden Nutzung dieses Modells, um Entscheidungen in einem Anwendungsfall zu treffen. Die folgenden Abschnitte orientieren sich an diesem Aufbau. Dazu wird in Abschnitt 2.5.1 auf Systeme zur Sammlung und Aggregation von Daten aus verteilten Quellen eingegangen. Anschließend werden im Abschnitt 2.5.2 drei Ansätze zur Kontextmodellierung vorgestellt und diskutiert, welche Vorteile diese haben und wo weitere Herausforderungen bestehen. Da die Nutzung von Kontextdaten auf verschiedene Arten erfolgen kann, musste ein Anwendungsfall ausgewählt werden, der der Thematik der Arbeit mit der Integration und Dissemination von Diensten nahe kommt. Daher wird in Abschnitt 2.5.3 erläutert, welche Ansätze es im Bereich der Empfehlungssysteme gibt und welche Methoden sich für den vorliegenden Fall anwenden lassen. Abschließend wird in Abschnitt 2.5.4 dargestellt, welche anderen Arbeiten sich bereits mit der Problematik großflächiger pervasiver Systeme befasst haben und welche Herausforderungen und Lösungen dabei benannt wurden.

2.5.1 Systeme zur Aggregation von Daten und Erfassung von Situationsbeschreibungen

Um Situationsbeschreibungen zu erstellen, ist es notwendig Daten aus verschiedenen Quellen zu sammeln und zu bündeln. Das Bündeln verschiedener Informationsströme zu einem neuen Informationsstrom oder auch die Bündelung von verschiedenen Webanwendungen zu einer neuen Webanwendung wird als Mash-Up bezeichnet. Dieser Prozess der Aggregation bezieht sich im vorliegenden Szenario neben den Daten aus den internen Informationsquellen auch auf Daten aus externen Informationsquellen, die zum Beispiel als RSS-Feed zur Verfügung stehen oder auch direkt aus Web-Anwendungen abgefragt werden müssen. Daher wird im folgenden Abschnitt auf zwei Ansätze eingegangen, die sich zu diesem Zweck im Internet bereits etabliert haben und im weiteren Abschnitt auf einen Ansatz, der sehr aktuell und besonders aus Sicht des Datenschutzes relevant ist.

Aggregation und Filtern von verschiedenen Nachrichtenströmen

Um die Recherche über verschiedene Nachrichtenströme hinweg zu ermöglichen, veröffentlichte Yahoo im Jahr 2009 Yahoo Pipes. Dabei handelte es sich um eine durch den Nutzer konfigurierbare Web-Anwendung, die es ermöglicht, Nachrichtenströme aus dem Netz zu aggregieren und als einen neuen Strom zur Verfügung zu stellen. Die Konfiguration wurde dabei durch einen graphischen Editor für den Nutzer visualisiert und vereinfacht. Als Quellen konnten neben CSV-Dateien, wie sie häufig im Open-Data-Umfeld vorkommen, auch die weit verbreiteten Formate wie RSS-Feeds oder XML dienen. Nach der Festlegung der Quellen wurde durch den Nutzer ein Netzwerk mit verschiedenen Modulen erstellt, die für die Bearbeitung der Daten durch Filter und Manipulatoren sorgten. Abschließend wurde ein Output definiert, der wieder in Form von RSS, Atom oder JSON vorliegen konnte. So ließ sich ein derart erstellter Datenstrom durch einen Nutzer nun entweder mit einem Browser oder Feed Reader abonnieren oder durch die Pipe Badges in eine Webseite einbinden. Dies war dadurch

möglich, dass alle Pipes bei Aufruf der jeweiligen URL ungeschützt abrufbar waren. Yahoo stellte den Dienst zum 30. August 2015 ein [Gatti 15].

Verknüpfung von verschiedenen Web-Anwendungen

Um nicht nur Nachrichtenströme in vordefinierten Formaten verarbeiten zu können, sondern direkt auf Ereignisse in Web-Anwendungen reagieren zu können, existiert seit Dezember 2010 der Dienst IFTTT[6], was für „If This Than That“ steht. Das Ziel dieser ebenfalls kommerziellen Web-Anwendung ist es ein Ereignis, wie das Eintreffen einer Nachricht, in einer Web-Anwendung mit dem Auslösen einer Aktion in einer anderen Webanwendung zu koppeln. Ein in diesem Punkt vergleichbarer Dienst ist auch zapier.com. Bei IFTTT können die Web-Anwendungen zusätzlich auch einen Bezug zu Geräten des Nutzers haben. So ist es hier möglich, die Positionsdaten des Mobilgerätes des Nutzers auszuwerten, um die Lautstärke des Gerätes nach dem Verlassen des Arbeitsplatzes zu erhöhen. Die Web-Anwendungen werden dabei als Channels bezeichnet und können meist sowohl als Quelle eines Ereignisses als auch als Ziel einer Aktion genutzt werden. Neben den bereits genannten Web-Anwendungen und Android-Schnittstellen unterstützt IFTTT auch mehrere Schnittstellen von Heimautomatisierungssystememem wie das Belkin WeMo oder die Philips Hue Leuchte. Auch bei IFTTT ist der Nutzer abhängig von einem proprietären System, das nur eine begrenzte Auswahl an Diensten zur Verfügung stellt und ausschließlich durch seine eigenen Entwickler erweitert wird. Dies hat innerhalb der Nutzergemeinde bereits dazu geführt, dass Nutzer Kanäle zweckentfremdet haben, um ihre Ziele zu verwirklichen. Diese Methode ist nicht langfristig sinnvoll, da hier durch den IFTTT-Entwickler jederzeit Änderungen vorgenommen werden können, die diese Anpassung unbrauchbar machen. Des Weiteren ist problematisch, dass IFTTT die Nutzung der Web-Anwendungen seiner Nutzer nachverfolgen kann. Dies ist besonders kritisch, da es sich um ein Unternehmen handelt, das aktuell kein Entgeld von seinen Nutzern nimmt und somit potentiell die Nutzungsdaten und Nutzerprofile an andere Dienste weiterverkaufen kann.

Nutzergetriebene Verknüpfung von Ereignissen und Aktionen auch für schützenswerte Informationen und Daten

Neben den genannten kommerziellen Ansätzen gibt es auch Entwicklungen, die von Entwicklern frei zur Verfügung gestellt werden und auf eigener Hardware ausgeführt werden können. So wurde mit dem Huginn-Projekt [Cantino 13, Denham 13] ein System vorgestellt, das von den Entwicklern als Mischung aus IFTTT und Yahoo Pipes beschrieben wird. Es handelt sich dabei um eine Web-Anwendung, die es einem Nutzer erlaubt über eine einfache graphische Oberfläche eigene Agenten zu erstellen, die ähnlich zu IFTTT Aktionen ausführen, wenn vorbestimmte Situationen eintreten. Die Entwicklung erfolgt durch Beteiligung der Öffentlichkeit auf der Plattform GitHub. So sind hier auch bereits eine Vielzahl von Schnittstellen implementiert worden, wobei FTP, IMAP, RSS und Twitter nur eine kleine Auswahl darstellen. Der große Vorteil liegt in der öffentlichen Zugänglichkeit der Weiterentwicklung, da dies

[6] `https://ifttt.com` [Online letzter Zugriff 15.02.2018]

bei entsprechender Verbreitung die zeitnahe Korrektur von Fehlern und stete Erweiterung verspricht. Außerdem sind durch die Möglichkeit, die Software auf dem eigenen Rechner einzusetzen und für die eigenen Zwecke zu erweitern, die Einsatzszenarien nicht auf die durch die Entwickler vorgesehenen begrenzt. Abschließend stellt auch die Veröffentlichung unter der MIT-Lizenz[Open Source Initiative 17] einen Vorteil dar, da diese keinen Zwang zur Weitergabe von Weiterentwicklungen enthält und darüber hinaus die Möglichkeit einräumt, dass abgeleitete Werke unter einer anderen Lizenz veröffentlicht werden können [Rosen 04].

2.5.2 Umsetzungen mit verschiedenen Kontextontologien

Im folgenden Abschnitt soll gezeigt werden, welche Lehren sich aus vorangegangenen, verwandten Projekten ziehen lassen und wo noch Fehlstellen existieren, die sich durch diese Arbeit adressieren lassen. Dazu werden drei Projekte besprochen, die chronologisch aufeinander folgten.

SOUPA und CoBrA

Bei SOUPA, der Standard Ontology for Ubiquitous and Pervasive Enviroments [Chen 04a], handelt es sich um ein Ergebnis der UbiComp Special Interest Group aus dem Jahr 2004. Dabei ist SOUPA nicht eine konkrete, sondern eine modulare Ontologie, die sich aus verschiedenen Teilen zusammensetzt. Hier wird zwischen Core-Modulen, wie Person oder Space, und Extension-Modulen, wie Meeting oder Document, unterschieden. Die Konzepte der anderen Ontologien werden nicht direkt importiert, sondern abgeleitet, um unnötige Eigenschaften auszublenden. Eine grundlegende Kompatibilität wird gewährleistet, indem über eine Mapping-Funktion eine Abbildung auf die Originalklassen ermöglicht wird. Unter anderem werden Konzepte wie Person, Ereignis, Zeit und Intention unterstützt. Darüber hinaus gibt es Richtlinien (Policies) für die Gewährleistung von Sicherheit und Privatsphäre. Als Repräsentation und Sprache kann OWL und damit XML verwendet werden.

Als praktische Umsetzung von SOUPA wurde 2004 ebenfalls von Harry Chen CoBrA, die Context Broker Architecture, publiziert [Chen 04b]. Dabei handelt es sich um eine Systemarchitektur, die SOUPA mit OWL verwendet. Zentrales Element ist der Context-Broker, der als Agent das vorhandene Wissen verwaltet. Architektonisch sind hier auch mehrere Broker möglich, die sich austauschen können. Entsprechend der Idee, die Ontologie modular zusammenzusetzen, verwendet CoBrA eine eigene Ontologiemenge (COBRA-ONT) zur Erfassung und Verteilung des Wissens unter Verwendung existierender Ontologien. Im Context-Broker sitzt die CORE, die Context Reasoning Engine, die zur Schlussfolgerung (Reasoning), Interpretation, Nutzung und Detektion beziehungsweise Korrektur inkonsistenter Informationen verwendet wird. Das Modul zum Schutz der Privatsphäre ist hier umgesetzt als eine Funktionalität, die es dem Nutzer erlaubt, Regeln für seine Daten zu formulieren.

Das Unterstützen einer Verteilung der Ontologie kann als eine Kernanforderung für das kontextbewusste System gesehen werden. Einerseits müssen sich alle Module, die miteinander kommunizieren, auf ein für einander verständliches Wissensformat einigen, andererseits wird

durch Komponenten, die Wissen aus verschiedenen Domänen nutzen, schnell eine sehr komplexe Ontologiemenge entstehen. Die Verwendung von OWL ist für derart komplexe, im Vorfeld strukturierte Anwendungen geeignet, jedoch sind hier bei dynamischen Systemen, die eventuell auch resourcenbeschränkt sind, Probleme durch das Nachladen neuer Ontologien, wie auch bei der Verarbeitung der Ontologie selbst, zu erwarten. Außerdem erfordert die Formulierung der Regeln zur Privatsphäre von dem Nutzer eine separate Aktion. Ein „by-Design “ integrierter Schutz der Privatsphäre wäre hier nützlicher.

COMANTO

Im Jahr 2006 wurde COMANTO, die context management ontology, als Teil des DAIDALOS-Projektes vorgestellt [Roussaki 06]. Es handelt sich dabei um die dort verwendete Standardontologie. COMANTO beschreibt nur die allgemeinen Kontexttypen und Beziehungen in Form einer übergeordneten Ontologie und versucht so Domänen-, Anwendungs- und Situationsunabhängigkeit zu wahren. Dieser Ansatz findet sich auch in [Wang 04]. Dabei ist das Hauptziel der Austausch und die Synchronisation zwischen den verschiedenen Nutzern. Für die Realisierung eines konkreten Projektes sind somit jedoch stets Erweiterungen notwendig, die sich von den vorgegebenen Klassen und Relationen ableiten lassen. So existiert ein zentrales Wurzelobjekt, das SCE (Semantic Context Entity), von dem Knoten wie Person, Place, Preference und Activity erben. Die Umsetzung erfolgt in OWL und erlaubt so auch eine Verknüpfung mit anderen Ontologien.

Bei diesem Projekt ist der Ansatz einen Kern an Konzepten vorzugeben und es den konkreten Umsetzungen, somit den Entwicklern zu überlassen, die Ontologie auf das Szenario anzupassen. So ist eine generelle Kompatibilität gegeben. Jedoch fehlen in diesem Ansatz Konzepte zur Umsetzung von Datenschutz und Privatsphäre.

SOCOM

Das letzte Projekt, das in dieser Reihe betrachtet werden soll, ist SOCOM, das multi-Sensor Oriented Context Model [Tan 12], aus dem Jahr 2012. Die Betrachtung in diesem Modell konzentriert sich auf die Sensoren, da sie die Komponenten sind, die die Kantextdaten liefern. Unter einem Sensor wird hier ein „general contextual information provider“ verstanden, der sowohl physisch als auch logisch existieren kann. Durch den Fokus auf die Sensoren ist das Modell aus sich heraus domänenunabhängig. Um Kontextinformationen zu erfassen, wird ein Sensor an einer Entität angebracht und charakterisiert diese Entität durch seine gemessenen Werte. Die Implementierung erfolgt wie in den vorangegangenen Modellen mittels OWL. In diesem Modell wird zusätzlich noch RDFS, das Resource Description Framework Schema, genutzt, um die Struktur der Daten zu formalisieren.

Das Projekt zeigt, dass sich eine Domänenunabhängigkeit auch durch ein spezielles Sensormodell realisieren lässt. Jedoch ist eben dies auch die Schwachstelle, denn die Qualität der Sensormodellierung bestimmt auch die Qualität der Kontextdaten und damit die Verwendungsmöglichkeiten.

Fazit

Die vorgestellten Projekte zeigen, dass es für ein komplexes, kontextbewusstes System essentiell ist einen Weg zu finden sowohl unabhängig von den einzelnen Anwendungen zu bleiben, als auch einen hohen Wiederverwendungswert bei den einzelnen Komponenten zu erreichen. Darüber hinaus zeigt jedoch auch die weite Verbreitung von OWL, dass der Fokus häufig auf der Modellierung im Rahmen statischer Ontologien mit ressourcenstarken Systemen liegt. Des Weiteren ist die Herkunft der Kontextinformationen für den Nutzer intransparent oder erfordert ein komplexes Verständnis des Systems. Dies widerspricht jedoch dem Ideal, dass der Nutzer mit seinen Daten selbstbestimmt umgehen kann, sie also leicht verstehen und auch ihren Verbleib und Nutzen kontrollieren kann. Eine Erweiterbarkeit wurde meist mit der Unterstützung von OWL angenommen, da hier Konzepte existieren, um Ontologien miteinander zu verknüpfen. Die Komplexität, die sich hieraus für selbst einfache Anwendungen ergibt, scheint jedoch außer Acht gelassen worden zu sein.

Ein System, bei dem ein Weg gefunden wurde das Wissen mit den Komponenten transparent zu teilen und bei dem gleichzeitig bereits vom Designzeitpunkt her ein Grundkonzept für Datenschutz und Privatsphäre verankert war, konnte in den existierenden Arbeiten nicht gefunden werden. Eine Beteiligung der Nutzer bei der Weiterentwicklung und Pflege des Systems scheint bei den existierenden Konzepten allein durch ihre innere Komplexität stark erschwert.

2.5.3 Empfehlungssysteme für nutzergenerierte Dienstensembles

Da Empfehlungssysteme mit dem Szenario dieser Arbeit verwandt und konzeptionell vergleichbar mit kontextbewussten Systemen sind, werden diese im folgenden Abschnitt erläutert. Es wird dabei auch herausgestellt, welche Vor- und Nachteile die verschiedenen Ansätze untereinander und für das gegebene Szenario dieser Dissertation haben.

Ein Empfehlungssystem wird wie folgt definiert:

Definition 6 (Empfehlungssystem)

> Ein Empfehlungssystem (oft auch Recommender-System genannt) ist ein System, das einem Benutzer in einem gegebenen Kontext aus einer gegebenen Entitätsmenge aktiv eine Teilmenge „nützlicher“ Elemente empfiehlt.
>
> (aus [Klahold 09])

Diese Teilmenge wird auf Grundlage von Qualitätsfunktion bestimmt, die entsprechend des Profils des Nutzers, der Eigenschaften der Entitäten und des Kontextes der Situation für diese Teilmenge einen maximalen Nutzwert ergibt. Generell werden drei verschiedene Verfahren bei Empfehlungssystemen unterschieden. Diese sind die „Content-Based-Filtering“, „Collaborative Filtering“ und hybride Verfahren.

Beim Content-Based-Filtering stehen die Entitäten selbst im Mittelpunkt. Sie werden analysiert und auf Basis dieser Daten werden Ähnlichkeitsmaße zwischen den Entitäten bestimmt.

Wenn ein Nutzer eine Entität nutzt, können ihm danach Entitäten angeboten werden, die eine hohe Ähnlichkeit zu der genutzten aufweisen. Diese Methode ist damit unabhängig von anderen Nutzern und verlangt, dass die Entitäten sich dafür eignen auf eine Ähnlichkeit untersucht zu werden. Der Vorteil ist, dass sich gerade für Anwendungszwecke mit vielen gleichartigen Entitäten, wie wissenschaftlichen Publikationen in einer Datenbank, gut Ähnlichkeitsmaße finden und sich so auch neue Entitäten eingruppieren lassen. Problematisch sind für diese Methode Objekte, die sehr unterschiedlich sind, wie Videos und Textdateien. Ein Vertreter dieser Methode ist CiteSeer[7], eine Webseite, die unter anderem die Referenzen in wissenschaftlichen Artikeln nutzt, um diese zu gruppieren.

Beim Collaborative Filtering werden der Nutzer und sein Verhalten analysiert. Die Idee dahinter ist, dass ähnliche Nutzer auch ähnliche Präferenzen für die verschiedenen Entitäten haben. Für die Analyse selbst können direkte Eingaben des Nutzers in Form von Bewertungen verwendet werden, aber auch implizite Informationen wie das Aufrufen des Dienstes selbst oder das registrierte Ereignis, dass ein Nutzer eine Dienstbeschreibung angesehen hat. Der Vorteil von diesem System ist, dass die Entitäten nicht gleichartig sein müssen. Lediglich die Nutzerprofile müssen vergleichbar sein.

Primär sind nach [Klahold 09] drei Schwächen des Collaborative Filterings zu nennen. Die erste Schwäche wird als Kaltstart-Problem bezeichnet und begründet sich darin, dass Systeme, die noch keine Nutzeraktivitäten enthalten, Nutzern keine Empfehlungen geben können. Dazu kommt, dass Entitäten, die noch niemand verwendet hat, nicht bewertet werden können. Ebenso können neue Nutzer noch nicht klassifiziert werden und so kann ihnen initial auch nichts empfohlen werden. Die zweite Schwäche ist der Lemming-Effekt, der sich auch bei amazon beobachten lässt. Wenn ein neuer Bestseller veröffentlicht wird - z.B. ein weiterer Harry Potter Band - so wird dieser von den unterschiedlichsten Nutzern gekauft und damit auch sehr vielen Nutzern empfohlen, auch wenn diese sich primär für Gartenbau oder andere entfernte Themen interessieren. Die dritte Schwäche wird als Sparsity-Problem (engl. für Seltenheit) bezeichnet und beschreibt, dass die Empfehlungsgüte stark fällt, sobald mehr als 99,5% aller Empfehlungselemente keine Bewertung durch einen Nutzer haben [Melville 02].

Der wohl bekannteste Vertreter hier ist der Versandhändler amazon.com, der den Kunden auf Basis der betrachteten und gekauften Produkte andere Artikel empfiehlt, die von Kunden mit ähnlichen Interessen gekauft oder betrachtet wurden.

Hybride Verfahren versuchen durch Kombination der beiden Verfahren die jeweiligen Vorteile zu verstärken und die Nachteile zu verringern. Dabei gibt es vier Möglichkeiten diese zu kombinieren [Adomavicius 05]. Einerseits können die Verfahren separat angewendet und die Ergebnisse kombiniert werden oder es kann ein umfassendes Modell genutzt werden, welches beide Verfahren mit ihren Chartakteristika vereint. Alternativ dazu ist es möglich, einerseits Content-Based-Filtering durchzuführen und zusätzlich Charakteristika des Collaborative Filtering zu nutzen oder umgekehrt ein Collaborative Filtering mit Charakteristika des Content-Based-Filtering zu kombinieren.

[7] http://citeseer.ist.psu.edu/ [Online letzter Zugriff 15.02.2018]

Fazit

Für den Anwendungsfall dieser Dissertation stellt sich besonders das Collaborative Filtering als interessante Methode dar, da die Ähnlichkeit der Nutzer durch ihre Rollen innerhalb der Universität und ihre Gruppenzugehörigkeit durch ihre gemeinsamen Veranstaltungen bestimmbar ist. Außerdem können Interaktionen der Nutzer mit den Diensten gut erfasst werden, da viele universitätsinterne Dienste eine Authentifizierung erfordern. Gleichzeitig ist die Menge der Dienste, die beispielsweise für eine Veranstaltung potentiell verfügbar sind, sehr groß, da sie unter anderem Einträge in der Wikipedia, wissenschaftliche Publikationen sowie andere Lehr- und Lernmaterialien einschließen kann. Die Qualität und Relevanz innerhalb der Menge variiert dabei stark. Daher ist die Bewertung durch die Nutzer essentiell für die Bestimmung einer relevanten und qualitativ hochwertigen Teilmenge an Diensten. Die Bestimmung eines Ähnlichkeitsmaßes auf den Diensten selbst ist nicht trivial, da hier in erster Linie heterogene Entitäten vorliegen und die Dienste erst durch die Nutzer in das System eingetragen werden. Ein Ähnlichkeitsmaß ließe sich jedoch zwischen den Veranstaltungen bilden, die sich zwar über mehrere Semester hinweg weiterentwickeln, jedoch in den Kernbereichen gleiche Inhalte vermitteln. Somit wäre für das vorliegende Szenario ein Collaborative Filtering mit Charakteristika des Content-Based-Filtering ein vielversprechender Weg.

2.5.4 Herausforderungen bei großflächigen, pervasiven Systemen

Überträgt man die Idee einer intelligenten Umgebung von einem abgeschlossenen Bereich wie einem Raum auf ein Szenario mit einer heterogenen, offenen Struktur, ergibt sich eine Vielzahl an neuen Herausforderungen. Dies macht es erforderlich, die Konzepte, die sich für die kleinen, homogenen Strukturen etabliert haben, zu überdenken und Schwerpunkte zu setzen, die den veränderten Anforderungen entsprechen. So weitet sich die Heterogenität bei offenen Strukturen auch auf die Prozesse und Nutzer aus. In einem Szenario mit einem intelligenten Raum oder Haus sind die Nutzergruppen und Prozesse beschränkt, während sie bei einem Szenario wie einem intelligenten Campus vielfältiger sind und sich auch verändern können, wenn sich die Situation ändert.

Eine Betrachtung von Herausforderungen in diesem Umfeld wird in [Kiani 11] gegeben. Der Hauptaspekt, auf den hier eingegangen wird, ist die Koordination, da sich die Nutzung eines komplexen Kontextmodells oder der Aufbau einer Kommunikationsinfrastruktur ohne eine geeignete Methode zur Koordination der einzelnen Elemente des Systems erschwert. Im Fokus stehen dabei die Anforderungen von mobilen Geräten, da sie einerseits die Schnittstelle zum Nutzer darstellen und andererseits eigene Restriktionen bei Energieverbrauch und -kapazität, der Konnektivität und der Darstellung von Inhalten mitbringen. Als Lösung wird ein System vorgeschlagen, das einerseits die interagierenden Komponenten entkoppelt und andererseits koordinierende Einheiten wie den Kontextserver zentralisiert. Das so entstehende System soll dadurch sowohl geographisch, administrativ als auch in seinem Funktionsumfang skalierter sein.

Eine weitere Betrachtung erfolgt in [Cook 12] mit der Thematisierung der Skalierung des Pervasive Computings. Dabei wird ein pervasives System als skalierbar bezeichnet, wenn durch

seine Ausweitung die Fähigkeiten und der Nutzen des Systems stärker steigen als der Overhead. Des Weiteren werden einzelne Möglichkeiten der Skalierung des Pervasive Computings betrachtet, wobei gerade die Skalierung durch Cloud Computing einzelne Möglichkeiten benennt, die in der vorliegenden Arbeit genutzt werden. Neben der Verwendung von drahtlosen Netzwerkverbindungen im städtischen Bereich umfasst dies auch die Einbindung der Nutzer als Crowd zur Erfassung und Bewertung von Informationen. Ferner wird auf die Effekte eingegangen, die eine Ausweitung des Pervasive Computings auf den Menschen haben kann. So wird darauf hingewiesen, dass die Nutzer aktuell zu wenig Kontrolle über ihre Daten haben und dass die Designer zukünftiger, pervasiver Systeme grundlegend auf den Schutz der Privatsphäre der Nutzer und den Schutz der Interessen von Organisationen und Unternehmen achten müssen.

Die Skalierung führt ebenfalls dazu, dass Aspekte, die vorher nicht betrachtet werden mussten, nun ein neues Gewicht bekommen. Dazu gehört zum Beispiel die Wartung des gesamten Systems. So war es in kleinen, geschlossenen Systemen einfach davon auszugehen, dass jene, die das System konzipiert haben, auch für dessen Wartung und Instandhaltung sorgen können. Bei einem offenen System muss der Prozess der Wartung und Pflege jedoch umso mehr betrachtet werden, da durch die Heterogenität der Prozesse und Geräte häufiger Fachkenntnisse erforderlich sind. Analog dazu ist auch die Weiterentwicklung des Systems von der Skalierung betroffen. Daher sollte auch hier eine Methode gefunden werden, die das Problem auf viele, kleinere Probleme aufteilt und die Lösungen wiederverwendbar macht.

Weitere von der Skalierung betroffene Aspekte sind der Zugriff auf die Funktionalität der Dienste und die Mobilität. Für die verschiedenen Nutzergruppen und Nutzungssituationen müssen Methoden gefunden werden, die einerseits eine Anpassung des Dienstumfangs an die Nutzergruppe ermöglichen und andererseits eine für die Situation entsprechende Auswahl an Diensten bieten. Des Weiteren muss auch die Mobilität des Nutzers betrachtet werden, da dieser einerseits an verschiedenen Orten verschiedenen Prozessen nachgeht, jedoch bestimmte Prozesse auch mit ihm wandern und an verschiedenen Orten und auf unterschiedlichen Endgeräten unterschiedlich behandelt werden müssen.

Fazit

Es zeigt sich, dass ein System mit den genannten Anforderungen nicht nur auf verteilten Komponenten basieren, sondern auch seine internen Prozesse zur Wartung, Pflege und Erweiterung verteilen muss. Eine Entkopplung dieser Prozesse und Verteilung auf die verschiedenen Untereinheiten sollte es ermöglichen, eine Vielzahl an kleineren Aspekten zu behandeln, wie die Einbindung eines neuen Sensors oder die Definition einer neuen Nutzergruppe. Zusammengeführt können diese in einem großflächigen Umfeld durch Wiederverwendung einen größeren Nutzen entfalten. Die Methode zur Koordination könnte daher auf der Idee des Crowdsourcing beruhen, bei der das Problem auf viele, kleine, unabhängige Einheiten verteilt wird und bei der Wiederzusammenführung die besten Einzel- und Gesamtlösungen genutzt werden. Dafür werden neben den verteilten Strukturen zentrale Elemente benötigt, die hier den Prozess der Verteilung und Wiederverwendung koordinieren.

2.6 Synthese der Anforderungen für ein nutzergetriebenes, kontextbasiertes System im großflächigen Einsatz

Entsprechend der in diesem Kapitel vorgestellten Konzepte und Herausforderungen ergibt sich eine Reihe von Anforderungen. Diese werden im Folgenden in konzeptuelle, technische und soziale Anforderungen unterteilt.

Konzeptuelle Anforderungen

Das Herz eines kontextbasierten Systems ist die Art und Weise, wie es Kontextinformationen aufnimmt, verarbeitet und nutzt. Um dies für den vorliegenden Anwendungsfall möglichst effektiv zu machen, ist eine Ontologie notwendig, die sich verteilen lässt und nicht an einer zentralen Stelle vollständig vorliegen muss. Dies kann erreicht werden, indem eine mehrstufige Ontologie definiert wird, in der alle Teilnehmer ihre domänen- und anwendungsspezifischen Konzepte auf einheitliche Basiskonzepte zurückführen. Dabei sollten die Ontologien in standardisierten, offenen Formaten vorliegen, um eine allgemeine Kommunikation gewährleisten zu können [Baldauf 07].

Als Eingabe sollten sowohl virtuelle, logische als auch physische Sensoren unterstützt werden. Dabei sollte die Wiederverwendbarkeit und Erweiterbarkeit der Sensorschnittstellen im Fokus stehen, um für die jeweiligen Anwendungsfälle Sensoren nicht stets aufwändig anbinden zu müssen [Baldauf 07, Tan 12].

Die Empfehlung von Diensten sollte sich Konzepte des Collaborative Filterings zunutze machen, da sich im vorliegenden Anwendungsszenario die Nutzer gut in Gruppen einteilen lassen. Des Weiteren sind den Nutzergruppen Prozesse und Rollen zurechenbar, was eine Ausnutzung von Kontextquellen, wie Stunden- und Vorlesungsplänen, erlaubt. Eine weitergehende Betrachtung von Ähnlichkeitsmaßen könnte in weiteren Arbeiten die Grundlage für Empfehlungen bieten, die für neue, unerschlossene Dienstmengen genutzt werden können.

Technische Anforderungen

Der Fokus der technischen Anforderungen liegt auf der Bewältigung der Heterogenität auf mehreren Ebenen. So sollte auf der Service-Ebene eine zentrale Vereinheitlichung genutzt und eine Schnittstellentransformation hin zu einem Webinterface angestrebt werden. Dies erlaubt sowohl eine direkte Nutzung der Dienste durch den Anwender als auch ein Einbinden der Dienste in andere Anwendungen. Eine Einbindung der Dienstempfehlungen in die Anwendung des Nutzers ist dabei stets einer Dienstnutzung durch eine separate Anwendung vorzuziehen.

Um die Kommunikation innerhalb des Systems effektiv zu gestalten, kann eine verteilte Architektur mit verschiedenen Knoten genutzt werden. Diese Knoten sind dabei so zu konzipieren, dass sie innerhalb einer Domäne effizient kommunizieren können und nur bei der Kommunikation nach außen einer Transformation zwischen den Kontextmodellen bedürfen. Um bei Ausfall eines Knotens trotzdem auf die hier zuletzt verfügbaren Informationen zugreifen zu können, empfiehlt sich eine Redundanz der Knoten. Diese kann hier dazu beitragen, dass auch

wenn einzelne Knoten nicht erreichbar sind, deren Funktionalität auch über andere Knoten repliziert werden kann.

Der Kommunikationsaufwand sollte besonders für mobile Geräte gering gehalten werden, da diese als Endschnittstelle für den Nutzer besonders geeignet sind. Außerdem ist darauf zu achten, dass der Nachrichtenaustausch in offenen und standardisierten Formaten erfolgt und so überprüft werden kann, welche Informationen weitergegeben werden. Dabei ist auf eine breite Abdeckung von existierenden Protokollen und Schnittstellen zu achten.

Die Aggregation von Diensten für den einzelnen Nutzer sollte dabei möglichst auf einem Knoten erfolgen, über den dieser die Kontrolle hat. Bei Nutzergruppen sollte die Kontrolle ebenfalls in der Nutzergruppe selbst liegen. Für öffentliche Knoten sollten keine Zugangsbeschränkungen existieren, die eine Identifikation des Nutzers erfordern.

Um die Weiterentwicklung zu fördern, müssen ausschließlich solche Softwarekomponenten verwendet werden, die unter geeigneten Lizenzen stehen. Dabei sollte die Struktur des gesamten Softwaresystems so gewählt werden, dass die Lizenz einer Komponente nur diese selbst und nicht das gesamte System betrifft.

Soziale und organisatorische Anforderungen

Betrachtet man die sozialen Aspekte, innerhalb derer ein solches System genutzt wird, so ergeben sich für eine langfristige Unterstützung durch die Nutzer ebenfalls spezielle Anforderungen. So waren laut einer Umfrage des ibi Research an der Universität Regensburg 50% der Befragten der Meinung, dass jeder Einzelne selbst für den Schutz seiner Daten sorgen muss [Wittmann 13]. Dies ist jedoch nur effektiv möglich, wenn der Nutzer selbst die Kontrolle über seine Daten behält und genau darüber informiert ist, wo seine Daten verwendet und gespeichert werden. Daher muss der Nutzer für diese Form der Akzeptanz über den Verbleib und die Verwendung seiner Daten mitbestimmen können und nicht nur über den Umfang der benötigten Daten informiert werden.

Der Schutz der Daten selbst sollte direkt im Design des Systems fundiert werden. Diese Herangehensweise wird auch als Privacy-by-Design [Langheinrich 01] bezeichnet und dient unter anderem auch als Konzept für die elektronische Gesundheitskarte [Schaar 10]. Dabei werden Aspekte wie die möglichst direkte Kontrolle des Nutzers über den Speicherort und die Nutzungsart seiner persönlichen Daten, Datensparsamkeit und die Verwendung von anonymisierten oder pseudonymisierten Daten, wo dies möglich ist, verlangt. Dies kann durch einen monodirektionalen Informationsfluss unterstützt werden, der die privaten und schützenswerten Daten auf den Systemen belässt, die unter der Kontrolle des Nutzers stehen. Die weniger persönlichen Daten werden zu diesen geschickt und dienen dort der Empfehlung von Diensten.

Der Aspekt des Administrationsaufwandes ist ebenfalls zu betrachten. Hier ist einerseits klar, dass es eine Gruppe von Nutzern geben muss, die über ein fundiertes Fachwissen verfügt, um das System in seinem Kern zu administrieren.

Jedoch kann gerade der Aspekt der Anpassung an den Nutzer hier nur durch den Nutzer selbst erfolgen, wenn die Schnittstellen verständlich gestaltet sind und die Nutzer zum Teilhaben

am System motiviert werden. Dies kann zum Beispiel mit Mitteln der Gamification erreicht werden.

Abschließend ist auch die Frage nach einer Methode zur Erweiterung des Systems zu stellen. Dies einer zentralen Instanz zu überlassen, die für alle Nutzer neue Prozesse und Dienste integriert, widerspricht dem Konzept eines offenen und dynamischen Systems. Daher ist auch hier ein nutzergetriebener Ansatz erstrebenswert, bei dem die Nutzer ihre Prozesse und Dienste weitestgehend allein in das System einbinden können und nur in technischen Fragen Administratoren konsultieren müssen.

Die Bedingung dafür ist, dass die verschiedenen Teile des Systems unter geeigneten Lizenzen veröffentlicht werden, die zum einen die Nutzung und Weiterverbreitung fördern und zum anderen auch geeignet sind einem Missbrauch vorzubeugen. So ist es notwendig, dass die Lizenzen an die einzelnen Bestandteile angepasst werden und miteinander kompatibel sind.

Zusammenfassung der Anforderungen

- Konzeptuelle Anforderungen
  - Verteiltes, skalierbares Kontextmodell mit Unterstützung dezentraler Weiterentwicklung
  - Nachhaltige Unterstützung logischer, virtueller und physischer Sensoren
  - Dienstempfehlung auf Basis von nutzergenerierten Regeln/Verknüpfungen zu Prozessen und Kontextfakten innerhalb der jeweiligen Domäne
  - Monodirektionaler Datenfluss zur Sicherung schutzwürdiger Daten nach dem Konzept des Privacy-by-Design
- Technische Anforderungen
  - Zentrale Vereinheitlichung auf Ebene der Dienste
  - Plattformunabhängigkeit durch Transformation zu einer browserfähigen, graphischen Schnittstelle
  - Umsetzung mit autarken, domänenspezifischen Knoten und nutzereigenen Knoten
  - Replikation und Redundanz von Knoten zur Sicherung der Funktion bei Ausfällen
  - Implementierung unter Nutzung von Systemen und Softwarekomponenten mit geeigneten Lizenzen und aktiven Entwicklergemeinden
  - Nutzung einer Kommunikationsinfrastruktur, die geeignet ist über Netzwerkgrenzen hinweg effizient Daten zu übertragen
  - Einbindung der Dienstempfehlungen in den Software-Workflow des Nutzers
  - Unterstützung mobiler Geräte als Endpunkte der Dienstempfehlung
- Soziale und organisatorische Anforderungen

- Kontrolle des Datenflusses durch den Nutzer
- Integration von geeigneten Motivationskonzepten zur Steigerung und Aufrechterhaltung der Beteiligung von Anwendern an der Verbesserung und Weiterentwicklung des Systems
- Verteilung des Wartungsaufwandes auf verschiedene Nutzerrollen
- Veröffentlichung des Systems und seiner Schnittstellen unter Lizenzen, die zu einer Erweiterung einladen und gleichzeitig einen Schutz von Eigenentwicklungen bieten

2.7 Zusammenfassung

In diesem Kapitel wurden die für diese Arbeit relevanten Grundlagen vorgestellt. Dabei wurden zu den jeweiligen Thematiken Vor- und Nachteile einzelner Ansätze und Forschungsarbeiten erläutert und es wurden Aspekte dargestellt, die jeweils als Anforderung für das in dieser Arbeit entwickelte Konzept und dessen Umsetzung dienen. Des Weiteren wurden die Verwendung von Crowdsourcing und Diensteorientierung und die Verankerung des Schutzes der Privatsphäre im Design und Konzept der Arbeit motiviert. Die Inhalte dieses Kapitels werden in den folgenden Abschnitten zusammengefasst.

Beginnend mit einer Diskussion des Kontext-Begriffs und einer Gegenüberstellung der Kontextdefinitionen aus [Dey 00] und [Chen 00] wurde für diese Arbeit eine eigene Definition abgeleitet, die Kontext auf jene darstellbaren Informationen begrenzt, die Situationen beschreiben, in denen Anwendungen einen Mehrwert für den Nutzer bieten. Anschließend wurde das Erfassen von Kontextinformationen thematisiert und die drei Sensorklassen physisch, logisch und virtuell nach [Indulska 03] erläutert. Darauffolgend wurde eine Übersicht über verschiedene Modellierungsansätze von Kontext gegeben, die sich an den Klassifikationen von [Bettini 10] und [Strang 04] orientieren. Für alle Ansätze wurde ausgeführt, welche Vor- und Nachteile sie zur Modellierung des Kontextes im Rahmen dieser Arbeit bieten und es wurde festgestellt, dass unter der Maßgabe eines Systems, das durch den Nutzer aktiv weiterentwickelt werden soll, Ansätze wie RDF ausscheiden, da hier ein großes Fachwissen erforderlich ist. Ein logik- oder regelbasiertes System stellt für diesen Fall die beste Lösung dar, da hier der Kompromiss aus Darstellbarkeit von Informationen und Verständlichkeit für den Nutzer geeignet erscheint.

Im nächsten Abschnitt wurde die Diensteorientierung für die Sicht auf die Vorgänge und Prozesse an Hochschulen motiviert. Service-orientierte Architekturen wurden als Systemarchitekturen in der Informatik entsprechend der Definition von [Melzer 07] vorgestellt und es wurde auf die Idee der Pervasive University aus [Tavangarian 09a] als Verknüpfung beider Ansätze eingegangen. Die in diesem Umfeld bereits entstandenen Dissertationen von [Zender 10] und [Dressler 10] haben bereits Konzepte und Umsetzungen auf Netzwerk- und Service-Ebene vorgestellt. Es wurde dargestellt, wie die vorliegende Arbeit von der Dienstinfrastruktur und ihren Protokollen abstrahiert und die Pervasive University um eine soziale Sicht erweitert. Dabei wurde der Nutzer in den Prozess der Dienstvermittlung integriert und erhielt somit die Möglichkeit, aktiv an der Dienstauswahl und Weiterentwicklung des Systems mitzuwirken.

Mit den Hochschulen als Zielumfeld für diese Arbeit wurde der Begriff des Dienstes unter diesem Aspekt und der Zielstellung dieser Arbeit genauer betrachtet. Ausgehend von den technischen Dienstdefinitionen von [Melzer 07] und [Erl 08] sowie der Dienstleistungsdefinition von [Gab 13] wurde eine eigene Definition gefunden, die von der technischen Ebene abstrahiert und einen Dienst als Entität definiert, die mit einer physischen oder virtuellen Schnittstelle ausgestattet ist und über diese Interaktionen anbietet, die entweder eine Information über ein Objekt zurückliefern oder dieses manipulieren. Ein Dienst kann hierbei alleine oder auch zusammen mit anderen Diensten ein Objekt darstellen.

Mit Hochschule als Umfeld und der eigenen Dienst-Definition wurde an Hand der in [Dey 00] zur Klassifikation von Kontext vorgeschlagenen Relationen (Ort, Identität und Aktivität) versucht, real vorkommende Dienste an Hochschulen zu klassifizieren, um daraus Rückschlüsse auf die Relevanz und Eignung zur Dienstempfehlung zu erhalten. Es zeigte sich, dass diese Klassifikationen geeignet sind, um Regeln zu formulieren, auf denen eine Dienstauswahl ohne persönliche Identifikation des Nutzers basieren kann. Jedoch zeigte sich damit auch, dass die Identitäten des Benutzers bzw. dessen Rollen innerhalb der Prozesse entscheidend sind für eine Dienstauswahl. Davon ausgehend wurde der Begriff des Nutzers genauer betrachtet und es wurde für diese Arbeit eine Definition gegeben, die den Nutzer auf Personen begrenzt, die auf Dienste über deren Schnittstellen zugreifen und für sich persönlich verwenden. Als Erweiterung der Identitätsrelation wurde der Grad an Anonymität als Ausgangspunkt einer Klassifikation genommen und es wurde gezeigt, dass dieser gut geeignet ist, um die Zielgruppe von Diensten einzugrenzen.

Zur Klassifikation der Dienste und der Situationen, in denen sie einen Mehrwert bieten, wurde im nächsten Abschnitt Crowdsourcing als aussichtsreicher Ansatz vorgestellt. Diese Einschätzung wurde im Laufe der Erstellung dieser Arbeit durch [Cook 12] geteilt, die dem Crowdsourcing aus Sicht des Pervasive Computings das Potential zusprechen großflächig, kostengünstig und effektiv Daten zu sammeln und daraus Informationen zu extrahieren. Gleichzeitig werden in dieser Publikation Motivation der Nutzer, Schutz vor böswilliger Manipulation und die Wahrung der Privatsphäre als Herausforderungen benannt, die ebenfalls in der vorliegenden Arbeit thematisiert werden. Unter Einbeziehung des Crowdsourcings in das Konzept dieser Arbeit ergab sich die Betrachtung, dass hier eine zum Teil ungesteuerte, kontinuierliche Weiterentwicklung eines heterogenen, verteilten Systems zu erwarten ist, welches an eine sich wandelnde Umgebung angepasst wird. Dies führte zu der Überlegung, dass sich in der Evolutionstheorie nach [Darwin 59], die sich ja ebenfalls mit einer kontinuierlichen Anpassung an eine sich verändernde Umwelt befasst, Faktoren finden lassen, die die Entwicklung begünstigen und die sich auch auf die nutzergetriebene Dienstempfehlung dieser Arbeit übertragen lassen. Aus den identifizierten Faktoren wurden drei Prämissen entwickelt, die die kontinuierliche Weiterentwicklung intelligenter Umgebungen allgemein und der Dienstempfehlung in intelligenten Umgebungen im Speziellen fördern. So wurde ein Nebeneinander von verschiedenen, nutzergenerierten Beschreibungen von Diensten und Dienstumfängen des gleichen Dienstes als sinnvoll identifiziert, da sich so geeignetere Dienstempfehlungen erstellen lassen. Gleichzeitig lassen sich so bei einer Änderung der technischen Umwelt leichter Dienste als Vorlagen auswählen, aus denen sich mit geringem Aufwand neue Versionen ableiten lassen, die den neuen Gegebenheiten angepasst sind. Dazu müssen die verwendeten Sprachen und Protokolle der Beschreibungen und Dienste offen zugänglich und erweiterbar sein. Dies befördert auch einen Austausch von Diensten und Beschreibungen über Domänengrenzen hinweg.

Abschließend wurde eine Klassifikation erstellt, die die möglichen Entwicklungsprozesse in drei Gruppen zusammenfasst, erläutert, wodurch diese sich auszeichnen, und darstellt, welche Systemeigenschaften diese befördern und welche Anforderungen diese Prozesse an die Nutzer stellen.

Nachdem die organisatorischen und konzeptionellen Grundlagen des Konzeptes dieser Arbeit beschrieben wurden, wurde im Abschnitt 2.4 die technische Seite genauer betrachtet. Dazu wurde ausgehend von den Definitionen von [Weiser 91] und [Cook 07] für pervasive bzw. intelligente Umgebungen eine eigene Definition erstellt, die eine intelligente Umgebung auf einen räumlichen Bereich beschränkt, in dem einem Nutzer durch ein digitales System eine reaktive oder auch proaktive Unterstützung zuteil wird, die auf den Informationen basiert, die dem System über den Nutzer und dessen Umgebungen zur Verfügung stehen. Anschließend wurden verschiedene Systemarchitekturen für verteilte Umgebungen vorgestellt und ihre Eignung für intelligente Umgebungen diskutiert. Dabei wurden besonders bei Peer-To-Peer-Architekturen und Broker-Architekturen Vorteile identifiziert. Neben den Architekturen wurden auch die Architekturstile nach [Tanenbaum 08] betrachtet und unter den Anforderungen intelligenter Umgebungen diskutiert. Den Abschluss der technischen Betrachtung bildete eine Zusammenfassung der Anforderungen für intelligente Umgebungen an Hochschulen, die die Aspekte der Unterstützung heterogener Geräte, Skalierbarkeit und Nutzerorientierung als zentral herausstellt.

Abschnitt 2.5 befasste sich mit den verwandten Arbeiten im Forschungsfeld dieser Dissertation. Ziel war dabei die Identifikation von Aspekten, die sich als Anforderungen für das in dieser Arbeit entwickelte System eignen, sowie von Fehlstellen, die durch diese Arbeit adressiert werden können. Dabei wurde von einer Übersicht der Systeme für kontextbewussten Zugriff auf Dienste abgesehen und auf die bereits existierenden Übersichten, z.B. [Bellavista 13], verwiesen. Stattdessen wurde eine Systematik entsprechend dem Aufbau von kontextbewussten Systemen erstellt und es wurden Arbeiten zur Sammlung, Abbildung und Nutzung von Kontextdaten hinsichtlich der Zielstellung dieser Arbeit ausgewählt und diskutiert. Bei den Systemen zur Aggregation von Daten und Erfassung von Situationsbeschreibungen gibt es bereits eine Reihe kommerzieller Systeme, die versuchen Nutzern durch die Verknüpfung von Nachrichten- und Informationsströmen mit Filtern und Aktionen Assistenz anzubieten. Diese Systeme sind proprietär und erlauben es so dem Nutzer nicht, das System für eigene Zwecke zu erweitern. Gleichzeitig vertraut der Nutzer bei intensiver Nutzung einem kommerziellen Anbieter sehr sensible Daten zu seiner Position und seinem Kommunikationsverhalten an. Das einzige gefundene offene und freie System war [Cantino 13], das es Nutzern erlaubt eigene Agenten zu erstellen, die bei Erfüllung bestimmter Bedingungen eigene Aktionen ausführen können. Dazu sind jedoch aktuell noch weitreichende Kenntnisse in verschiedenen Programmiersprachen notwendig und die Schnittstellen und Anwendungsfälle sind begrenzt. Gleichzeitig wird jedoch mit der Offenheit der Entwicklung und der Lizenzierung unter der MIT-Lizenz auch gezeigt, wie sich Weiterentwicklungen fördern lassen. Die Betrachtung der Systeme mit Umsetzung von Kontextontologien zeigte, dass die Entwicklung hin zu einer Umsetzung mit mächtigen Ontologien mit OWL oder RDF geht. Diese sind besonders zur Modellierung komplexer Situationen geeignet, jedoch sind sie selbst sehr komplex und verlangen vom Nutzer ein weitreichendes Verständnis ihrer Konzepte und von dem verarbeitenden System viele Ressourcen. Da es die verschiedensten Möglichkeiten der Verwendung von Kontextdaten gibt, wurde für den dritten Teil der Systematik der Bereich der

Empfehlungssysteme ausgewählt. Hier zeigte sich besonders das Collaborative Filtering als interessante Methode zur Anwendung in dieser Arbeit. Dabei werden Ähnlichkeiten zwischen den Eigenschaften der Nutzer einer Entität ermittelt, um darüber eine Qualitätsfunktion zu bilden, die aussagt, wie sehr einem anderen Nutzer mit bekannten Eigenschaften diese Entität nutzt. Den Abschluss der Diskussion verwandter Arbeiten bildete die Betrachtung der Herausforderungen großflächiger, pervasiver Systeme. So wird in [Kiani 11] besonders auf die koordinatorischen Herausforderungen eingegangen, die sich durch die Erweiterung der intelligenten Umgebungen auf große, räumliche Gebiete und eine Vervielfachung der eingebundenen Geräte ergibt. Darüber hinaus werden in [Cook 12] auch die sozialen Aspekte wie die Wahrung der Privatsphäre, der Unterstützung bei der Verwendung oder auch die Frage nach der Skalierung der Schnittstellen zur Interaktion mit den Systemen gestellt.

Abgeschlossen wurde dieses Kapitel durch eine Zusammenfassung der Anforderungen an ein nutzergetriebenes, kontextbasiertes System für den großflächigen Einsatz. Dabei wurden die in diesem Kapitel vorgestellten Grundlagen, Überlegungen sowie die Vorteile und Nachteile verwandter Arbeiten zusammengefasst und in konzeptuelle, technische und soziale Anforderungen eingeteilt.

Kapitel 3

CASA - Das Konzept für den kontextbewussten Dienstzugriff

Die im vorherigen Kapitel vorgestellten Ansätze und Überlegungen führten zur Entwicklung des Konzeptes für den kontextbewussten Dienstzugriff (engl. Context-Aware Service Access). Der Fokus des im folgenden CASA genannten Konzeptes liegt dabei auf der Erweiterung des Wissens und der Funktionalität des Systems durch den Nutzer als Ansatz für Nachhaltigkeit und Kompatibilität. Dabei bildet der CASA-Knoten das Herzstück als modulare und eigenständige Einheit.

In diesem Kapitel soll daher das Konzept des CASA-Knotens und das Konzept eines Netzes von Knoten dargestellt und diskutiert werden. Die Reihenfolge orientiert sich an einem Top-Down-Ansatz. Zuerst wird in Abschnitt 3.1 dargestellt, wie sich existierende Systemarchitekturen durch das CASA-Konzept erweitern lassen. Im entwickelten Konzept wird gezeigt, wie das Modell der Service-orientierten Architektur Anwendung findet, um eine Verbindung von Kontext und Diensten zu realisieren. Im Anschluss wird in Abschnitt 3.2 auf das Netz der CASA-Knoten eingegangen. Es wird ein Konzept beschrieben, das die Trennung des Kontextes zwischen verschiedenen Domänen ermöglicht und private, öffentliche und gruppenbezogene Informationen verwalten kann. Des Weiteren wird in Abschnitt 3.3 die Organisation der Knoten und des Netzes erläutert. Es wird gezeigt, wie

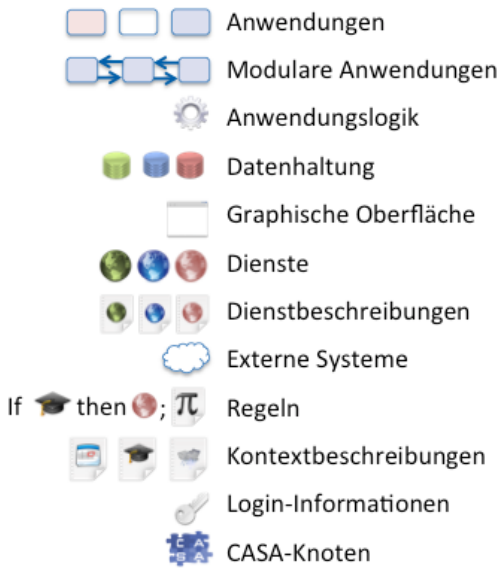

Abbildung 3.1: Verwendete Symbole in diesem Kapitel

dies eine Dienstauswahl von verschiedenen Dienstanbietern entsprechend den Anforderungen der Situation und über die verschiedenen Domänen hinweg ermöglicht.

Darauf folgend wird im Abschnitt 3.4 der Knoten selbst genauer betrachtet. Hier wird ebenfalls zuerst mit Architektur und Organisation begonnen. Danach wird genauer auf die Aktionen und Regeln innerhalb eines Knotens eingegangen. Ferner werden die Module zur Aggregation und Integration der Dienste sowie zur Erfassung der Kontextinformationen beschrieben.

Im Anschluss daran wird speziell auf die Möglichkeiten zur Nutzerunterstützung eingegangen und erläutert, wie sich der Knoten an die verschiedenen Nutzerfähigkeiten und Bedürfnisse anpassen lässt.

3.1 Erweiterung der Systemarchitekturen mit dem CASA-Ansatz

Ziel des CASA-Konzeptes ist die Erweiterung existierender Systeme um Funktionalitäten, die die Nutzer selbst hinzufügen und die entsprechend der Nutzersituation eingebunden werden. Dazu ist es notwendig, dass die Funktionalitäten bereits als Dienste verfügbar sind oder sich als solche in Form einer Webseite für den Nutzer erschließen lassen. Des Weiteren ist es Voraussetzung, dass die vorhandenen Systeme, in die Dienste integriert werden, diese Integration unterstützen oder andernfalls so erweitert werden, dass sie die Integration erlauben.

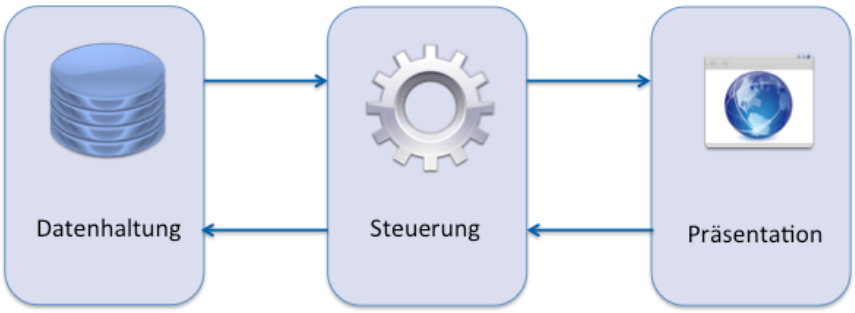

Abbildung 3.2: Klassische 3-Schichten-Architektur einer Anwendung

Die auf diese Art erweiterten Systeme werden im Folgenden als Wirtssysteme bezeichnet. Bei den hier betrachteten Wirtssystemen liegt in der Regel eine Drei-Ebenen-Architektur vor. Diese besteht, wie in Abbildung 3.2 dargestellt, aus der Präsentationsschicht, der Steuerungsschicht und der Datenhaltungsschicht. Dabei liegen in der Datenhaltungsschicht Informationen vor, die bei Anfragen durch einen Nutzer von der Steuerungsschicht aufbereitet und diesem über die Präsentationsschicht bereitgestellt werden. Der Nutzer interagiert ausschließlich mit der Präsentationsschicht, z. B. durch einen Browser oder den Client der Anwendung. Dies schützt die Datenebene vor unberechtigtem Zugriff und erlaubt eine klare Aufgabentrennung innerhalb des Systems.

Neue Inhalte und Dienste, die diesen Systemen hinzugefügt werden sollen, können unterschiedlicher Art sein. Dies beginnt beim statischen Einbinden von Hinweiswebseiten in bestimmten Schritten eines Prozesses und kann bis zur vollständig nutzer- und situationsab-

hängigen Integration von dynamischen Webseiten, Dokumenten und Funktionalitäten führen. Die Vielfalt der Quellen dieser Dienste reicht dabei von einer Datenquelle innerhalb des Wirtssystems bis hin zu einer vollständig separaten Web-Anwendung auf einem entfernten System.

Um in ein Wirtssystem diese neuen Inhalte einzubinden, ist es notwendig in der Steuerungsschicht eine Erweiterung durchzuführen, die die neuen Inhalte anfragen kann. Des Weiteren wird in der Präsentationsschicht eine Erweiterung eingebracht, die es erlaubt diese anzuzeigen. Dabei wird die native Steuerungsschicht an die Erweiterung so angebunden, dass sie dieser Informationen mitteilt, auf Basis derer sie entscheidet, ob zusätzliche Inhalte dargestellt werden sollen. Diese werden dann aus der jeweiligen Quelle bezogen und über die Erweiterung in der Präsentationsschicht dargestellt. Dies ist in Abbildung 3.3 mit der Einbindung einer externen Quelle dargestellt.

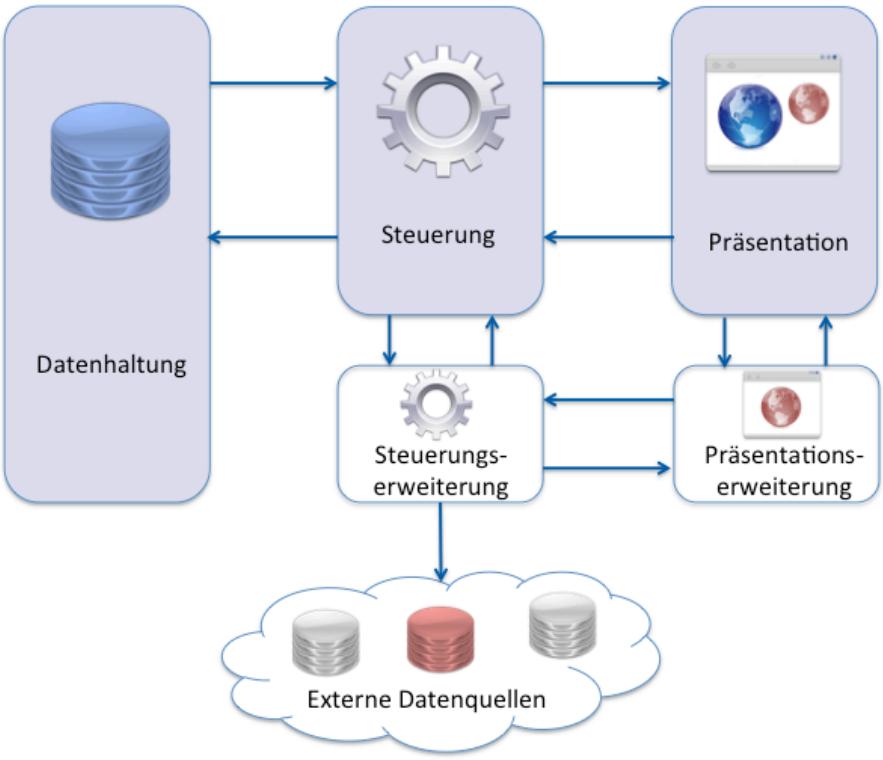

Abbildung 3.3: Erweiterte 3-Schichten-Architektur einer Anwendung mit Einbindung externer Inhalte (rot)

Erfolgt die Umsetzung auf diese Art, so kann der Aufwand bei der Erweiterung der Software überschaubar gehalten werden, da nur wenige Änderungen am Wirtssystem nötig sind. Jedoch setzt dies voraus, dass es diese Art der Erweiterung, z. B. durch ein Plugin-Konzept, unterstützt.

Sollen nicht nur externe Dienste in ein System importiert, sondern auch Dienste exportiert werden, so muss dies durch die Steuerungserweiterung in der Anwendung ermöglicht werden

und auch andere Wirtssysteme müssen über die entsprechende Erweiterung verfügen. Dies ist dargestellt in Abbildung 3.4.

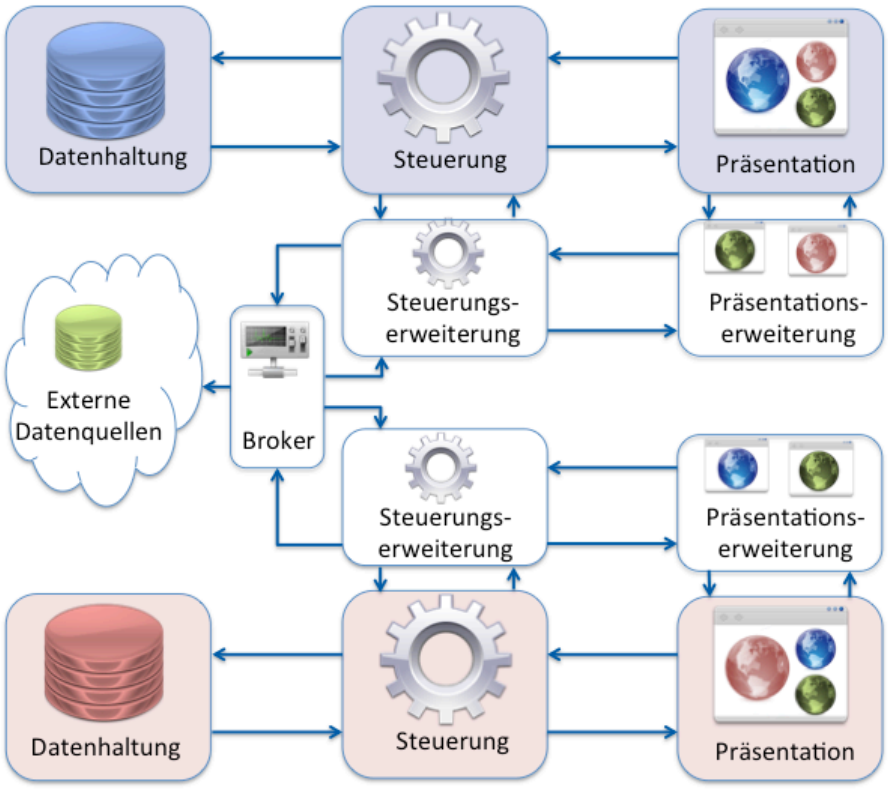

Abbildung 3.4: Erweiterte 3-Schichten-Architektur zweier Anwendungen mit Einbindung externer Inhalte (grün), sowie gegenseitiger Einbindung der Inhalte (rot und blau)

Dabei erlaubt das Modul, das an der Steuerungsschicht des Wirtssystems sitzt, auch Anfragen an die Steuerungsschicht, die nicht aus der wirtseigenen Präsentationsschicht kommen, sondern über einen externen Broker gestellt wurden. Die Antwort besteht hier aus einer Information (z.B. einer URL), die eine Einbindung über die Präsentationsschicht des Wirtssystems erlaubt. Dabei ist jedoch zu beachten, dass Sicherheitsstrukturen wie Logins zu berücksichtigen sind, da die Präsentationsschicht eventuell die Dienste andernfalls nicht fehlerfrei darstellen kann.

Die Architektur des CASA-Ansatzes orientiert sich an diesem Schema, das vergleichbar ist mit dem einer Service-orientierten Architektur. Dabei nimmt der CASA-Knoten die Rolle eines Brokers ein. Er verwaltet die Beschreibungen von Diensten, die durch Dienstanbieter (Provider) zur Verfügung gestellt werden. Dieser kann dabei sowohl der direkte Anbieter eines speziellen Dienstes sein als auch eine Instanz, die lediglich über die Dienstinformationen verfügt. Consumer kann eine Applikation mit einer CASA-Schnittstelle oder ein weiterer CASA-Knoten sein. Bei mehreren CASA-Knoten erfolgt die Dienstempfehlung kaskadierend mittels einer gerichteten Verknüpfung. Weiteres hierzu ist in Kap. 3.3.1 erläutert.

In Abbildung 3.5 ist diese Systemarchitektur dargestellt. Die CASA-Schnittstelle umfasst dabei im Folgenden die Einheit aus Steuerungs- und Präsentationserweiterung auf Seiten der Anwendung.

Entsprechend einer klassischen Service-orientierten Architektur erfolgt in einem ersten Schritt durch die Nutzer eine Eintragung (Publish) von Dienstbeschreibungen und Regeln, die in der Abbildung mit π dargestellt sind. Startet ein Nutzer die Anwendung, so stellt die CASA-Schnittstelle eine Dienstanfrage mit der aktuellen Situation, hier der Nutzerrolle, an den CASA-Knoten. Dieser Schritt (Find), wird durch den CASA-Knoten mit Dienstbeschreibungen beantwortet, die auf Basis der durch den Nutzer eingegebenen Regeln ausgewählt wurden. Dabei können auch externe Kontextquellen, wie im Beispiel ein Wetterdienst, angefragt werden, um die Situation möglichst gut zu erfassen. Abschließend werden die Dienstbeschreibungen durch die CASA-Schnittstelle ausgewertet und die passenden Dienste direkt von den externen Dienstanbietern eingebunden (Bind).

Im Unterschied zu einem Repository existieren in einem CASA-Knoten neben den Dienstbeschreibungen auch Regeln und Fakten zur aktuellen Situation, die für die Zuordnung zwischen Anfrage und Dienstbeschreibung genutzt werden. Die Zuordnung zwischen Anfrage und Dienst ist so nicht nur von der Anfrage, sondern auch von der Situation, dem Kontext, abhängig.

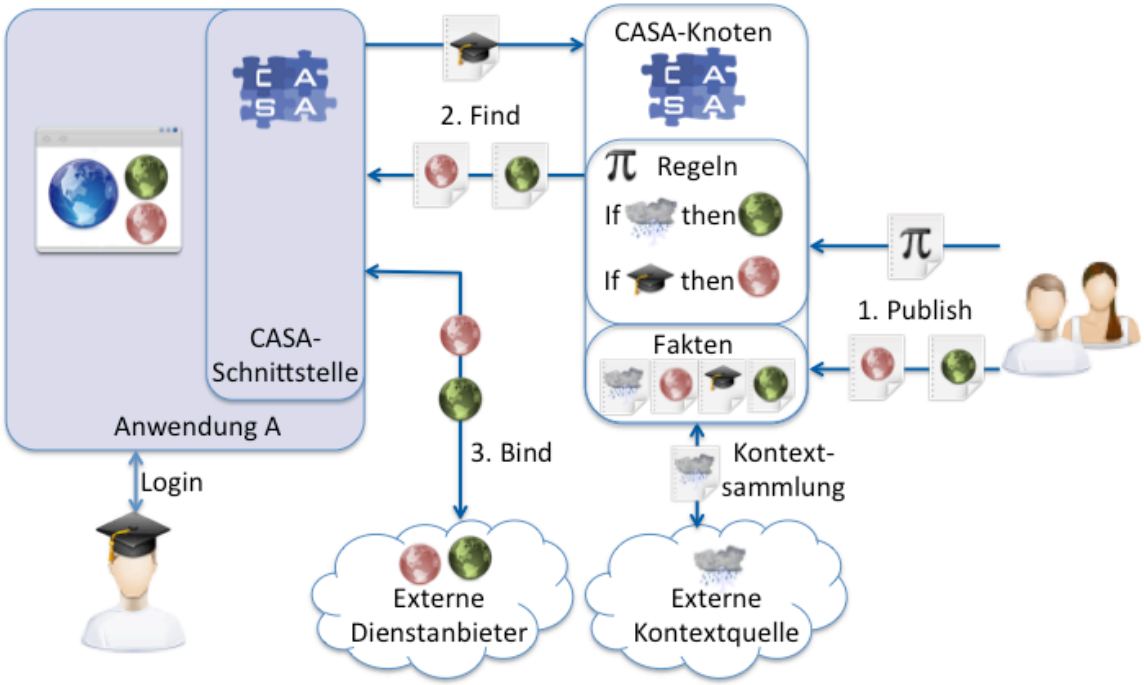

Abbildung 3.5: Organisation des CASA-Konzeptes

Um, wie bereits erwähnt, auch Sicherheitsstrukturen zu beachten und die Daten der Nutzer zu schützen, ist eine Klassifizierung der Knoten notwendig. Diese wird im folgenden Abschnitt genauer ausgeführt.

3.2 Typologie der CASA-Knoten

Um eine allgegenwärtige, skalierbare und domänenunabhängige Dienstvermittlung zu gewährleisten, ist es notwendig eine Trennung zwischen den Domänen, ihren Kontextmodellen und den verschiedenen Stufen der Privatsphäre zu realisieren. Die Domänen wurden klassifiziert nach öffentlichen Bereichen (z.B. Nahverkehr, Kartenmaterial), Gruppen-Bereichen (mehrere Personen mit unterschiedlichen Rollen, die auf gemeinsame Dienste zugreifen) und privaten Bereichen (eine Person, die vollen Zugriff auf die privaten Kontextdaten hat (Position, Konten, usw)).

Betrachtet man eine Domäne, so stellt sich heraus, dass innerhalb dieser Domäne das Kontextmodell weitgehend heterogen ist. Die Bedeutung des Konzeptes Ort über verschiedene Lernmanagementsysteme hinweg ist ähnlich. Zwischen den Domänen Nahverkehr und Lernmanagementsystem ist jedoch die Ausgestaltung des Konzeptes Ort verschieden. Es bietet sich daher an global einige Grundkonzepte vorzugeben, von denen aus die Domänen ihre eigenen speziellen Konzepte erstellen. Diese sind innerhalb einer Domäne homogen und lassen sich weiterverwenden, während sie zwischen den Domänen heterogen sind.

In den folgenden Abschnitten folgt die Erläuterung der konzipierten Knoten, was sie auszeichnet und welche mögliche Beispiele für eine Realisierung sind.

3.2.1 Private Knoten zur Dienstauswahl nach privaten Kontextdaten

Die Kontextdaten, die am besten geeignet sind, um die Situation eines Nutzers zu beschreiben, sind Informationen, die es erlauben ein komplexes Profil von diesem zu erstellen. Dazu gehören Daten über Identität, Position, Kontakte, Kommunikation, Termine und Zugehörigkeit zu Gruppen. Folglich sind diese Informationen auch am schützenswertesten.

Daher sollte der Nutzer stets die Kontrolle darüber haben, wo und wie diese Daten verwendet werden und sie auch selbstständig löschen können.

Das Ziel der privaten Knoten ist es ein System zu erstellen, das die Zugangsdaten des Nutzers verwendet, um aus anderen Systemen Informationen abzurufen. Das System selbst steht jedoch nicht für Anfragen von außen zur Verfügung. Somit werden private Dienste für den Nutzer von einem Knoten abgerufen, auf den nur er zugreifen kann. Die Analyse der Situation auf Basis der privaten Daten erfolgt somit nur auf dem System, das auch unter der persönlichen Kontrolle des Nutzers steht.

Dieser Knoten ist daher die Schnittstelle des Nutzers zu anderen Knoten, die Dienste von Domänen verwalten, in denen er Mitglied einer Gruppe ist oder die öffentlich sind. Die Verbindung wird jedoch nur hergestellt, wenn die nutzereigenen Regeln eine Situation erkannt haben, in der hier ein Bedarf an Informationen vorliegt.

Das Ziel sollte es sein, diesen Knoten auf einem persönlichen Gerät des Nutzers, wie einem privaten Server oder einem mobilen Gerät zu betreiben. Durch eine Anbindung an bereits existierende Datensammlungen, z.B. Kalender oder Sensoren (z.B. Positionssensor in der Smart Watch), ist die Möglichkeit gegeben Kommunikationsaufwand einzusparen. Konzep-

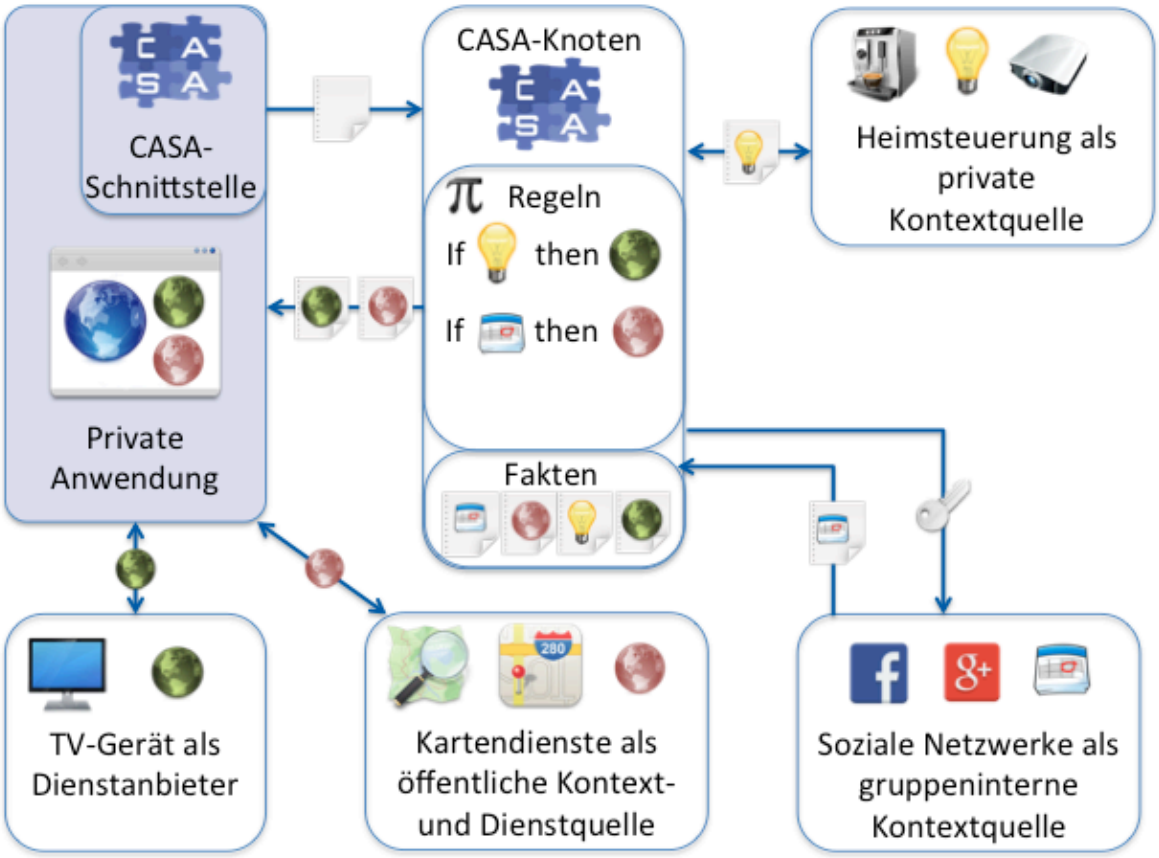

Abbildung 3.6: Privater Knoten mit Nutzung von internen und externen Diensten und Kontextquellen

tionell kann der private Knoten eines Nutzers auch auf einem entfernten System betrieben werden, jedoch sollte hier der Zugriff von außen gesichert sein.

Abbildung 3.6 illustriert einen privaten Knoten, der über die Zugangsdaten des Nutzers Zugriff auf dessen soziale Netzwerke nimmt, um dort Informationen zu Terminen zu ermitteln. Liegt ein Termin vor, werden dessen Informationen genutzt, um einen Kartendienst zu konfigurieren. Die Ausgabe erfolgt in der Anwendung des Nutzers, die zum Beispiel auf seinem mobilen Gerät laufen kann. Als zweite Funktion ist hier die Heimsteuerung als Kontextquelle angebunden, um bei Eintreten einer definierten Situation, hier das Einschalten einer Lampe, einen Informationsdienst auf dem TV-Gerät zu aktivieren.

3.2.2 Gruppen-Knoten zur Dienstauswahl in Gruppen

Die Situation eines Nutzers ist häufig bestimmt durch seine Zugehörigkeit zu Gruppen. Diese ordnen ihm eine Rolle innerhalb einer Menge von Nutzern zu, die in gleichen Prozessen aktiv sind oder gleiche Interessen haben. Typische Gruppen an Hochschulen sind Teilnehmer einer Veranstaltung oder Mitarbeiter an einem Lehrstuhl. Da die Mitglieder einer Gruppe häufig gleiche Arbeitsabläufe (Prozesse) haben, können für sie auch die gleichen Dienste relevant

sein. Daher ist es sinnvoll diese Schnittstelle als einen Knoten zu bündeln, der die Identität des Nutzers als Mitglied der Gruppe bestätigt und auf dieser Grundlage relevante Dienste anbietet. Des Weiteren ist es sinnvoll dem Nutzer die Möglichkeit zu geben, Dienstvorschläge und Situationsbeschreibungen auch mit anderen Mitgliedern seine Gruppe zu teilen.

Diese Knoten sind an Systeme angebunden, die einer Gruppe, wie zum Beispiel einem Lehrstuhl, direkt zugeordnet sind. Alternativ können die Systeme auch verschiedene Gruppen verwalten, so wie bei einem Lernmanagementsystem. Der Zugriff durch den Nutzer kann dabei entweder über seinen privaten Knoten erfolgen, der Dienstanfragen stellt, oder durch eine Schnittstelle in dem Wirtssystem, das über ein Plugin den Nutzern Zugriff auf die von CASA angebotenen Dienste bietet. Dabei kann der Knoten, der zu diesem System gehört, nur über das Wissen verfügen, das bereits in den Daten des Wirtssystems vorhanden oder das öffentlich verfügbar ist. Es besteht zum Schutz der privaten Daten nicht die Möglichkeit Informationen aus dem privaten Knoten abzufragen.

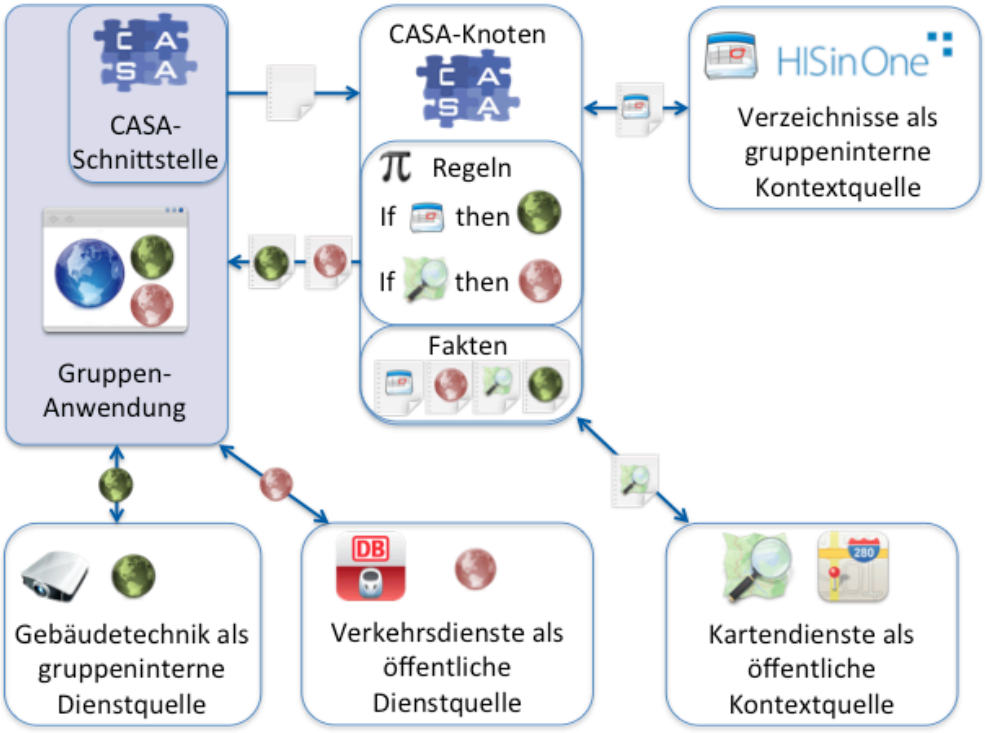

Abbildung 3.7: Gruppen-Knoten mit Nutzung von gruppeninternen und externen Informationsquellen und Einbindung von gruppenbeschränkten und öffentlichen Diensten

Des Weiteren ist es möglich, verschiedene CASA-Knoten an verschiedene Wirtssysteme zu koppeln, wenn die Nutzeridentifikation identisch ist. So könnten in einem Lernmanagementsystem personalisierte Dienste aus einer Unibibliothek angeboten werden, wenn die Identität des Nutzers gesichert ist.

3.2.3 Öffentliche Knoten zur personenunspezifischen Dienstauswahl

Viele Kontextinformationen, die genutzt werden können, um einem Nutzer einen Dienst zu empfehlen, sind öffentlich zugänglich und benötigen keine Zugangsdaten. Diese Daten können Informationen zum öffentlichen Nahverkehr, zu Sehenswürdigkeiten oder zum Wetter beinhalten.

Um diese Daten aufzubereiten und in dem hier verwendeten Fall zu nutzen, lässt sich ebenfalls ein CASA-Knoten anwenden. Der Knoten ist dabei für eine Domäne, wie zum Beispiel das Wetter, zuständig und bereitet die Daten eines Wetterinformationsdienstes für andere CASA-Knoten oder angeschlossene Webseiten auf. Dabei sind für die Kommunikation mit dem Knoten keine Zugangsdaten notwendig.

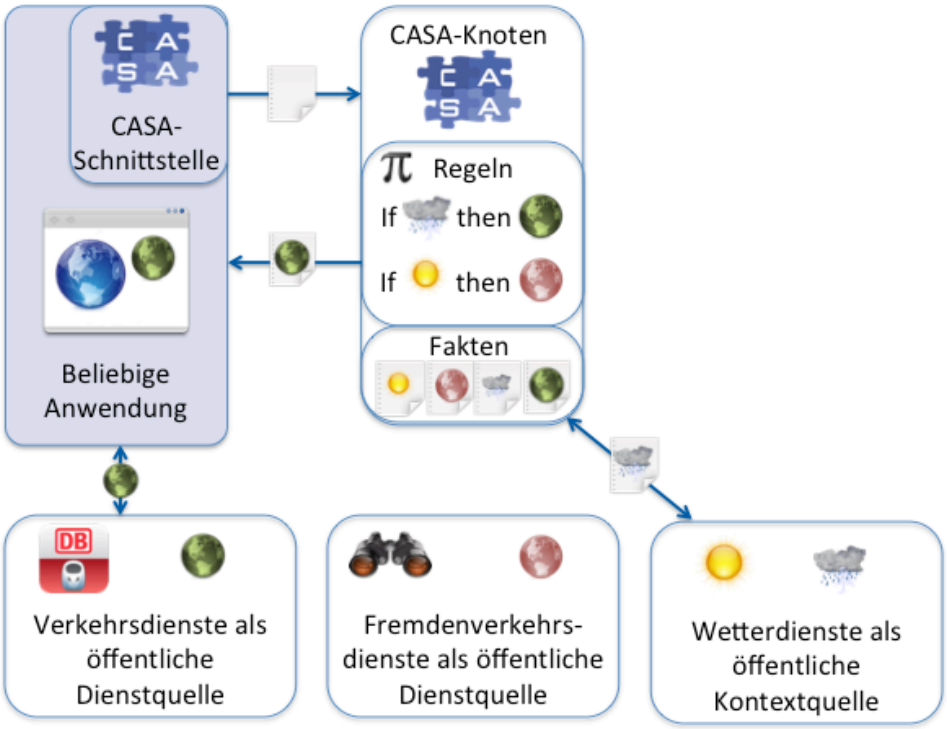

Abbildung 3.8: Öffentlicher Knoten mit Nutzung und Einbindung von ausschließlich öffentlich zugänglichen Informationsquellen und Diensten

So ein öffentlicher Knoten kann von jeder Person erstellt werden und es ist nicht notwendig, dass die betreibende Gruppe oder Person einen Bezug zu den verwendeten Daten hat. Für die Korrektheit der Daten, die der Knoten liefert, ist der Betreiber des Knotens verantwortlich, ebenso wie für die Frage, ob und wie die Daten vom öffentlich zugänglichen Informationsanbieter weiterverwendet werden dürfen.

Die Einbindung dieser Knoten kann auch in Webseiten erfolgen, jedoch verwenden diese Knoten keine nutzerspezifischen Daten und so sind die empfohlenen Dienste nutzerunspezifisch. Sie wären geeignet, wenn es um die Einbindung einer Regenvorhersage für einen Veranstaltungsort oder um die Nahverkehrsverbindungen an einem Ort ginge.

3.3 Organisation der CASA-Knoten

Unter dem Fokus der Organisation der Knoten werden im Folgenden die Abläufe zwischen den Knoten erläutert. Dabei ist es das Ziel, dass die Knoten trotz ihrer Domänenspezifik in der Lage sind über die verschiedenen Domänen hinweg zu kommunizieren. Des Weiteren sollen die Knoten sowie die durch sie erweiterten Anwendungen die verschiedenen Zugriffsrechte der Nutzer anwenden können, um so auch komplexe Anfragen an das Netz der CASA-Knoten zu stellen, ohne jedoch direkten Zugriff auf die Dienste zu haben.

Im Folgenden wird daher zuerst erläutert, wie mittels einer Kaskadierung der Knoten die Dienste für den Nutzer und seine Situation angepasst werden können. Danach wird aufgezeigt, wie die Kommunikation zwischen den Knoten selbst abläuft. Abschließend wird erläutert, wie Orchestrierung genutzt werden kann, um innerhalb eines Netzes von Knoten komplexe Anfragen zu bearbeiten.

3.3.1 Dienstabdeckung durch kaskadierende Dienstauswahl

Um eine Dienstauswahl über mehrere Domänen hinweg zu gewährleisten, nutzt CASA das Konzept der Kaskadierung von Knoten. Darunter wird eine einseitig gerichtete Verknüpfung mehrerer Knoten verstanden, wobei die Anfragen stets von den geschützteren Knoten zu den weniger geschützten Knoten gerichtet sind.

Dieses Prinzip setzt voraus, dass der Nutzer mit seinem Knoten weitere Knoten verknüpft hat, in denen er Mitglied von Gruppen ist oder die für seine hinterlegten Regeln öffentlich verfügbare Informationen bereitstellen können. Dazu stellt er in seinem privaten Knoten eine Dienstanfrage, die dieser Knoten zuerst selbst beantwortet und dann an die verknüpften Knoten weitergibt. In Abbildung 3.9 ist beispielhaft ein CASA-Netz dargestellt, das aus verschiedenen Knotenarten besteht.

Wie dargestellt evaluieren die verbundenen Knoten die Situation des Nutzers auf Basis der ihnen zur Verfügung stehenden Daten und nutzen dabei, ebenso wie der private Knoten, auch öffentlich zugängliche Informationen sowohl direkt über ihre eigenen Importer als auch über öffentliche Knoten. Die Informationen zu den Diensten, die von den Gruppen-Knoten als relevant identifiziert werden, werden an den privaten Knoten weitergeleitet. Dieser ist nun in der Lage auf Basis seiner eigenen Regeln und Filter die Dienstkandidaten auszuwählen, die für die Situation des Nutzers relevant sind.

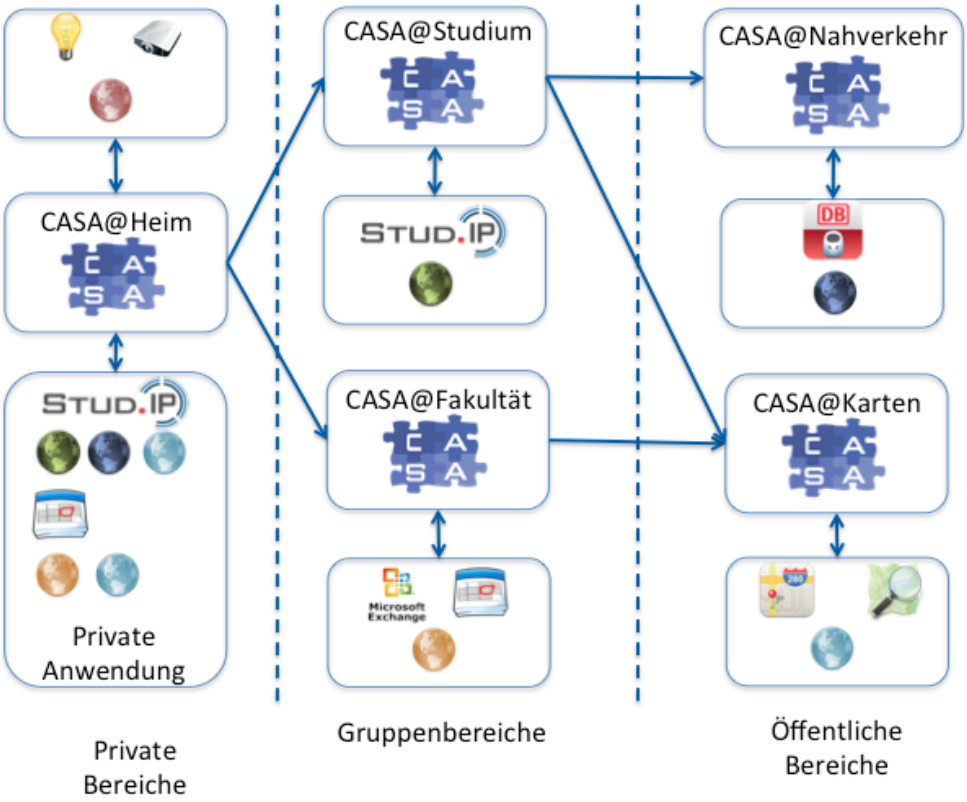

Abbildung 3.9: CASA-Netz mit einem privaten Knoten, zwei Gruppen-Knoten und zwei öffentlichen Knoten

3.3.2 Kommunikation zwischen Knoten

Die Kommunikation, die zwischen verschiedenen Knoten abläuft, orientiert sich ebenso wie die Kommunikation eines Knotens mit dem Plugin und den Dienstanbietern an dem Modell einer Service-orientierten Architektur [Melzer 07]. Dabei lässt sich der Knoten, von dem die erste Anfrage ausgeht, als der Broker sehen, der bei einer Anfrage sein Wissen über andere Knoten nutzt, um diese nach weiteren Diensten anzufragen. Der Dienstnutzer bleibt dabei das Plugin, wobei sich jedoch die anderen Broker und die von ihnen verwalteten Dienstanbieter zu einem komplexen Dienstanbieter zusammenfassen lassen.

Die Anfrage an andere CASA-Knoten wird über Regeln in Form von Aktionen definiert. Somit ist es dem Nutzer möglich, Kontrolle über die von ihm in Form von Anfragen ausgehenden Informationen zu behalten. Er kann selbstständig festlegen, wie sparsam er mit der Freigabe seiner Daten umgehen möchte.

3.3.3 Orchestrierung von Knoten

Die Nutzung verschiedener CASA-Knoten für eine Dienstauswahl kann diese verfeinern. So kann, wie in Abbildung 3.10 dargestellt, ein CASA-Knoten mit anderen Knoten verbunden werden. Dabei ist die Dienstauswahl abhängig von den genutzten Kontextquellen. Eine Integration mit Kontextinformationen ohne Bezug zur aktuellen Situation des Nutzers führt zu einer nutzerunspezifischen Dienstauswahl.

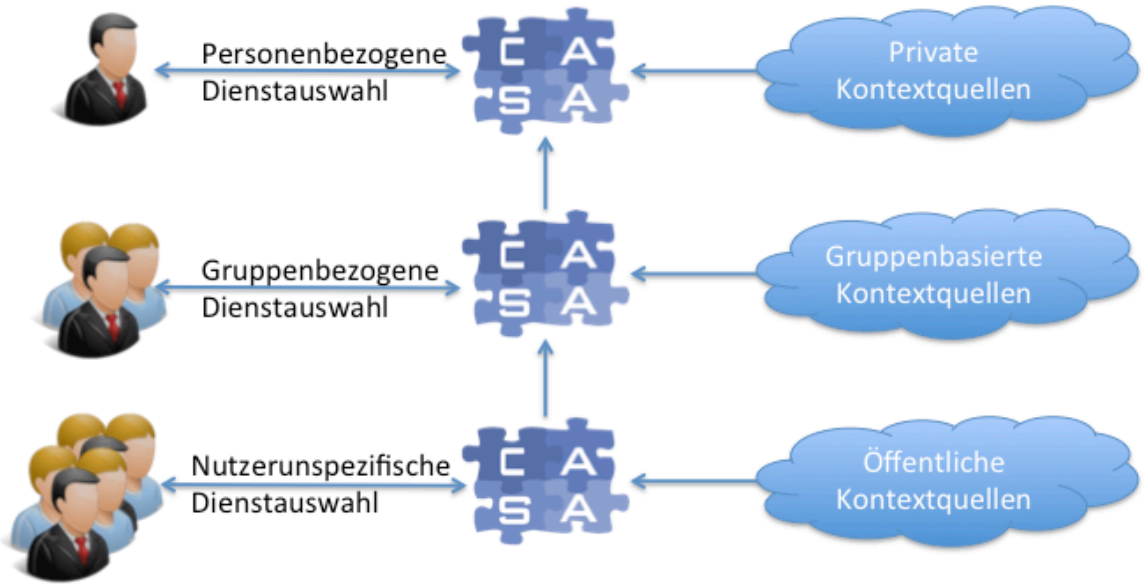

Abbildung 3.10: Informationsfluss zwischen privaten und öffentlichen Knoten

Erfolgt die Integration mit Hilfe von gruppenbasierten Kontextquellen, wie zum Beispiel Gruppen-Kalendern, ist die Auswahl bereits gruppenspezifisch. Die von dem öffentlich genutzten Knoten bereits normalisierten Kontextinformationen können hier durch den Knoten angefragt werden, der durch die Gruppe genutzt wird. Die Informationen aus dem Gruppen-Knoten stehen dem öffentlichen Knoten jedoch nicht zur Verfügung. Analog verhält es sich zwischen einem privat genutzten und einem durch eine Gruppe genutzten Knoten. So kann gewährleistet werden, dass Informationen aus dem privaten oder gruppenorientierten Bereich diese nicht in Richtung weniger geschützter Bereiche verlassen. Neben den Kontextinformationen können so auch Dienstvorschläge aus verschiedenen Knoten gesammelt für einen Nutzer angeboten werden.

3.4 Der CASA-Knoten und seine Funktionen

Im folgenden Abschnitt wird der CASA-Knoten beschrieben. Er entspricht der modularen Grundeinheit des Systems. An diese Einheit können andere Knoten angeschlossen werden, um aus ihnen ein Netz zu bilden. Innerhalb dieser Einheit können verschiedene Module verwendet werden, um Sensoren einzubinden oder Daten weiterzuverbreiten.

Um diese Einheit genauer zu beschreiben, wurden im Folgenden zuerst der grundlegende Aufbau erläutert und anschließend die einzelnen Aspekte innerhalb des Knotens betrachtet. Dies beginnt mit den Regeln und Aktionen, die ein Knoten ausführen kann. Danach werden die Dienste betrachtet, die in diesem Zusammenhang verwaltet werden. Anschließend wird der Aspekt der Aggregation und die Integration von Informationen und Diensten durch den Knoten erläutert. Danach wird gezeigt, wie durch Importer die Kontexterfassung realisiert wird.

3.4.1 Architektur und Organisation des CASA-Knotens

Der CASA-Knoten ist die zentrale Einheit zur Transformation und Verarbeitung von Kontextinformationen. Um diese Einheit an die verschiedenen Domänen anpassbar zu machen, muss sie modular gestaltet sein. In Abbildung 3.11 ist ein Aufbau des Knotens mit den verschiedenen Schichten und Modulgruppen gegeben. Zusätzlich ist in der Abbildung eine Zuordnung der CASA-Komponenten zu den Schichten eines kontextorientierten Systems (Erfassung, Transformation, Nutzung des Kontextes) gegeben. Des Weiteren ist dargestellt, welche Komponenten mit der Erfassung und Verarbeitung welcher Arten von Kontext (roh, normalisiert, komplex) befasst sind.

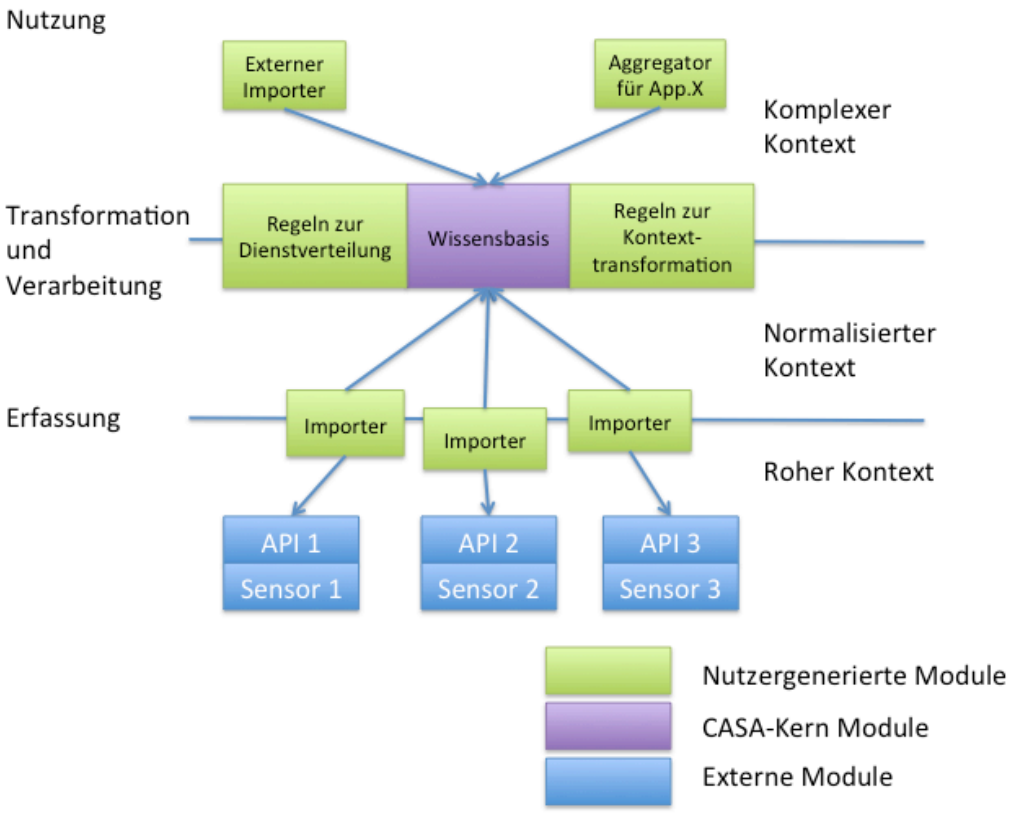

Abbildung 3.11: Architektur eines CASA-Knotens mit den verschiedenen Modulen

Beginnend mit den Quellen der Sensordaten, stellen die Importer die erste Modulgruppe dar. Sie dienen als Schnittstelle zu den Sensoren, unabhängig davon, ob sie physisch, logisch oder virtuell sind. Des Weiteren dienen sie der Umwandlung der rohen Kontextinformationen in einen normierten, niedrig gradigen Kontext (low-level). Genauer wird auf die Importer in Abschnitt 3.4.5 eingegangen.

Die Anfragen an die Importer werden vom Herz des CASA-Knotens, dem wissensbasierten System, ausgegeben. Dieses kann beispielsweise durch ein regelbasiertes System bzw. durch ein Expertensystem umgesetzt werden. Dieses Modul ist für die Haltung verschiedener Daten verantwortlich. Diese umfassen die domänenspezifischen Regeln, die zur Steuerung der Dienstempfehlung genutzt werden, ebenso wie die Regeln, die die Transformation der Kontextinformationen zu komplexen Kontextinformationen realisieren. Diese Informationen werden als Fakten in der Logik gespeichert, da sie notwendig sind, um die Regeln zu evaluieren. Schließlich werden in diesem Modul auch die Dienstbeschreibungen und Aktionen gespeichert, die auf Basis der Regeln empfohlen und ausgelöst werden. In Abschnitt 3.4.2 wird genauer auf die Aktionen und Regeln eingegangen.

Der CASA-Knoten selbst bietet keine Dienste für den Nutzer an. Die Dienste werden lediglich durch ihre Beschreibungen verwaltet und als Fakten innerhalb der Wissensbasis genutzt. Sie stellen somit kein separates Modul dar und sind eine Untergruppe der Aktionen. Die verschiedenen Arten von Diensten werden in Abschnitt 3.4.3 erläutert.

Am Ende der Kontextverarbeitung stehen die Module zur Integration und Aggregation. Sie stellen die Verbraucher innerhalb des CASA-Knotens dar. Sie erstellen Anfragen an das wissensbasierte System nach Dienstempfehlungen oder auch nach komplexen Kontextinformationen, die in anderen CASA-Knoten verwendet werden sollen. Für beide Fälle ist der Schutz der Informationen Teil der Aufgaben des CASA-Knotens, was hier eine Einbindung in existierende Authentifizierungs- und Autorisierungssysteme erforderlich machen kann. Genauer wird dies im Abschnitt 3.4.4 beschrieben.

3.4.2 Regeln und Aktionen im CASA-Knoten

Ein CASA-Knoten wird durch seine Kontextdefinitionen und die darauf basierenden Regeln für eine Domäne spezifiziert. Dabei können die Regeln und die Regelmenge durch den Nutzer erweitert und verändert werden. Unter einer Regel wird hierbei der erste Teil einer typischen, bedingten Anweisung verstanden, wie sie als If-Then-Konstruktion in der Informatik bekannt ist. Der zweite Teil (Then) wird im Folgenden als Aktion betrachtet, die durch die Erfüllung der Bedingungen der Regel ausgeführt wird. Eine Regel beschreibt dabei stets eine Situation, die durch verschiedene Kontextinformationen charakterisiert wird. Formal untersucht also die Menge aller Regeln in einem Knoten die vorliegenden Kontextinformationen auf definierte Situationen, denen Aktionen zugeordnet sind. Des Weiteren kann den Regeln ein Rang zugeordnet werden, der die Relevanz der hervorgerufenen Aktion bestimmt.

Definition 7 (Regeln $\mathscr{S}_{\mathscr{C}}$) Eine Regel bildet den vorliegenden Kontext $c \in \mathscr{C}$ auf eine endliche Menge von Paaren (a, p) mit den Aktionen $a \in \mathscr{A}$ und dem Rang $p \in \mathbb{R}$ ab. Mit $\mathscr{S}_{\mathscr{C}}$ wird die Menge aller Regeln innerhalb eines Knotens beschrieben.:

$$\begin{aligned} \mathscr{S}_{\mathscr{C}} : \mathscr{C} &\to \mathscr{P}(\mathscr{A} \times \mathbb{R}) \\ c &\mapsto \{(a_1, p_1), (a_2, p_2), \ldots, (a_n, p_n)\} \end{aligned}$$

Regeln können in verschiedene Klassen unterteilt werden. Entsprechend des Ziels ihrer Aktionen kann zwischen Transformationsregeln und Interaktionsregeln unterschieden werden. Transformationsregeln verändern die Wissensbasis des Systems und können neue Fakten einfügen sowie vorhandene Fakten ändern oder löschen. Diese Regeln dienen der Kontexttransformation und werden genutzt, um Kontextinformationen, die durch die Importer in die Wissensbasis eingespeist wurden, in normalisierten Kontext zu überführen und für andere Regeln nutzbar zu machen. Des Weiteren können sie die vorliegenden Informationen auf besondere Situationen untersuchen, die Eigenschaften anderer mit diesen Fakten verbundenen Entitäten ändern oder nicht relevante Fakten verwerfen.

Interaktionsregeln haben Aktionen, die nach außen gerichtet sind. Sie dienen der Beantwortung von Anfragen, die von außerhalb des Knotens kommen oder fordern neue Kontextinformationen von Importern an. Zu den Aktionen, die sie auslösen können, gehört auch die Empfehlung von Diensten oder das Ausführen von Programmen.

Der Zeitpunkt der Evaluation der Regeln kann auch genutzt werden, um die Regeln zu unterscheiden. So liegt bei den Transformationsregeln meist ein reaktives Verhalten vor, da diese durch das Einfügen neuer Fakten durch einen Importer angestoßen werden, also durch einen äußeren Auslöser. Abgesehen davon ist jedoch auch ein proaktives Verhalten möglich, bei dem die Regeln jederzeit die aktuellen Kontextinformationen auf das Vorhandensein von Situationen überprüfen. Diese Regeln können genutzt werden, um Aktionen auszulösen, wenn ein Nutzer sich an einem vorher definierten Ort aufhält. Dabei liegt hier durch den Nutzer keine direkte Anfrage vor, sondern es wird eine indirekte Anfrage genutzt, die der Nutzer als Regel definiert hat.

Die Formulierung der Regeln durch den Nutzer erfordert die Kenntnis der vorliegenden Domäne und des verwendeten Kontextmodells. Abseits der domänenspezifischen Regeln ist es jedoch auch möglich auf Basis des allgemeinen Kontextmodells Regeln zu definieren, die domänenübergreifend verwendbar sind.

Eine Herausforderung in diesem Zusammenhang ist die Tatsache, dass die Regeln widerspruchsfrei sein müssen, um das System nicht in einen unsicheren Zustand zu bringen. Der Einsatz von oszillierenden oder unerfüllbaren Regeln kann zu Fehlern führen. Ein möglicher Ansatz zur Sicherstellung der Konsitenz kann hier der Einsatz von Testdaten und Sandboxes sein. Des Weiteren kann auch der Nutzer eingebunden werden, um fehlerhafte Regeln zu melden und zu korrigieren. Die genauere Untersuchung von Methoden, um Fehler zu verhindern, ist jedoch nicht Ziel dieser Arbeit.

3.4.3 Dienste im CASA-Knoten

Die Dienstbeschreibungen liegen im CASA-Knoten als Fakten vor. Die Empfehlung und Konfiguration des Dienstes erfolgt durch eine Aktion, die wiederum Teil einer Regel ist. So kann ein Dienst, der von einer externen Webseite angeboten wird, durch die Aktion in der Regel eines CASA-Knotens konfiguriert werden. In der vorliegenden Implementierung von CASA werden diese Regeln in DRL, der Drools Rule Language, implementiert [Salatino 16].

```
1
2  rule "MensaService"
3    when
4      r : Request()
5      exists($p1 : Position() from r.getConditions() and eval(closeTo($p1,
           new Position(54.0748861, 12.1161766),1000d)))
6    then
7      Website s = new Website();
8      s.setTitle("Mensa Speiseplan heute");
9      s.setTargetURL("http://www.studentenwerk-rostock.de/index.php
10             ?lang=de&mainmenue=4&submenue=47");
11     s.setSource("MensaService");
12     ArrayList<Entity> results = r.getResults();
13     results.add(s);
14     r.setResults(results);
15 end
```

Quelltext 3.1: Regel, die einem Nutzer in der Nähe der Rostocker Mensa den Speiseplan als Dienst anbietet

Als Beispiel sei hier ein Dienst gegeben, der den aktuellen Speiseplan der Rostocker Mensa anzeigt. Wie in Quelltext 3.1 dargestellt, sind die Position der Mensa als GPS-Koordinate sowie die URL Teile der Regel.

Basis der Entscheidung diesen Dienst zu empfehlen ist das Request-Objekt, das als Fakt in die Wissensbasis aufgenommen wird. Anschließend wird geprüft ob die Bedingungen zum Auslösen dieser Regel erfüllt sind, in diesem Fall ob bei den Bedingungen des Requests eine Position enthalten war, die in der Nähe der Mensa ist (s. Zeile 5). Mittels dieser Bedingungen können belibige Fakten beigefügt werden, die dann für die Konfiguration der Dienste genutzt werden können.

Somit ist es bereits dem Knoten, der den Dienst als erstes empfiehlt, möglich sein Wissen zu nutzen, um den Dienst zu konfigurieren. Die Knoten, die den Dienst einbinden, müssen nicht über das semantische Wissen verfügen um den Dienst inhaltlich zu konfigurieren, sondern nur über das Wissen, das notwendig ist, um ihn in die jeweilige Anwendung zu integrieren.

3.4.4 Aggregation und Integration der Dienste

Wie bereits in Abschnitt 3.4.1 erläutert ist es notwendig, dass die Dienste, die für eine Nutzersituation relevant sind, aggregiert und in die jeweilige Repräsentation integriert werden. Dies erfolgt durch Module, die anwendungsspezifisch erstellt werden und jeweils in die Anwendungen eingebunden sind, mit denen der Nutzer interagiert. Dabei verantworten die Module die Transformation der Dienstbeschreibungen in die für die Anwendung notwendigen Zielformate. Des Weiteren sind sie dafür verantwortlich, die Dienste zu gruppieren und in eine Ordnung zu bringen.

Formal existiert eine Repräsentation $\mathscr{R}$, die mit einem Kompositionsoperator $\circ : \mathscr{R} \times \mathscr{R} \to \mathscr{R}$ die verschiedenen Repräsentationen der Dienste in einer Repräsentation zusammenführt. Des Weiteren wird davon ausgegangen, dass es eine Repräsentationsfunktion $rep_a : \mathbb{R} \to \mathscr{R}$ zu jeder Aktion a gibt. Sie gibt eine Repräsentation $r_a \in \mathscr{R}$ von a unter Berücksichtigung einer gegebenen Präferenz zurück. Die Menge aller relevanten Aktionen in Bezug auf einen gegebenen Kontext c, die auch lokal relevant sind, ist daher:

$$ap(c) := \bigcup_{\substack{s \in \mathscr{S}_{\mathscr{C}}, \text{ und} \\ s \text{ ist lokal relevant zu } c}} s(c)$$

Die Menge $ap(c)$ enthält damit alle Aktionen-Präferenz-Paare, die sowohl relevant zum Ort als auch zum restlichen, aktuellen Kontext c sind. Anschließend lässt sich die Menge $ap(c)$ nach der Präferenz sortieren, so dass eine Liste $ap'(c)$ entsteht:

$$ap'(c) := \left[(a,p) \in ap(c), \text{ sortiert nach } p\right]$$

Das Ergebnis der Anwendung von CASA lässt sich also wie folgt ausdrücken:

$$\begin{aligned} \text{CASA} : \mathscr{C} &\to \mathscr{R} \\ c &\mapsto \circ_{(a,p) \in ap'(c)} rep_a(p) \end{aligned}$$

Alle möglichen Aktionen-Präferenz-Paare (ap) werden ermittelt und entsprechend ihrer Präferenz sortiert (ap'). Für jede Aktion a wird anschließend eine geeignete Repräsentation entsprechend der jeweiligen Präferenz p erstellt. Dies erfolgt mittels der Berechnung von $rep_a(p)$ für jedes Aktions-Präferenz-Paar (ap). Abschließend werden diese separaten Repräsentationen durch den Kompositionsoperator $\circ$ zu einer finalen Repräsentation zusammengeführt und dem Nutzer präsentiert.

3.4.5 Importer als Schnittstelle zu heterogenen Kontextquellen

Die Module, die zum initialen Erschließen des Kontextes verwendet werden, werden im CASA-System als Importer bezeichnet. Sie dienen als Schnittstelle zu den Kontextquellen und damit zu den Sensoren. Diese stellen ihre Informationen proprietär in einem beliebigen Format zur

Verfügung und erfordern es daher, dass die Importer dieses Format einerseits verstehen und andererseits in der Lage sind, es in ein normalisiertes und für den jeweiligen CASA-Knoten verständliches Format umzuwandeln. Somit ist es bereits hier möglich die Daten an der Quelle zusammenzufassen und an das Kontextmodell anzupassen und damit den Kommunikationsaufwand zu verringern.

Die Importer sind dabei in jedem Fall sensor- und eventuell auch schon domänenspezifisch. Für eine gute Wiederverwendbarkeit empfiehlt es sich an dieser Stelle Importer zu koppeln, so dass der sensorspezifische Teilimporter wiederverwendet werden kann, während der domänenspezifische Teilimporter vom Umfang her kompakt bleibt. Alternativ kann der domänenspezifische Transformationsvorgang auch im CASA-Knoten durch Transformationsregeln erfolgen.

3.5 Nutzereinbindung im CASA-Konzept

Dem Nutzer bei seinen Aufgaben behilflich zu sein erfordert auch, ihn bei der Nutzung des CASA-Systems zu unterstützen. Dazu muss zuerst geklärt werden, welche Interaktionsmöglichkeiten dem Nutzer offen stehen und wie sich eine komfortable Interaktion mit dem Nutzer realisieren lässt, die ihn nicht überfordert. Dazu wird in Abschnitt 3.5.1 dargelegt, wie die Nutzer in Nutzungsprofile aufgeteilt werden können und welche Aktionen den verschiedenen Profilen zugeordnet werden.

Die Unterstützung des Nutzers selbst bei der Verwendung des Systems kann auf verschiedene Weise erfolgen. Betrachtet werden im Folgenden zwei Möglichkeiten, einerseits eine direkte Unterstützung durch die Verwendung von domänenspezifischen Sprachen (DSL), die genutzt werden können, um die Formulierung von Situationen und Regeln zu vereinfachen (Abschnitt 3.5.2). Andererseits kann man den Nutzer auch indirekt unterstützen, indem man ihn motiviert, sich intensiv mit dem System zu befassen. Dies kann zum Beispiel durch Gamification erfolgen (Abschnitt 3.5.3).

3.5.1 Nutzerinteraktionen

Um die Umsetzung dieser Anforderungen zu unterstützen, wurde eine Matrix mit den verschiedenen Nutzergruppen und den Nutzungsvarianten erarbeitet. Dies erlaubt eine Einordnung des Nutzers und eine Folgerung, welche Funktionalitäten für ihn relevant sind. Zum besseren Verständnis wird stets ein Vergleich mit ähnlichen Nutzertypen bei der Wikipedia herangezogen.

Gelegenheitsnutzer
: Dieser Nutzer zeichnet sich dadurch aus, dass er das System in erster Linie für seine persönlichen Zwecke nutzt. Er verwendet das System so, wie es ist, und konfiguriert höchstens fertige Situationen und Dienste, um das System seinen Ansprüchen anzupassen. Er steht nicht im Kontakt mit anderen Nutzern und agiert nach außen anonym im System. Vergleichbar ist dieser Nutzer mit einem Wikipedia-Nutzer, der in erster Linie

Artikel konsumiert und an den Artikeln höchstens grammatikalische oder orthographische Korrekturen vornimmt.

Fortgeschrittener Nutzer
: Als fortgeschrittener Nutzer hat er ein Grundverständnis über die Struktur des Systems und seine Funktionsweise. Dieser Nutzer wirkt aktiv bei der Weiterentwicklung des Systems für eine oder mehrere Domänen mit. Dabei liegt sein Ziel in der Optimierung des Systems für sich selbst und andere Nutzer. Seine Aktivitäten sind weitreichender, weshalb eine Authentifizierung und Protokollierung notwendig ist, um sicherzustellen, dass er keinen Missbrauch des Systems betreibt. Dies kann über ein Versionsverwaltungssystem realisiert werden. Ferner geht er auf Vorschläge anderer Nutzer ein und kann existierende Inhalte und Situationen so abändern, dass sie den neuen Anforderungen entsprechen oder auch neue Elemente in das System einfügen. Ihm obliegt die inhaltliche Verantwortung für das System. Diese Nutzer sind vergleichbar mit dem Autoren der Wikipedia. Sie schreiben aktiv neue Artikel und passen existierende Artikel inhaltlich an. Außerdem können sie bereits existierende Artikel löschen oder die Änderungen anderer Nutzer rückgängig machen. Daher ist hier eine Authentifizierung und Protokollierung notwendig.

Experte
: Dieser Nutzer hat ein tiefgreifendes Verständnis der Funktionsweise des Systems und der inneren Prozesse. Er erweitert das System auf grundlegender Ebene. Dies schließt das Kontextmodell ein und auch die Erstellung von neuen Schnittstellen oder die Anbindung weiterer Kontextquellen. Er kann auch für den Betrieb eines Systems verantwortlich sein, z.B. als Administrator innerhalb einer Fakultät. Der Vergleich mit der Wikipedia fällt hier schwer, da die Wikipedia eine zentrale Anlaufstelle darstellt und nicht wie das CASA-System über eine dynamische Verbindung verschiedener Domänen-Knoten funktioniert. Die Rolle des Experten wird bei der Wikipedia durch Server-Administratoren und Software-Entwickler wahrgenommen, die für den Betrieb und die technische Weiterentwicklung des Systems verantwortlich sind.

In der folgenden Tabelle 3.1 sind die vorgestellten Nutzerprofile mit ihren Kompetenzen dargestellt. Dabei bezieht sich die Sicht der Tabelle auf die Nutzung eines Gruppen- oder öffentlichen Knotens. Für einen von einem Nutzer privat verwendeten Knoten ergibt sich, dass der Nutzer hier auch Elemente löschen und Änderungen nicht nur vorschlagen, sondern auch direkt durchführen kann.

Um diese Form der Nutzerinteraktionen umzusetzen, empfiehlt sich eine graphische Oberfläche, die dem Nutzer bereits aus anderen Systemen bekannt ist. So wäre es möglich, den Prozess an der Wikipedia zu orientieren. Hier ist bei jedem Artikel ein Diskussionsbereich verfügbar, in dem die Nutzer die Inhalte kommentieren und auf Fehlstellen oder Fehler hinweisen können. Parallel dazu ist es auch bei den meisten Artikeln jederzeit möglich, kleinere Fehler direkt durch eine Bearbeitung zu korrigieren. Lediglich das Löschen von Artikeln wird über einen separaten Diskussionsbereich abgestimmt. Entscheidend ist dabei die Versionskontrolle, die es in der Wikipedia ermöglicht, Änderungen nachzuverfolgen und auch fehlerhafte Bearbeitungen schnell rückgängig zu machen.

Bereich	Kontextquellen					Situationen					Dienste				
	Nutzen	Fordern/Vorschlagen	Erstellen	Bearbeiten	Löschen	Nutzen	Fordern/Vorschlagen	Erstellen	Bearbeiten	Löschen	Nutzen	Fordern/Vorschlagen	Erstellen	Bearbeiten	Löschen
Casual User	x	x				x	x				x	x			
Advanced User	x	x				x	x	x	x	x	x	x	x	x	x
Expert User	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Tabelle 3.1: Zuordnung von Nutzerprofilen zu Interaktionen mit dem System

Für das in dieser Arbeit entwickelte System empfiehlt sich einerseits eine direkte Möglichkeit für jeden Nutzer, Vorschläge für neue Dienste oder Situationen zu machen und gleichzeitig für die fortgeschrittene Gruppe von Nutzern die Möglichkeit diese auch direkt umsetzen zu können. Parallel dazu sollten jedoch alle Änderungen durch eine Versionskontrolle oder zumindest ein Log mitgeschrieben werden. Für das Löschen von Inhalten empfiehlt es sich ebenfalls eine Instanz einzurichten, in der darüber abgestimmt werden kann oder zumindest eine Begründung gegeben werden muss. Eine Wiederherstellbarkeit aus der Versionshistorie sollte gewährleistet sein.

3.5.2 Einsatz von DSLs und Formularen

Bei der Einführung eines neuen Systems kann nicht davon ausgegangen werden, dass der Nutzer mit diesem sofort effizient arbeiten kann. Gerade wenn ein System ein tiefergehendes Verständnis, wie zum Beispiel die Kenntnis der Syntax und Semantik einer Regelsprache fordert, ist es sinnvoll auch Vereinfachungen zu diskutieren. Eine dieser Vereinfachungen ist die Nutzung von domänenspezifischen Sprachen [Fowler 10]. Diese erlauben eine Übersetzung von normalsprachlichen Sätzen in Regeln. Dazu werden verschiedene Sprachkonstrukte mit Hilfe regulärer Ausdrücke in die Regelsyntax übersetzt. Ihr Einsatz bietet sich besonders in Situationen an, in denen das Beherrschen der Regelsprache nicht vorausgesetzt werden kann und der Umfang sowie die Komplexität der möglichen Regeln nicht zu groß ist.

Alternativ oder ergänzend zum Einsatz von DSLs können auch Formulare genutzt werden, um den Nutzern die Definition von Regeln, Situationen und Diensten zu erleichtern. Diese können entweder über eine DSL-Konvertierung in eine Regel überführt werden oder auch direkt eine Regel erstellen. Des Weiteren lassen sich die DSLs auch nutzen, um dem Nutzer einmal definierte Regeln als normalsprachlichen Satz zur Kontrolle anzuzeigen. Damit hat er die Möglichkeit seine Eingaben selbst zu kontrollieren. Diese Aussage ist wiederum geeignet, um von anderen Nutzern bewertet zu werden, was die Motivation durch den Aspekt der sozialen Reputation stärken kann.

3.5.3 Motivation des Nutzers

Neben der Vereinfachung der Nutzung des Systems lässt sich auch aktiv auf den Nutzer einwirken und seine Motivation steigern. Eine Variante, die sich hierfür eignet, ist Gamification. Dies bezeichnet die Verwendung von spieltypischen Elementen in spielfremden Kontexten. Diese Elemente können zum Beispiel die Vergabe von Punkten oder das Erstellen einer Rangliste von Nutzern beinhalten. Weit verbreitet ist auch das Darstellen von Fortschrittsbalken oder die Vergabe von virtuellen Auszeichnungen (engl. badges). Ziel ist das Auslösen positiver Gefühle beim Nutzer während der Verwendung des Systems. So könnten Nutzer für die Interaktion mit dem System eine kleine Menge von Punkten bekommen und für die eigene Autorenschaft von Diensten wiederum eine große Menge. Dies würde es erlauben eine Rangliste zu erstellen, mit der Nutzer sich untereinander vergleichen können. Außerdem ließen sich so fortgeschrittene Nutzer identifizieren, die für Fragen und Vorschläge zur Verfügung stehen. Für bestimmte Aktionen können auch virtuelle Belohnungen vergeben werden, die es erlauben, dass die Nutzer einerseits direkt ein positives Feedback bekommen, wenn sie eine Interaktion durchgeführt haben, und andererseits ist es damit möglich, den Nutzer zu motivieren weitere Interaktionen durchzuführen, um zur nächstbesseren Auszeichnung zu kommen. Im Rahmen dieser Arbeit wurde das prototypisch umgesetzt und wird in Kapitel 4.4.1 genauer erläutert.

3.6 Rechtliche Aspekte des CASA-Konzeptes

Das CASA-Konzept berührt in der Realisierung verschiedene Rechte von Nutzern, Betreibern von Knoten und Dienstanbietern. Daher ist es notwendig an dieser Stelle einen Überblick über die Rechtsbereiche zu geben, die betroffen sind. Dieser Überblick stellt dabei keine abschließende rechtliche Betrachtung dar, sondern soll verschiedene rechtliche Aspekte der Arbeit hervorheben und diskutieren.

Der Abschnitt beginnt mit einem Einblick in die rechtlichen Implikationen für Nutzer der Dienstempfehlung sowie Betreiber von öffentlichen oder gruppenöffentichen Knoten. Abschließend wird ein Lizenzmodell für CASA vorgeschlagen, welches durch die Verwendung etablierter Lizenzen für die verschiedenen CASA-Bestandteile versucht, die Weiterentwicklung zu fördern.

3.6.1 Rechtliche Berührungspunkte durch die Dienstempfehlung

Urheberrecht

Es ist zu beachten, dass durch die Trennung der Knoten nach ihrem Grad an Privatsphäre (vgl. 3.2) für verschiedene Knotenarten verschiedene Aspekte des Urheberrechts (UrhG) zu berücksichtigen sind.

In einem privaten Knoten, dessen Dienstempfehlungen nur einer einzelnen Person zugänglich sind, kann somit davon ausgegangen werden, dass die Dienstempfehlung zum eigenen Ge-

brauch geschieht. Hier kann man nach § 53 Abs. 1-3 des Urheberrechts argumentieren, dass Kopien von urheberrechtlich geschütztem Material in den Bereich der Privatkopie fallen, da eine nicht gewerbliche und nicht öffentliche Nutzung erfolgt.

Macht nun eine Hochschule über CASA-Gruppen-Knoten urheberrechtlich geschütztes Material für eine begrenzte Menge von Studenten zugänglich, so kann man argumentieren, dass der § 52a des Urheberrechts, die sogenannte Intranetklausel, anwendbar ist. Hier ist jedoch darauf zu achten, dass nur Auszüge veröffentlicht werden, die einen kleinen Teil des Gesamtwerkes ausmachen[1].

Für alle CASA-Knoten und insbesondere für die öffentlichen gilt, dass das Recht verlangt, dass keine technischen Schutzmaßnahmen umgangen werden, um Inhalte fremder Seiten darzustellen. So ist zwar das Setzen von direkten Links auf Inhalte nach dem sogenannten Paperboy Urteil zulässig[2], jedoch dürfen zum Beispiel Startseiten nicht umgangen werden, wenn der Seitenbetreiber durch technische Methoden wie Session-IDs versucht, dies zu verhindern[3].

Werden fremde Inhalte nicht nur als Link empfohlen, sondern direkt eingebunden, wie es im CASA-Konzept vorgesehen ist, so stellt dies nach einem Beschluss des Europäischen Gerichtshofs keine Urheberrechtsverletzung dar[4]. Bedingung dafür ist jedoch, dass die Wiedergabe nicht für ein neues Publikum bestimmt ist und auch kein anderes technisches Verfahren verwendet wird.

Telemediengesetz und Rundfunkstaatsvertrag

Das Telemediengesetz (TMG) und der Rundfunkstaatsvertrag (RStV) sind ebenfalls relevant für die Betreiber von CASA-Knoten, denn sie enthalten die rechtlichen Bestimmungen zu Telemedien, die im Besonderen im Internet angebotene Dienstleistungen wie Webseiten umfassen [Krumme 17, Sjurts 17]. Die Regelungen gelten dabei unabhängig davon, ob mit der Dienstleistung ein kommerzielles Interesse verfolgt wird oder nicht.

So gilt für Webseiten, die ausschließlich privaten oder familiären Zwecken dienen, keine Impressumspflicht nach § 55 Abs. 1 RStV. Dies trifft auf private CASA-Knoten zu, da diese dem Konzept nach nur für den sie betreibenden Nutzer zugänglich sind. Für gruppenöffentliche und öffentliche Knoten gilt nach dem genannten Paragraphen eine eingeschränkte Impressumspflicht, wenn diese nicht gewerbsmäßigen Zwecken dienen und keine Inhalte anbieten, die in der Regel gegen Entgelt angeboten werden. Letzteres kann zu Abgrenzungsschwierigkeiten führen, weshalb es sicherer ist ein vollständiges Impressum zu hinterlegen, auch wenn man auf ein gewerbsmäßiges Interesse beim Betreiben des CASA-Knotens verzichtet.

Abschließend ergibt sich für die Betreiber eines CASA-Knotens aus § 13 Abs. 1 des TMG die Pflicht, den Nutzer zu Beginn eines Nutzungsvorganges über „Art, Umfang und Zwecke der Erhebung und Verwendung personenbezogener Daten [...] in allgemein verständlicher Form zu unterrichten“. Dies schließt im Falle von CASA im Besonderen die Vorgänge zur Erstellung

[1] Das Landgericht Stuttgart zog in seinem Urteil vom 27.09.2011 Az. 17 O 671/10 die Grenze bei 20%.
[2] Urteil des Bundesgerichtshofs vom 17.07.2003 - I ZR 259/00
[3] Urteil des Bundesgerichtshofs vom 29.04. 2010 - I ZR 39/08
[4] Beschluss des EuGH vom 21.10.2014 - C-348/13

von Regeln und zur Verknüpfung von Diensten durch den Nutzer ein, da hier nutzerbezogene Daten erfasst werden. Jedoch kann es auch bei CASA-Knoten, die ausschließlich generische Dienste anbieten, wie es bei Verkehrs- oder Wetterinformationssystemen der Fall ist, notwendig sein, eine Datenschutzerklärung für den Nutzer anzugeben, da auch die IP-Adresse des Nutzers als personenbezogenes Datum gilt.

Fazit

Inhalte, die bereits öffentlich legal und frei verfügbar sind, lassen sich in CASA legal einbinden. Für geschützte Inhalte sind besondere Pflichten zu beachten, die zum Beispiel verhindern, dass man sich einer Herkunftstäuschung (Plagiats) schuldig macht, indem man den Eindruck erweckt, es handle sich um ein eigenes Werk. Dies ist bei einer kommerziellen Nutzung unlauterer Wettbewerb. Bei eben dieser gewerblichen Nutzung ist auch zu beachten, dass man nicht gegen andere Gesetze, wie zum Beispiel das Leistungsschutzrecht für Presseverleger verstößt.

Des Weiteren trägt man als Autor eine Verantwortung für die Links, die man in CASA einstellt oder die man als Betreiber eines Knotens verbreitet. So darf nicht auf rechtswidrige Inhalte verlinkt werden. Dies sollte auch zumindest stichprobenartig durch den Betreiber eines CASA-Knotens kontrolliert werden. Sollte hier ein Verstoß bekannt werden, muss unverzüglich Abhilfe geschaffen werden. Außerdem sollte der Betreiber eines Knotens protokollieren, welche Inhalte von welcher Person verlinkt und verknüpft wurden, um hier gegebenenfalls auch die betreffende Person vom Knoten auszuschließen.

Es empfiehlt sich auch sichtbar einen Hinweis einzubringen oder zu verlinken, in dem klar angegeben wird, von wem die verknüpften Inhalte stammen, um so den Verdacht einer Zueigenmachung auszuschließen. Dies, in Verbindung mit einem Impressum und der Datenschutzerklärung, sollte einen Standard für gruppenöffentliche und öffentliche Knoten darstellen.

3.6.2 Lizenzen für die Nutzung, Verbreitung und Weiterentwicklung von CASA

Für die Realisierung des CASA-Konzeptes stellt sich die Frage nach den rechtlichen Rahmenbedingungen, unter denen das System veröffentlicht und verbreitet werden kann. Das deutsche Urheberrecht gewährt demjenigen, der ein Werk geschaffen hat auch das Recht zu entscheiden, wie dieses Werk genutzt, veröffentlicht und verändert werden darf (vgl. §§ 12, 15, 39 und 62 UrhG). Daher ist ein Werk ohne eine ausdrückliche Erlaubnis des Urhebers nicht zu verändern oder weiterzuentwickeln. Dies macht Überlegungen notwendig, unter welche Lizenz CASA gestellt werden kann.

Dabei steht nicht eine Kommerzialisierung im Fokus der Lizenzauswahl, sondern das Ziel, dass das System genutzt und weiterentwickelt werden kann. Eine closed-source Lizenz, also eine Lizenz, unter der der Quellcode nicht offengelegt wird, scheidet aus, da das System auf eine Beteiligung der Nutzer bei der Weiterentwicklung setzt. Dies allein setzt schon eine Verbreitung unter Weitergabe der Quellcodes voraus.

Diese Offenlegung der Softwarebestandteile kann jedoch nach verschiedenen Modellen und Lizenzen erfolgen. Hier ist zu diskutieren, ob CASA mit all seinen Bestandteilen als freie Software unter einer solchen Lizenz veröffentlicht werden sollte oder ob dies zu Konflikten mit dem Konzept selbst führen kann. So definiert die Free Software Foundation freie Software als eine, die dem Nutzer die folgenden vier Freiheiten gewährt:

> - The freedom to run the program as you wish, for any purpose (freedom 0).
> - The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1). Access to the source code is a precondition for this.
> - The freedom to redistribute copies so you can help your neighbor (freedom 2).
> - The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.
>
> (entnommen von [Free Software Foundation 17])

Diese Freiheiten decken sich mit der Idee von CASA, dass jeder Nutzer CASA-Knoten nutzen, betreiben und weiterentwickeln können soll. Einzig die Frage, wie weit die Freiheit der Weitergabe und Weiterverwertung gehen muss, ist hier zu diskutieren. In diesem Punkt sind die Lizenzen hauptsächlich nach der Stärke ihres Copyleft-Aspektes zu unterscheiden. Dieser beschreibt die Verpflichtung des Entwicklers eine Veränderung an einem geschützten Werk unter der gleichen oder einer kompatiblen Lizenz freizugeben. In Abbildung 3.12 sind verschiedene kompatible Lizenzen nach diesem Aspekt sortiert.

Bei freizügigen Lizenzen besteht diese Pflicht nicht und eine veränderte Bibliothek oder Anwendung kann auch unter einer proprietären Lizenz kommerzialisiert werden [IFROSS 17]. Lizenzen mit schwachem Copyleft enthalten Einschränkungen und können verlangen, dass die Veränderungen an dem Werk selbst freizugeben sind. Im Fall der LGPL bedeutet dies zum Beispiel, wenn in einem Java-Programm eine LGPL-lizensierte Bibliothek eingebunden wird, so muss die Lizenz des Programms es erlauben, dass die Bibliothek verändert und untersucht werden darf (z.B. durch Reverse Engineering). Der Quellcode des Programms, in dem die Bibliothek verwendet wird, muss selbst nicht veröffentlicht werden [Turner 04].

Den anspruchsvollsten Ansatz verfolgen Lizenzen mit strengem Copyleft. Hier sind nicht nur die Veränderungen am Werk freizugeben, sondern auch das davon abgeleitete Werk, was bei der GPL bedeuten kann, dass eine Anwendung, die eine unter GPL lizensierte Bibliothek nutzt, auch unter diese Lizenz gesetzt werden muss.

Für das CASA-System mit seinen Kernkomponenten ergibt sich hier die Lösung, dass nach Möglichkeit alles unter der Apache 2.0 Lizenz [The Apache Software Foundation 04] veröffentlicht wird. Dies erlaubt viele Freiheiten, wie mit dem System von Seiten der Nutzer und Entwickler umgegangen wird. So ermöglicht sie die Integration von einzelnen Bestandteilen in Systeme, deren Quellcodes nicht veröffentlicht werden. Die domänenspezifischen CASA-Kontextmodelle müssen somit nicht zwingend nach außen freigegeben werden. Bedingung

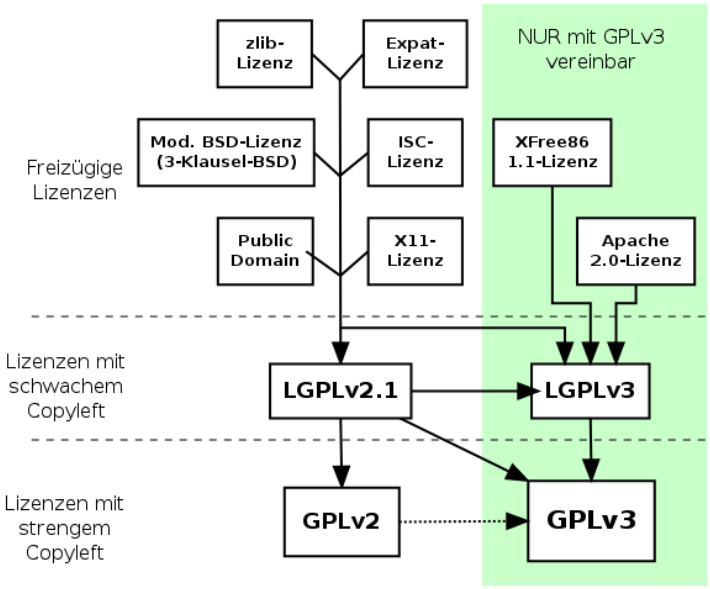

Abbildung 3.12: Übersicht über die Kompatibilität verschiedener Software Lizenzen. Entnommen aus [Smith 14]

dafür ist jedoch, dass alle der verwendeten externen Software-Bibliotheken dazu kompatibel sind.

Für die öffentlichen CASA-Knoten ergibt sich im Weiteren noch die Notwendigkeit, dass die hier verfügbaren Daten auch von anderen Quellen, im Speziellen von anderen CASA-Knoten, genutzt werden dürfen. Hier bietet sich eine Kennzeichnung als Open-Software-Service [Crossland 08] nach der Definition der Open Knowledge Foundation an. Diese legt fest, dass die Daten des Dienstes frei nach der Definition der von offenem Wissen verfügbar sein müssen[5]. Nutzerbezogene Daten sind dabei nicht öffentlich zu machen, sondern nur dem Nutzer zugänglich zu machen. Für gruppenöffentliche Knoten kann eine solche Regelung nicht getroffen werden, da dies dem CASA-Konzept widerspricht, das verlangt, dass diese Daten nicht öffentlich verfügbar gemacht werden sollen. Die Genehmigung zur Nutzung der Daten eines gruppenöffentlichen Knotens durch einen anderen oder einen privaten Knoten bleibt somit Bestandteil der direkten Absprachen zwischen den Betreibern des Knotens und den Nutzern.

Abschließend ist auch die Frage nach Haftung und Gewährleistung ein Grund für eine Lizenz der Software. Die Apache Lizenz schützt hier ebenfalls die Entwickler und damit auch die zuarbeitenden Nutzer vor Ansprüchen Dritter.

[5] http://opendefinition.org/od/1.1/de/ [Online letzter Zugriff 15.02.2018]

Kapitel 4

Implementierung des CASA-Konzeptes

Das im vorhergehenden Kapitel entwickelte CASA-Konzept wurde prototypisch implementiert, um die Machbarkeit zu zeigen und um der Evaluation zu dienen. Dabei wurde in einem ersten Schritt das CASA-System mit seinen Modulen umgesetzt, um die Funktionalität, Nutzbarkeit und Skalierbarkeit zu evaluieren. In einem zweiten Schritt wurde darauf aufbauend ein Gruppen-Knoten realisiert, der Kontextinformationen des Stud.IP nutzt und so eine komplexe anwendungsbezogene Implementierung darstellt.

In Abschnitt 4.1 wird die Struktur des allgemeinen CASA-Systems mit seinen Modulen, deren Aufgaben und den jeweiligen Verbindungen zwischen den Modulen erläutert. Es wird ebenso dargestellt, welche Werkzeuge und Softwareprodukte in den einzelnen Modulen genutzt werden. Abschnitt 4.2 beschreibt darauf aufbauend die einzelnen Module des Basissystems genauer in ihrer Funktion und Umsetzung und zeigt auf, wie ein CASA-Knoten generell erstellt und konfiguriert wird. Zur Demonstration eines anwendungsbezogenen Knotens wird in Abschnitt 4.3 die Umsetzung des Gruppen-Knotens für das Stud.IP erläutert. Dabei wird ausgeführt, welche Schritte notwendig sind, um eine domänen- und anwendungsspezifische Erweiterung zu entwickeln und diese auf dem Basissystem aufzusetzen.

Anschließend werden weitere Implementierungen erläutert, die auf dem CASA-System basieren und Anknüpfungspunkte für weitere Arbeiten liefern können. Dabei wird in Abschnitt 4.4.1 eine Gamification-Erweiterung erläutert, die über verschiedene Domänen und Anwendungen hinweg die Motivation des Nutzers zur Interaktion mit dem System steigern soll. Dies ist besonders relevant, da das Crowdsourcing für diese Arbeit auf der Motivation der Nutzer, mit anderen Dienste zu teilen, basiert. Daneben wurde eine Erweiterung umgesetzt, die sich speziell mit der Verknüpfung von Diensten mit Orten befasst und bereits existierende Verknüpfungen in anderen Systemen nutzt. Diese wird in Abschnitt 4.4.2 genauer beschrieben.

4.1 Struktur des CASA-Systems

Die Beschreibung der Struktur der Umsetzung des CASA-Systems erfolgt in zwei Schritten. Da sich die Struktur des Systems abhängig von seiner Funktion unterscheidet, wird zunächst die Systemarchitektur in den verschiedenen Varianten des CASA-Systems beschrieben und anschließend die Softwarearchitektur eines einzelnen Knotens.

Vorweg seien hier einige Begriffe und ihre Verwendung definiert.

CASA-Netzwerk
: Das CASA-Netzwerk bezeichnet eine Implementierung des im Konzept dieser Arbeit beschriebenen kontextbewussten und nutzergetriebenen Dienstempfehlungssystems. Es geht über den einzelnen CASA-Knoten hinaus und umfasst auch Komponenten, die sich außerhalb des Knotens befinden sowie andere verbundene CASA-Knoten.

CASA-Knoten
: Ein CASA-Knoten umfasst verschiedene Module, die als Ganzes einen CASA-Knoten bilden, der den zentralen Baustein eines CASA-Netzwerkes darstellt.

Module im CASA-Knoten
: Als Module im CASA-Knoten werden jene Bestandteile des CASA-Knotens bezeichnet, die eigenständig installierbar, lauffähig und testbar sind. Durch ihre Verbindung lässt sich ein CASA-Knoten bilden. Ein Beispiel ist das Modul CASA-Importer, das externe Kontextquellen erschließt.

Komponente innerhalb eines Moduls
: Als Komponenten werden einzelne Bestandteile eines Moduls bezeichnet, die eine spezifische Aufgabe haben. Ein Beispiel ist hier die Facebook-Komponente, die innerhalb eines CASA-Importers verwendet werden kann, um Informationen aus dem sozialen Netzwerk Facebook zu erschließen.

4.1.1 Systemarchitektur des CASA-Systems

Zur Erläuterung der Systemarchitektur, also der Beschreibung der einzelnen Systemteile und der sie verbindenden Schnittstellen, wird das CASA-System in drei verschiedenen Varianten (A,B,C) von CASA-Netzwerken betrachtet.

Variante A - der private CASA-Knoten

Die Variante A ist in Abbildung 4.1 dargestellt. Sie besteht im Kern aus einem an ein Netzwerk angebundenen Server, auf dem eine Laufzeitumgebung für Java-basierte Webanwendungen, ein JavaEE Anwendungsserver, mit CASA-Knoten als Anwendung aufgesetzt sind. Der Nutzer kann über einen Computer mit einem Browser über das Netzwerk auf den CASA-Knoten zuzugreifen. Die Anbindung des Servers an das Internet ist optional, jedoch ist ohne eine Anbindung die Dienstempfehlung auf Dienste und Nutzer beschränkt, die sich im gleichen Netzwerk befinden.

Als Anwendungsserver wurde hier die quelloffene Variante von Oracles GlassFish Server[1] genutzt. Die Entscheidung für diese Umgebung beruht auf der intensiven Dokumentation und der aktiven Entwicklung des Projektes. Generell würden jedoch auch andere JavaEE Anwendungsserver, wie zum Beispiel Apache TomEE[2] oder WildFly von Red Hat[3] für das Ausführen von CASA in Frage kommen.

Variante A eignet sich im Besonderen für die Ausführung eines privaten CASA-Knotens, da hier die direkte Kontrolle des Nutzers über den Server möglich ist. So kann der Anwendungsserver mit dem CASA-Knoten auf einem Kleincomputer, wie zum Beispiel einem Raspberry Pi[4]ausgeführt werden, der im Heimnetzwerk des Nutzers eingebunden ist. Der Nutzer kann so den Zugriff von außen selbst bestimmen. Durch die physische Kontrolle, die er über den Server und damit den Knoten hat, kann er jederzeit das System starten, stoppen und nach Belieben auch alle Daten löschen.

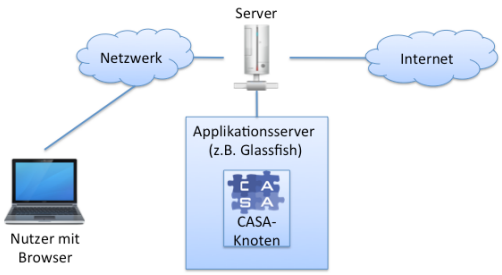

Abbildung 4.1: Variante A - Physische Sicht auf einen privaten CASA Knoten

Die Umsetzung mit Java und die Trennung zwischen dem physischen Server mit dem CASA-Knoten und dem Computer, den der Nutzer verwendet, um das CASA-System zu nutzen, ist optional. Das CASA-Konzept schreibt nicht die Verwendung von Java vor, es ist lediglich eine Realisierung, die sich durch den großen Umfang an frei verfügbaren Software-Bibliotheken angeboten hat.

Auch wäre es möglich, dass der Nutzer den CASA-Knoten auf dem Gerät ausführt, welches er stets mit sich führt. Jedoch bedingt dies, dass dieser Computer den CASA-Knoten mit seinen Modulen im Hintergrund ausführt und somit auf einem potentiell ressourcenbeschränktem Gerät Leistungskapazitäten bindet und die Batterie zusätzlich belastet.

Für die Kontrolle der eigenen Daten wäre diese Variante interessant, da so eine ortsunabhängige Kontrolle durch den Nutzer gewährleistet wäre. Daher könnte diese Variante in einer Erweiterung des CASA-Systems genauer betrachtet werden. Es könnte dann auch evaluiert werden, mit welchen Technologien sich hier eine resourcensparendere Implementierung realisieren ließe, die, ohne die Leistung zu sehr zu beeinflussen, auch auf einem Smartphone im Hintergrund ausgeführt werden kann. Dies hätte den Vorteil, dass geräteeigene Apps als mögliche Dienste eingebunden werden können. Somit könnte bei Eintritt einer selbstdefinierten Situation auch eine gerätespezifische Aktion ausgeführt werden.

[1] `https://javaee.github.io/glassfish/` [Online letzter Zugriff 15.02.2018]
[2] `http://tomee.apache.org/` [Online letzter Zugriff 15.02.2018]
[3] `http://wildfly.org/` [Online letzter Zugriff 15.02.2018]
[4] `https://www.raspberrypi.org/` [Online letzter Zugriff 15.02.2018]

Variante B - der gruppenöffentliche CASA-Knoten

Die Variante B, die in Abbildung 4.2 dargestellt ist, zeichnet sich dadurch aus, dass der CASA-Knoten an eine andere Anwendung gekoppelt ist. Dies ermöglicht eine Integration der Dienstempfehlungen direkt in die graphische Schnittstelle der Anwendung und einen Zugriff des CASA-Knotens auf die Datenbestände der Anwendung. Somit eignet sich diese Variante für die Nutzung als gruppenbasierter CASA-Knoten, der Inhalte in einer Anwendung bereitstellt, die von mehreren Nutzern verwendet wird. Die Schnittstelle zwischen der Anwendung und dem CASA-Knoten kann dabei auf verschiedene Weise umgesetzt werden. So kann das Plugin einen von dem Knoten publizierten Aggregator-Web-Service ansprechen und so Daten aus dem Knoten abfragen und bei Bedarf auch Daten an den Knoten geben.

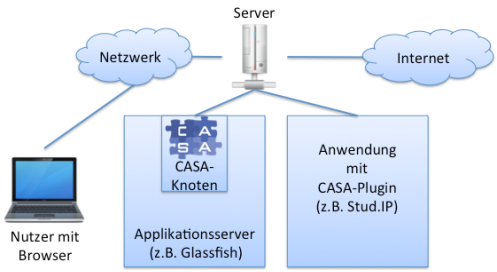

Abbildung 4.2: Variante B - Physische Sicht auf einen CASA-Knoten, der über ein Plugin an eine Anwendung angebunden ist

Diese Methode erlaubt eine Kommunikation über Technologiegrenzen hinweg, da so auch eine Anwendung, die nicht in Java umgesetzt ist, mit dem CASA-Knoten kommunizieren kann. Über das intern verwendete Framework Apache Camel[5] lassen sich auch andere Schnittstellen umsetzen. So können durch das Framework direkte Nachrichtenwege etabliert werden, falls die Anwendung und der CASA-Knoten in der selben Java-Laufzeitumgebung ausgeführt werden. Ebenso können Schnittstellen zu Datenbanken darüber aufgebaut werden. Abgesehen davon verhält sich die Variante B analog zur Variante A. So ist auch hier die graphische Schnittstelle des CASA-Knotens über das Netzwerk erreichbar. Erfolgt eine Einbindung der Dienstempfehlung in die Anwendung, so benötigt der Nutzer hier auch Zugang zu der Anwendung, um diese nutzen zu können. Eine Verbindung zum Internet ist, analog zu Variante A, optional.

Variante C - der öffentliche CASA-Knoten

Variante C behandelt den Fall, dass Dienste oder Informationen aus Anwendungen eingebunden werden, die sich nicht im selben Netzwerk wie der CASA-Knoten befinden und bei denen nicht die Möglichkeit eines CASA-Plugins besteht. Diese Variante ist in Abbildung 4.3 dargestellt und umfasst im Besonderen Dienste und Anwendungen, die für eine breite Öffentlichkeit angeboten werden, wie dies bei einem Karten-, einem Wetter- oder einem Nahverkehrsdienst der Fall ist.

Die Kommunikation erfolgt hier über ein Importer-Modul des CASA-Knotens, das in Abschnitt 3.4.5 konzipiert wurde und dessen Implementierung in Abschnitt 4.2.2 erläutert wird. Generell handelt es sich hierbei um domänen- und anwendungsspezifische Schnittstellen, die die Informationen aus einer entfernten Datenquelle erfassen und für den CASA-Knoten aufbereiten. Wie der Importer die Informationen aus der Datenquelle extrahiert, ist abhängig von der jeweiligen Quelle. So wurde für OpenStreetMaps ein Modul entwickelt, das die Roh-

[5] `http://camel.apache.org/` [Online letzter Zugriff 15.02.2018]

daten der Karten abfragt und die XML-Strukturen nach für den CASA-Knoten relevanten Objekten, wie zum Beispiel Nahverkehrshaltestellen, durchsucht. Im Gegensatz dazu wurde für das soziale Netzwerk Facebook prototypisch ein Modul entwickelt, das die für den Nutzer interessanten Daten über eine Facebook-eigene Programmierschnittstelle bezieht. Diese Beispiele sollen verdeutlichen, dass es auf Grund der Heterogenität der Anwendungen und Quelldatenformate keinen einheitlichen Weg gibt, diese zu erschließen.

Mit dieser Variante ist es möglich, einen CASA-Knoten zu erstellen, der die Daten von einer oder mehreren entfernten Anwendungen bezieht, die zwar voneinander unabhängig sind, jedoch zusammen einen Mehrwert für den Nutzer erbringen können. So ist hier ein öffentlicher Knoten möglich, der über einen Importer die nächstgelegenen Haltestellen aus OpenStreetMaps ermittelt und anschließend die Abfahrtspläne über eine Verkehrsinformationsseite für den Nutzer bereitstellt. Da die Informationsquelle in dieser Variante nur über das Internet erreichbar ist, ist hier eine Verbindung zum Internet zwingend notwendig.

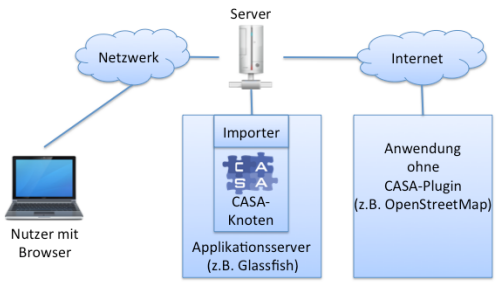

Abbildung 4.3: Variante C - Physische Sicht auf einen CASA-Knoten, der ohne Plugin an eine externe Anwendung angebunden ist

4.1.2 Software-Architektur des CASA-Systems

Nachdem die einzelnen Systemteile beschrieben wurden, die mit einem CASA-Knoten in Verbindung stehen, soll nun die in dieser Arbeit entwickelte Software-Architektur des CASA-Knotens selbst betrachtet werden. Da der Begriff der Software-Architektur auf verschiedene Arten definiert wird, musste für diese Arbeit eine geeignete Definition ausgewählt werden, die neben den Bestandteilen des Systems auch die Kopplung an die zuvor beschriebenen anderen Systemteile umfasst. In Posch et. al. wird Softwarearchitektur definiert als:

> [Softwarearchitektur ist] „die Zerlegung des Systems in seine Hauptbestandteile auf der obersten Ebene [...]. Sie definiert die Architekturbausteine, deren Verantwortlichkeiten, Instanzen, Schnittstellen und wie diese miteinander interagieren."
>
> (entnommen aus [Posch 11]).

In dieser Definition fehlt der Aspekt der Bedingungen, die durch äußere Systeme an das System gestellt werden.

Es wird daher auf die Definition von Qin verwiesen:

> Software-Architektur ist definiert als: Die fundamentale Organisation eines Systems, ausgedrückt durch seine Komponenten, ihre Beziehungen zueinander und zur Umgebung sowie die Richtlinien, die das Design und die Entwicklung leiten.
>
> (frei übersetzt nach [Qin 08, Seite 13])

Basierend auf dieser Definition wird in diesem Abschnitt zuerst auf Richtlinien und Anforderungen eingegangen, die zur Entwicklung dieser Architektur führten und dargelegt, wie diese

die Architekturentwicklung beeinflussten. Dabei wird die Umsetzung des CASA-Knotens mit seinen verschiedenen Modulen erläutert.

Bei der Entwicklung der Softwarearchitektur standen im Besonderen die technischen Anforderungen im Fokus, die in Abschnitt 2.6 formuliert wurden. Diese lassen sich in zwei wesentliche Herausforderungen unterteilen, die Heterogenität und die Skalierbarkeit.

Die Heterogenität liegt auf mehreren Ebenen vor. Zum einen verfügt der Nutzer heute über eine Vielzahl möglicher Endgeräte, die zur Nutzung einer kontextbasierten Dienstempfehlung verwendet werden können, während gleichzeitig die Nutzung auch als Einbindung in externe Anwendungen möglich sein soll. Zum anderen sind die Objekte, die durch die Dienste erschlossen werden, heterogen. Diese können von kleinen Entitäten, wie einem Video, über eine ganze Webseite bis hin zu einem physischen Gerät wie einem Drucker reichen. Als Schnittstelle wird für diese Arbeit von einem Browser ausgegangen, weshalb die Objekte teilweise erst in Webseiten eingebettet werden müssen, die dann von einem Browser wiedergegeben werden können.

Die Skalierbarkeit des Systems ist ebenfalls eine mehrschichtige Herausforderung. So ist die Zahl der möglichen Domänen, in denen das System Anwendung finden kann, sehr hoch, da jeder Nutzer, jede Nutzergruppe und jede Anwendung eine eigene Domäne definieren kann. Die Software muss daher sowohl von einem einzelnen Nutzer als auch von einer großen Gruppe von Nutzern in Anspruch genommen werden können.

Um diese beiden Hauptanforderungen zu adressieren, wurde für den CASA-Knoten eine Softwarearchitektur mit separaten Modulen entwickelt. Diese ist in Abbildung 4.4 als Bausteinsicht dargestellt.

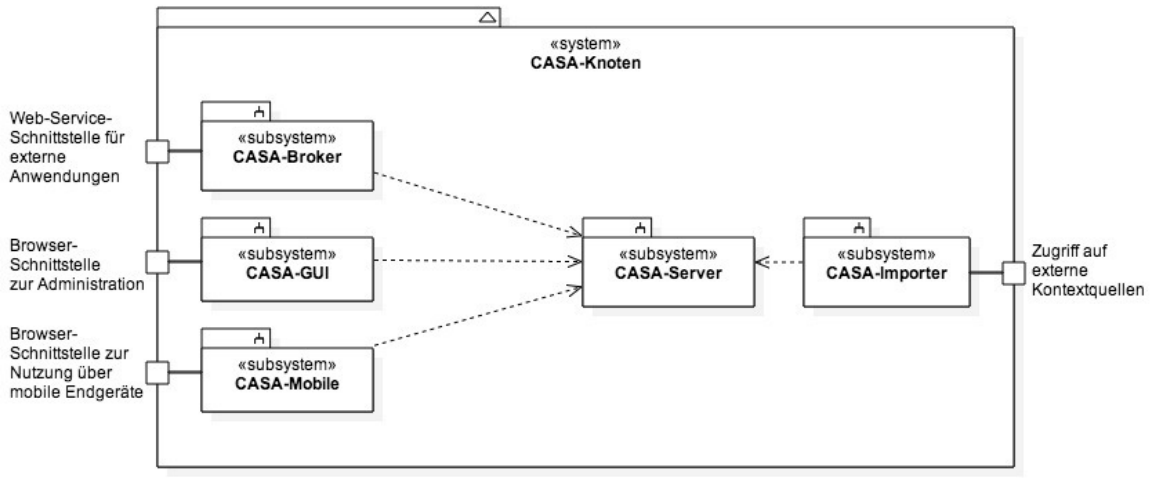

Abbildung 4.4: Bausteinsicht des CASA-Knotens als System mit seinen Subsystemen

Kern des CASA-Knotes ist der CASA-Server. Wie in Abbildung 4.4 dargestellt, sind alle anderen Module von diesem abhängig. Dabei bezieht sich diese Abhängigkeit jedoch nicht auf die Ausführbarkeit, sondern auf die Kommunikation, da diese vom CASA-Server verwaltet wird. Dieses Modul implementiert das wissensbasierte System aus dem CASA-Konzept, in dem das Wissen um die Situation des Nutzers mit den Regeln zur Dienstempfehlung verknüpft

wird. Realisiert wurde diese Einheit mit dem regelbasierten System JBoss Drools[6], das ein Unterprojekt von JBoss Kie[7] ist. Ausschlaggebend für die Wahl dieses Softwareproduktes war die gute Dokumentation, die Umsetzung in Java und quelloffene Verfügbarkeit.

Von diesem Modul aus bestehen Verbindungen zu allen anderen Modulen im CASA-Knoten. Diese wurden mit der Kommunikationsinfrastruktur von Apache Camel[8] umgesetzt. Es handelt sich dabei um eine quelloffene und in Java implementierte, nachrichtenvermittelnde Middleware, die unter anderem geeignet ist, um Anwendungen zu verbinden, die in der gleichen JVM[9] ausgeführt werden. Weitere Erläuterungen zur inneren Umsetzung des CASA-Servers werden in Abschnitt 4.2.1 gegeben.

Auf der rechten Seite der Abbildung 4.4 ist das Modul CASA-Importer dargestellt. Es wird unabhängig vom CASA-Server ausgeführt, jedoch publiziert es seine Camel-Schnittstellen über MulticastDNS [Cheshire 13] und kann so von einem parallel ausgeführten CASA-Server gefunden und dynamisch eingebunden werden. Wie in Abschnitt 3.4.5 bereits konzipiert, realisiert dieses Modul das Erschließen von externen Kontextquellen. Dazu werden die für den Betrieb des jeweiligen CASA-Knoten notwendigen Importer-Komponenten in diesem Modul gebündelt. So können für einen öffentlichen Knoten mit der Domäne Verkehrsinformationen Importer-Komponenten zu Kartendiensten, Schnittstellen des Nahverkehrs und Schnittstellen des Fernverkehrs gekapselt werden. Auf diese Art kann die Heterogenität adressiert werden, da die jeweiligen Importer-Komponenten spezifisch an den Sensor oder die Datenquelle angepasst werden und nicht an die Domäne. Für die Domäne entscheidend ist die Kombination verschiedener Importer-Komponenten sowie die Übersetzung in das domänenspezifische Kontextmodell. Letzteres wird im Implementierungsabschnitt zu den Kontextmodellen 4.2.6 erläutert, während der innere Aufbau der Importer in Abschnitt 4.2.2 tiefergehend beschrieben wird.

Da der Fokus des CASA-Servers die Erschließung der Kontextdaten und die Empfehlung von Diensten ist, war es notwendig, für den Nutzer Schnittstellen zu schaffen, die es ihm erlauben, die Funktionen des CASA-Servers zu konfigurieren und auch die Dienstempfehlung zu nutzen. Für die Konfiguration wurde das Modul CASA-GUI implementiert, das eine Administrationsoberfläche für verschiedene Funktionen des CASA-Knotens anbietet und auch für die Konfiguration der CASA-Importer genutzt werden kann. Es wurde mit Hilfe des Java-Frameworks Vaadin[10] entwickelt. Dieses unter der Apache 2.0 stehende quelloffene Werkzeug ermöglicht die Entwicklung von serverseitig ausgeführten Webanwendungen in Java. Neben dem Modul CASA-GUI wurde auch CASA-Mobile mit Vaadin entwickelt. Dabei handelt es sich um die Schnittstelle für mobile Geräte zur Nutzung der ortsabhängigen personalisierten und der ortsunabhängigen Dienstempfehlungen des CASA-Knotens durch den Nutzer. Der Fokus lag hier auf einer plattform- und geräteunabhängigen Schnittstelle, die die Heterogenität der Endnutzergeräte adressiert und gleichzeitig genutzt werden kann, um die empfohlenen Dienste direkt aufzurufen. Weitere Erläuterungen zu den graphischen Oberflächen des CASA-Knotens werden in den Abschnitten 4.2.3 und 4.2.4 gegeben.

[6] `https://www.drools.org/` [Online letzter Zugriff 15.02.2018]
[7] `http://www.kiegroup.org/` [Online letzter Zugriff 15.02.2018]
[8] `http://camel.apache.org/` [Online letzter Zugriff 15.02.2018]
[9] Java Virtual Machine, Laufzeitumgebung für Java-Anwendungen
[10] `https://vaadin.com/home` [Online letzter Zugriff 15.02.2018]

Das letzte Modul des CASA-Knotens mit der Bezeichnung CASA-Broker wird ausschließlich für CASA-Knoten benötigt, deren Dienstempfehlungen in eine externe Applikation eingebunden sind, wie dies bei Variante B der Fall ist. Dieses Modul kapselt die anwendungsspezifischen Anforderungen und erlaubt so eine Wiederverwendung des jeweiligen Brokers in verschiedenen Domänen. Dazu werden innerhalb dieses Moduls die Dienste, die von dem CASA-Server als Empfehlung geliefert wurden, an die jeweilige Anwendung angepasst. Die Verbindung zum CASA-Server ist, wie auch bei den anderen Modulen, mit Camel realisiert und ermöglicht das direkte Einfügen von Fakten sowie das Abfragen von Dienstempfehlungen. Die Schnittstelle nach außen ist hier jedoch nicht der Browser, sondern ein Web-Service, der es der externen Anwendung erlaubt, auf die genannten Funktionalitäten zuzugreifen. Der CASA-Broker realisiert das in Abschnitt 3.4.4 vorgestellte Konzept des Aggregators, dessen genauere Implementierung in Abschnitt 4.2.5 erläutert wird.

Zusätzlich zum CASA-Broker kann, je nach angebundener Anwendung, auch noch ein anwendungsspezifisches Plugin nötig sein, um die Einbindung zu realisieren. Dies ist zwar nicht direkt Teil des CASA-Knotens, jedoch Teil des CASA-Netzwerks und wird daher in Abschnitt 4.3 genauer erläutert.

Im Folgenden werden die einzelnen Module mit ihren Komponenten genauer vorgestellt und es wird gezeigt, wie die Anforderungen aus den jeweiligen Konzepten adressiert werden.

4.2 Module des CASA-Knotens und deren Konfiguration

Da das Konzept dieser Arbeit darauf basiert, dass ein CASA-Knoten einer Domäne entsprechend erstellt und konfiguriert wird, wird im Folgenden zuerst erläutert, wie ein CASA-Knoten allgemein erstellt wird, während in den anschließenden Abschnitten auf die einzelnen Subsysteme und ihre Konfiguration eingegangen wird.

Um einen CASA-Knoten zu erstellen, ist es zuerst notwendig, die Laufzeitumgebung für den Knoten zu schaffen. Dies erfordert einen laufenden JEE Anwendungsserver, wie zum Beispiel Glassfish. In einem nächsten Schritt müssen die Module heruntergeladen werden, die für den Knoten benötigt werden. Um die Entwicklung mit der Öffentlichkeit zu teilen, sind die Quellen über das freie Versionsverwaltungssystem Git [Torvalds 17] auf der Plattform GitHub[11] verfügbar. In Abbildung 4.5 ist

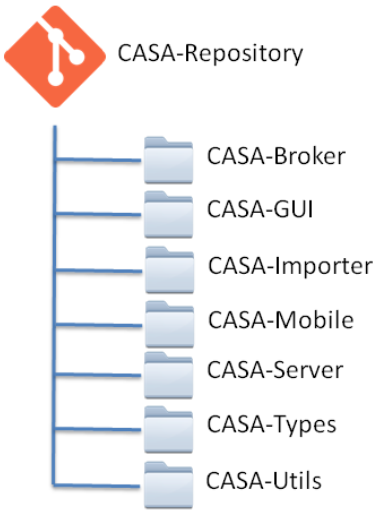

Abbildung 4.5: Struktur des CASA-Repositories auf GitHub

[11] https://github.com/Talrik/CASA/tree/develop [Online letzter Zugriff 15.02.2018]

ein Überblick über die im Repository vorhandenen Module gegeben. Von diesen werden für einen CASA-Knoten im Regelfall alle außer dem Broker benötigt, dieser ist ausschließlich für eine Einbettung in externe Anwendungen notwendig. Die Module CASA-Types und CASA-Utilities enthalten jeweils Bibliotheken, die für die anderen Module benötigt werden. Nach dem Einbinden der jeweils gewünschten Kontextmodelle aus dem CASA-Types-Modul können die anderen Module installiert werden, indem die Web-Archive-Dateien aus den build-Ordnern des Repositorys im JavaEE Server gestartet werden.

Ein direkt aus diesen Dateien gestarteter CASA-Knoten läuft in einer Standardvariante, damit verfügt dieser jedoch noch nicht über Fakten oder Regeln. Diese können über die Administrationsoberfläche von CASA-GUI hinzugefügt werden. Alternativ können bereits vor dem Start des Knotens Regeln und Fakten im Modul CASA-Server konfiguriert werden. Dies wird im nächsten Abschnitt ausführlich erläutert.

4.2.1 Zentrale Organisation der Kontextverwaltung

Um das Modul CASA-Server zu initialisieren ist es notwendig, dass im JavaEE-Server ein Kontextmodell aus dem Modul CASA-Types hinterlegt wurde. Dabei ist auf die Version zu achten, da das Kontextmodell einer steten Weiterentwicklung unterworfen ist. Die Version dieser Bibliothek gibt Auskunft über die Kompatibilität zu anderen Modulen und Komponenten. Da der Server darauf angewiesen ist die Anfragen, Fakten und Regeln von anderen Modulen interpretieren und bearbeiten zu können, publiziert er die von ihm verwendete Version des Kontextmodells über MulticastDNS. Module, die eine ältere Version als die des CASA-Servers verwenden, können potentiell zu fehlerhaften Zuständen führen. Innerhalb eines Knotens kann durch deren Ausschluss davon ausgegangen werden, dass nur Module mit kompatiblen Kontextmodellversionen an den CASA-Server angebunden werden.

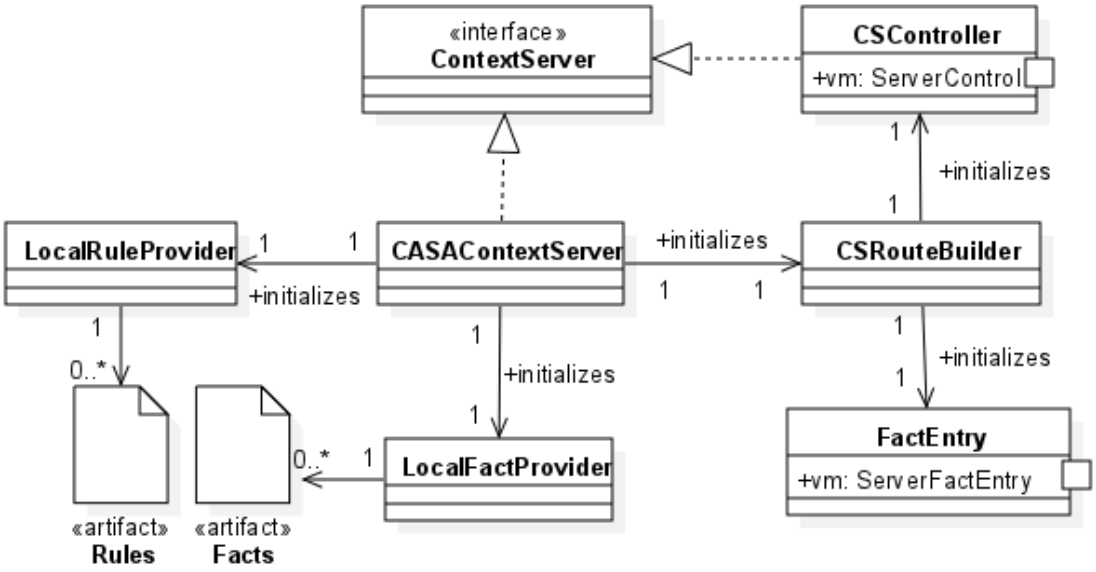

Abbildung 4.6: Das Subsystem CASA-Server mit seinen wichtigsten Klassen

Abbildung 4.6 zeigt die wichtigsten Klassen des Moduls CASA-Server. Die Klasse `CASAContextServer` ist die Hauptklasse dieser Web-Anwendung und ist als Singelton Bean[12] realisiert, damit der JavaEE Server diese Klasse nicht mehrfach instanziiert. Dies stellt sicher, dass innerhalb eines CASA-Knotens nur ein CASA-Server erzeugt werden kann. In einem ersten Schritt wird die Klasse `CSRouteBuilder` initialisiert, die die Kommunikation für den Server kapselt. Mit der Klasse `FactEntry` wird die nach außen gerichtete Schnittstelle zum Einfügen neuer Fakten von dieser erstellt und für andere Module des CASA-Knotens verfügbar gemacht. In der Klasse `CSControl`, die ebenfalls durch `CSRouteBuilder` initialisiert wird, wird die Schnittstelle nach außen für die Steuerung des gesamten Moduls durch andere Module erzeugt. Die Kommunikation zwischen diesen Komponenten erfolgt über Apache Camel, wobei die Kommunikationsendpunkte durch `CASAContextServer` mit Hilfe von MulticastDNS innerhalb des lokalen Netzwerkes publiziert werden. In einem nächsten Schritt werden die Klassen `LocalFactProvider` und `LocalRuleProvider` initialisiert. Sie stellen dem `CASAContextServer` das initial vorhandene Wissen in Form von Fakten und Regeln zur Verfügung, wobei der Pfad zu diesen konfiguriert werden kann. Mit diesem Wissen erzeugt `CASAContextServer` eine `StatefulKnowledgeSession`. Dies ist eine Klasse aus Drools, die die Wissensbasis repräsentiert und persistente, also dauerhafte, Änderungen an ihr ermöglicht.

Um diese Wissensbasis mit weiteren Fakten zu füllen, werden Importer benötigt, deren Aufbau und Funktionsweise im nächsten Abschnitt erläutert werden.

4.2.2 Das Modul CASA-Importer

Das Modul CASA-Importer bündelt die Funktionalität zum Einbinden neuer Fakten und zum Aktualisieren bestehender. Die Quelle dieser Daten wird als Sensor betrachtet, der seine Informationen in einem proprietären Format und über eine proprietäre Schnittstelle zur Verfügung stellt. Um das Problem der Heterogenität der Sensordaten möglichst kompakt zu kapseln, ist es Teil der Aufgabe der Importer, die Daten aus ihrem Ursprungsformat in ein Format zu konvertieren, das mit dem Kontextmodell des Servers, das durch die CASA-Types-Bibliothek definiert wird, kompatibel ist. Um dies zu realisieren, wurde das Modul CASA-Importer exemplarisch mit einigen Importer-Komponenten umgesetzt. Ein vereinfachtes Klassendiagramm zur Verdeutlichung des allgemeinen Aufbaus ist in Abbildung 4.7 dargestellt. Die wichtigste Klasse eines Importer Moduls ist der `CASAImporterManager`. Diese macht die von ihr verwalteten Importer-Komponenten über MulticastDNS im lokalen Netzwerk bekannt und ermöglicht es so, dass diese vom CASA-Server gefunden werden. Für das Erschließen komplexer oder mehrerer Datenquellen ist es auch möglich, mehrere ImporterManager zu verwenden, die dann wiederum von einem übergeordneten ImporterManager verwaltet werden. Die Importer-Komponenten werden direkt in den ImporterManager eingebunden und werden dann von diesem bei Bedarf angefragt. Der ImporterManager kapselt somit die Kommunikation des Subsystems und ermöglicht, dass die Importer-Module unabhängig von der verwendeten Kommunikationsmethode wiederverwendet werden können.

[12] Singleton bezeichnet ein Entwurfsmuster, das sicherstellt, dass von dieser Klasse nur eine Instanz erzeugt wird.

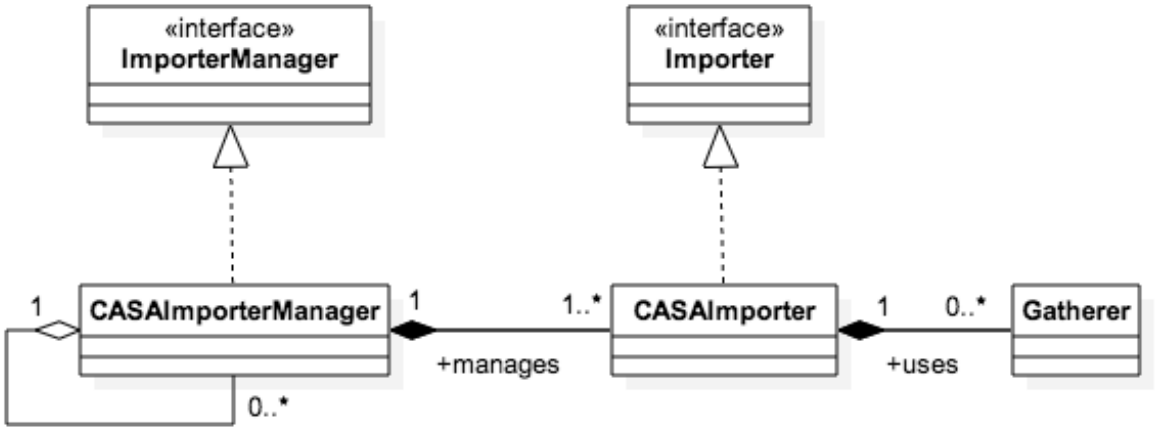

Abbildung 4.7: Vereinfachtes Klassendiagramm des Moduls CASA-Importer

Die Bezeichnung des Importers als `CASAImporter` in der Abbildung dient nur zur einfacheren Darstellung, da im Rahmen dieser Arbeit einige Importer-Komponenten realisiert wurden, deren Präfix sich nach der von ihnen erschlossenen Datenquelle richten, z.B. `FacebookImporter` oder `StudIPImporter`.

Im Inneren der Importer-Klasse erfolgt die Konvertierung der Daten und gegebenenfalls auch schon eine Aggregation oder Selektion von Daten anhand von Parametern, die der Importer bei seinem Aufruf durch den ihn verwaltenden ImporterManager bekommt.

Die Komponente wiederum bündelt quellenspezifisch eine Reihe von `Gatherer`-Klassen, die für die eigentliche Extraktion der Daten aus der Quelle verantwortlich sind. Somit wird hier die Kommunikation mit der Schnittstelle gekapselt, während die Logik zur Konvertierung in der Importer-Klasse gebündelt ist. Mit dieser Architektur wird die Heterogenität der Sensoren adressiert, da für einen Sensor die Logik zur Anfrage der Schnittstelle nur einmal erstellt werden muss und dann in verschiedenen Domänen verwendet werden kann.

Das Anfragen der Importer kann über zwei verschiedene Wege erfolgen. Zum einen ist es möglich, dass die Importer-Komponente in den Regeln benannt wird, sodass diese nur dann angefragt wird, wenn eine Regel existiert, die dies explizit verlangt. In den Regeln wird dann auch festgelegt, ob aus dem Importer alle Daten ausgelesen werden sollen oder welche Parameter für den Importer verwendet werden müssen. Außerdem spezifizieren die Regeln, wie aktuell die Daten gehalten werden sollen. So sind die GPS-Informationen eines Gebäudes statisch, während die Positionsdaten eines Mobiltelefons in einem definierten Intervall aktualisiert werden sollten. Die Alternative hierzu ist das Starten der Importer aus dem Modul CASA-GUI heraus, das im nächsten Abschnitt erläutert wird.

4.2.3 Administrationsschnittstellen für das CASA-Konzept

Zur Interaktion des Nutzers mit dem System war es notwendig, exemplarisch eine grafische Administrationsschnittstelle zu schaffen. Daher wurde in dem Modul CASA-GUI eine Web-

Anwendung erstellt, die für die Verwaltung des CASA-Knotens genutzt werden kann. Dabei implementiert das Modul eine graphische Oberfläche für alle Funktionen, die `CSController` als Schnittstelle im CASA-Server nach außen anbietet. Auf diese Art kann das Modul CASA-GUI auch mit anderen Implementierungen des CASA-Servers kombiniert werden, da auch diese das Interface `ContextServer`, das in Abbildung 4.8 dargestellt ist, implementieren.

Die Funktionen umfassen das Starten und Stoppen des Servers ebenso wie das Sichern und Wiederherstellen der Wissensbasis des Systems. Neben den Grundfunktionen können über diese Schnittstelle auch die Regeln und Anfragen eingesehen, verändert oder gelöscht werden, die aktuell auf dem Knoten installiert sind. Über die in dem Interface `ContextServer` deklarierten Funktionen hinaus, ermöglicht das Modul auch eine Übersicht über die im CASA-Netzwerk vorhandenen Importer. Diese können manuell in den Knoten eingebunden und konfiguriert werden. Mit dem Starten und Stoppen der Importer über diese Anwendung können die Fakten manuell in die Wissensbasis importiert werden. Außerdem ist es auch möglich manuell Fakten zu bearbeiten, zu löschen oder zu erstellen.

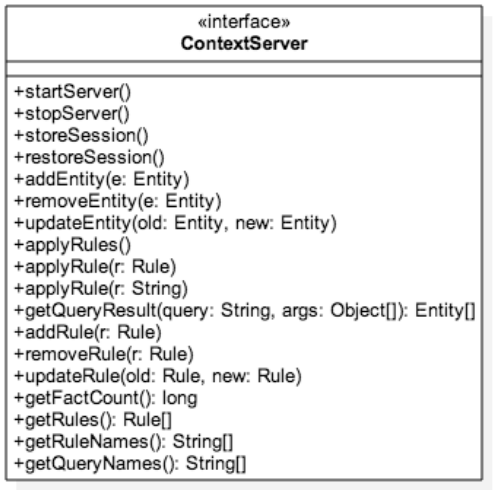

Abbildung 4.8: Interface ContextServer, dessen Funktionen alle über die graphische Oberfläche nutzbar sind

Diese Oberfläche dient konzeptionell exklusiv der Administration. Für die Nutzung der Dienstempfehlungen in der Evaluation wurde eine weitere graphische Schnittstelle geschaffen. Dies wird im nächsten Abschnitt erläutert.

4.2.4 Schnittstelle für mobile Geräte

Neben der Einbindung von CASA-Knoten in bestehende Anwendungsumgebungen ist gerade die direkte Interaktion des Nutzers mit dem System eine Möglichkeit, um viele Kontextinformationen des Nutzers verwenden zu können, ohne dass diese außerhalb seines Zugriffs gespeichert werden. Um diesen direkten Zugriff zu illustrieren, wurde im Rahmen dieser Arbeit eine Schnittstelle geschaffen, die eine Nutzung des CASA-Knotens durch mobile Geräte

realisiert. Dabei handelt es sich um eine Web-Anwendung, wobei diese durch das vaadin Framework[13] für die Nutzung auf mobilen Geräten konzipiert ist.

Dieses Modul ist im Speziellen für mobile Geräte gedacht und daher nicht für die Nutzung auf Desktop-Systemen geeignet. Eine Desktop-Nutzung kann jedoch zu Testzwecken durch den Safari-Browser erfolgen.

Die Anwendung ermittelt beim Aufruf über den Browser des mobilen Gerätes über mDNS die im lokalen Netzwerk verfügbaren Knoten und gibt dem Nutzer die Möglichkeit, sich mit diesen zu verbinden. Des Weiteren kann die Anwendung auch entfernte CASA-Knoten über ihre IP-Adresse einbinden. Dies erlaubt die Nutzung von entfernten öffentlichen und gruppenöffentlichen Knoten. In einem nächsten Schnitt legt der Nutzer fest, welche von den Knoten angebotenen Anfragen abgerufen und welche Informationen für die Anfragen verwendet werden sollen. Somit kann der Nutzer selbst bestimmen, welche Informationen preisgegeben werden. Die empfohlenen Dienste werden daraufhin aggregiert dargestellt. Eine Sortierung erfolgt lediglich danach, ob sie einer Aktivität, einem Ort oder keinem von beiden zugeordnet sind.

Ziel ist es dabei, dass ein Nutzer diese Anwendung als Teil des privaten CASA-Knotens bei sich zu Hause oder in einer anderen von ihm kontrollierten Umgebung ausführt. Über ein mobiles Gerät ist es so möglich auf diese Anwendung zuzugreifen und darüber die Dienstempfehlungen des privaten Knotens auszulesen. Auch kann die Anwendung genutzt werden, um Gruppen-Knoten und weitere öffentliche Knoten einzubinden. Die Zugangsdaten zu den Gruppen-Knoten, ebenso wie die aggregierten Informationen aus diesen Quellen, werden dabei ausschließlich in dem privaten Knoten gespeichert.

4.2.5 Anfrage und Aggregation der Dienste

Damit eine Anwendung eine Anfrage an den CASA-Knoten für ihre Domäne richten kann, sind ein Plugin in der Anwendung und ein Aggregator in dem Knoten durch das Konzept vorgesehen. Letzterer führt die Anfrage für die Anwendung im CASA-Knoten aus und ermöglicht eine Aufbereitung der Ergebnisse. Außerdem ist es auf diesem Weg möglich, ein zweistufiges Caching von häufig auftretenden Anfragen durchzuführen, um den Knoten zu entlasten. Im Modul CASA-Stud.IP wurde ein solches Plugin für das Lernmanagementsystem Stud.IP umgesetzt. Das Modul CASA-Broker beinhaltet einen passenden, prototypisch umgesetzten Aggregator.

Während das Plugin spezifisch für die Anwendung entwickelt wird, ist der Aggregator nur für die Domäne des Knotens bestimmt. Wird eine Anwendung weiterentwickelt, muss so lediglich getestet werden, ob das zugehörige Plugin noch funktioniert oder angepasst werden muss. Dabei arbeitet das Plugin strikt mit dem Aggregator und nicht mit dem CASA-Server direkt, was es ermöglicht, diesen weiterzuentwickeln und dies durch den Aggregator für die Anwendungen und ihre Plugins transparent zu halten.

Der Aggregator bietet für die Applikationen bzw. deren Plugins nach außen einen Web-Service an, über den die Anfrage der Applikation gestellt werden kann. Zur Kommunikation mit dem

[13] https://vaadin.com/ [Online letzter Zugriff 15.02.2018]

CASA-Server nutzt der Aggregator das Camel-Messaging, um die Anfragen weiterzugeben und die Antworten zu empfangen. Zu den Aufgaben des Aggregators gehört auch das Aussortieren doppelter Dienste. Auf der Seite des Plugins erfolgt die Sortierung und Aufbereitung der Dienste nach Spezifikation der Applikation.

Bei dem Eingang einer Anfrage aus der Anwendung wird überprüft, ob es möglich ist, die gestellte Anfrage direkt aus dem Cache zu beantworten. Sollte sich eine Anfrage noch nicht im Cache befinden, so erstellt der Aggregator ein Request-Objekt und fügt es der Wissensbasis des CASA-Servers hinzu. Danach stößt der Aggregator das Ausführen aller Regeln an und fragt schließlich wieder das Request-Objekt ab, dem durch das Ausführen der Regeln alle als relevant klassifizierten Dienste hinzugefügt wurden. Das Request-Objekt kann anschließend direkt durch den Aggregator oder durch entsprechend konfigurierte Regeln aus der Wissensbasis entfernt werden.

Diese Architektur mit dem Plugin und dem Aggregator erlaubt es, die Kommunikation für die Anwendung transparent zu machen. Einzig bei einer Änderung des Kontextmodells, das im nächsten Abschnitt ausführlicher beschrieben wird, muss bei allen beteiligten Komponenten getestet werden, ob sie noch konsistent funktionieren. Hier erhöht jedoch die verwendete Strategie der Vererbung der Kontextklassen die Robustheit, so dass nicht jede Änderung an den Klassen zwingend zu Fehlern in den beteiligten Modulen führt.

4.2.6 CASA-Kontextmodell

Das Kontextmodell beschreibt die möglichen Entitäten und ihre Eigenschaften, die innerhalb des wissensbasierten Systems verwendet werden, um den Kontext zu beschreiben. Es wurde, ebenso wie die restliche Software, in Java umgesetzt. Dabei werden die Kontextklassen jeweils durch serialisierbare Java-Klassen implementiert. Die Serialisierbarkeit erlaubt es, dass die Objekte, die als Instanzen einer Klasse erstellt werden, abgespeichert oder auch von einem System auf ein anderes übertragen werden können. Außerdem verfügt das Kontextmodell über eine Vererbungshierarchie, die die einzelnen Kontextklassen in Relation zueinander setzt.

Das Konzept sieht dabei zwei Arten von Kontextklassen vor, die Basis- und die Erweiterungskontextklassen. Die Basiskontextklassen leiten sich von den Basiskontexttypen Ort, Identität, Aktivität und Zeit ab, die bereits von Dey [Dey 00] definiert wurden. Diese bilden im CASA-Kontextmodell die Grundlage, wobei für die Zeit keine separate Klasse eingeführt wurde. Dabei erben diese Klassen von der Klasse `Entity`, welches die Basiseigenschaften aller Objekte im Kontextmodell definiert und so zum Beispiel regelt, unter welchen Bedingungen zwei instanziierte Objekte als identisch gelten. Erweiterungsklassen sind Klassen, die domänenspezifisch die Basisklassen erweitern und von ihnen erben. Dabei kann es vorkommen, dass sich Notwendigkeiten nach Klassen ergeben, die semantisch keinen Bezug zu den Basisklassen haben. In diesem Fall können sie zwar erstellt werden, müssen jedoch von der Klasse Entity erben, um mit dem System verwendet werden zu können.

Diese Architektur mit Basis- und Erweiterungsklassen erlaubt es, dass auch Objekte aus verschiedenen Domänen verarbeitet werden können, solange sich die Verarbeitung auf die Attribute der Basisklassen bezieht.

Im Folgenden werden die drei Basisklassen `Person`, `Event` und `Place` als Implementierungen der Kontexte Identität, Aktivität und Ort beschrieben. Das Konzept der Erweiterungsklassen wird dabei am Beispiel der Erweiterungsklassen für die Stud.IP-Domäne erläutert.

Abbildung von Personen und Identitäten

Um das Konzept der Identität abzubilden, wurde die Klasse `Person` umgesetzt. Dabei wird einer real existierenden Person ein solches Objekt zugeordnet, in dem die Informationen zum Namen und der Email gespeichert werden. Zusätzlich verfügt das Objekt über eine Liste mit Identitäten, die dieser Person zugeordnet sind und über die sich domänenspezifische Rollen und Zugänge verwalten lassen. Die Identität selbst ist dabei wiederum ein Erbe der Klasse `Person`, wobei hier nicht die vollständigen Nutzerinformationen redundant gespeichert werden, sondern nur der CASA-Nutzername, über den sich die Person identifizieren kann.

Im Bereich der Stud.IP-Erweiterungsklassen wurde hier die Klasse `StudIPIdentity` umgesetzt. Dies war notwendig, da in diesem System neben der Identität und der globalen Rolle, die der Nutzer im System hat, auch innerhalb jeder Veranstaltung eine davon abweichende Rolle existieren kann, die auch geändert werden kann.

Events als Abbildung von Zeit und Aktivitäten

Um Aktivitäten und Zeiträume im Kontextmodell wiederzugeben, wurde die Klasse `Event` implementiert. Ein Objekt vom Typ `Event` verfügt dabei über eine Bezeichnung und kann über einen Start- und Endzeitpunkt verfügen. Ein ortsunabhängiger Zeitpunkt, wie „Aufstehen“, lässt sich dabei ebenso realisieren wie ein Zeitraum „Urlaub“. Für das Verknüpfen mit Orten und Personen werden Unterklassen benötigt, wie sie im Fall von Stud.IP mit den Klassen `Course` und `Lecture` realisiert wurden. Dabei bildet ein `Course` eine Vorlesungsreihe ab, verfügt also über eine Liste von Personen als Teilnehmer ebenso wie über eine Liste von Orten, die verwendet werden. Zusätzlich dazu kann hier eine Menge mit mehreren Objekten vom Typ `Lecture` hinzugefügt werden, die dann die einzelne Veranstaltung mit ihrem separaten Start- und Endzeitpunkten wiedergeben sowie den oder die konkreten Orte für diese Veranstaltung.

Abbildung von geografischen und logischen Orten

Um geographische und logische Orte im Kontextmodell abbilden zu können, wurde die Klasse `Place` implementiert. Diese erbt von `Entity` und benötigt lediglich einen Bezeichner, um initialisiert zu werden. Für geographische Orte können auch Koordinaten in Form von Breiten- und Längengrad angegeben werden. Umgesetzt wurden für das Stud.IP zusätzlich die Klassen `Building` und `Room`. Erstere erbt von `Place` und erfasst zusätzlich die Adresse und eine Liste von zugehörigen Räumen. Letzterer verweist lediglich zurück auf das zugehörige Gebäude und erlaubt die Verknüpfung mit Veranstaltungsklassen wie der `Lecture`. Zu Demonstrationszwecken wurde auch eine Klasse `Stop` umgesetzt, die als Basis-Klasse für Haltestellen

dient. So lassen sich über eine Verbindung mit der OpenStreetMap nahegelegene Haltestellen zu einer Position ermitteln und als Objekte abbilden.

4.2.7 Erweiterungsklassen und Pakete mit Domänenbezug

Die Klassensammlungen werden in Software-Bibliotheken verwaltet. Die Bibliothek im Modul CASA-Types enthält dabei neben den Basisklassen auch einige Klassen, die als Prototypen für die weitere Entwicklung genutzt werden können. Dazu gehören zum Beispiel die Klassen `Service`, `Website` und `LocationWebsite`. `Service` ist dabei allgemein und kann somit auch für einen Dienst bzw. eine Dienstleistung in der physischen Welt stehen. `Website` erbt von `Service` und konkretisiert diesen Dienst, indem hier eine URL eingefügt wird. `LocationWebsite` ergänzt `Website` um eine Ortsverknüpfung. Dies ist, wie bereits angeführt, lediglich als prototypische Vorlage zu verstehen und zeigt, wie das Kontextmodell verfeinert werden kann.

Da auch das Kontextmodell Teil eines sich weiterentwickelnden Systems ist, wird es mit einer Versionsnummer verknüpft, unter der es online verfügbar ist. Diese Versionsnummer kann von Regeln und Queries verwendet werden, um sicherzustellen, dass die Version des Kontextmodells, mit dem diese entwickelt wurden, kompatibel zu der in ihrer aktuellen Installation ist.

Für Knoten mit einem neuen Domänenbezug muss das Kontextmodell in der Regel erweitert werden. Dies erfolgt über Erweiterungspakete, die die Klassen bündeln, die nicht Teil von CASA-Types sind. Auch hier kann angegeben werden, welche CASA-Types-Version vorausgesetzt wird. Die Erweiterungspakete dienen als Grundlage der domänenspezifischen Regeln und sollten daher mit diesen synchron entwickelt und an diese angepasst werden.

Die Entwicklung des Stud.IP-Moduls, in dem dies umgesetzt wurde, wird im nächsten Abschnitt genauer betrachtet, wobei hier neben den bereits erwähnten Erweiterungsklassen für das Kontextmodell besonders die Struktur des Plugins sowie die Kommunikation mit dem WebService und dem Knoten im Vordergrund steht.

4.3 Umsetzung eines Gruppenknotens am Beispiel Stud.IP

Im Rahmen dieser Arbeit wurde eine Implementierung zum Einsatz im Produktionsprozess und als Demonstrator umgesetzt. Dabei lag neben der reinen Umsetzung des Konzeptes auch die Frage nach der Anwendbarkeit und Nachhaltigkeit des Konzeptes im Fokus.

Die Implementierung dient zudem als Ansatzpunkt für die Evaluation, bietet einen Mehrwert für die Universität und steht dem Graduiertenkolleg MuSAMA als Ausgangspunkt für weitere Forschungsfragen zur Verfügung. Von der Struktur des CASA-Konzeptes her ist der Gruppenknoten mit Anwendungsplugin in besonderem Maße komplex, da er eine Erweiterung des Kontextmodells, eine Einbindung in eine existierende Anwendung und eine Berücksichtigung von Nutzern in verschiedenen Rollen erfordert. Eine Erweiterung des existierenden Stud.IP-

Lernmanagementsystems an der Universität Rostock erwies sich dabei als passend für die gestellten Anforderungen.

Im folgenden Abschnitt wird die Architektur des umgesetzten Systems erläutert, wobei unter anderem darauf eingegangen wird, wie das CASA-Konzept sowohl mit verringertem Funktionsumfang für ein Produktivsystem als auch mit vollständigem Funktionsumfang für das Experimentalsystem umgesetzt werden konnte. Anschließend wird erläutert, welche Erweiterungsklassen umgesetzt wurden, um dem Konzept des Stud.IP gerecht zu werden. Abschließend wird auf die umgesetzten Nutzerschnittstellen eingegangen, die für die Verwaltung der gruppenöffentlichen Dienste erstellt wurden.

4.3.1 Architektur für eine hybride Nutzung als Produktiv- und Experimentalsystem

Ziel der Umsetzung war es, das in dieser Arbeit konzipierte System in einem produktiven Umfeld einsetzen zu können. Dabei ergaben sich einige Anforderungen, die zu einer Architektur mit zwei möglichen Nutzungsvarianten führten.

Die Umsetzung wurde unterstützt durch das Rechenzentrum der Universität Rostock, das das Stud.IP als eines von mehreren Lernmanagementsystemen der Hochschule verwaltet. Zusätzlich ergab sich die Möglichkeit einer Kooperation mit dem Juniorstudium-Projekt[14] der Universität Rostock, das interessierten Schülern die Möglichkeit gibt an Universitätskursen teilzunehmen und Prüfungen abzulegen. Hier wird Stud.IP in einer separaten Installation als Lernmanagementsystem eingesetzt.

Beide Projektpartner stellten die Anforderung, dass das System nur einen geringen Mehraufwand verursachen und in der Wartung unkompliziert sein sollte. Die Dienste sollten von den Nutzern, Tutoren und Administratoren des Stud.IPs kommen. Eine Einbindung von Empfehlungen basierend auf außeruniversitären Daten war nicht gefordert. Des Weiteren sollten für die Administratoren kein zusätzlicher Aufwand für die Pflege der Server-Instanzen notwendig sein sein.

Jedoch sollte das System als Demonstrator des CASA-Konzeptes ein möglichst großes Funktionsspektrum anbieten und auch für weitere Studien und Experimente zum CASA-System zur Verfügung stehen.

Als Lösung wurde die Systemarchitektur in der Variante B (vgl. Seite 87) umgesetzt, in der der Anwendungsserver für den CASA-Knoten auf einem beliebigen Rechner im Netzwerk installiert sein kann. Des Weiteren wurde der Cache für die Dienste im Stud.IP-Plugin in die Stud.IP-Datenbank überführt und somit persistiert. Für die Produktivversion ergab sich so eine Lösung, die vollständig in dem Plugin gebündelt und ohne den CASA-Knoten betrieben werden konnte. Die Eingabe der Regeln und Dienste erfolgte vollständig über das Plugin, weshalb ein Cache an dieser Stelle es erlaubte, die Kommunikation mit dem CASA-Knoten zu ersetzen.

Die Struktur des Plugins wurde so umgesetzt, dass das Plugin seine Dienstanfragen zuerst an den Cache und dann an einen eventuell konfigurierten CASA-Knoten stellt. Sobald ein CASA-

[14] https://juniorstudium.uni-rostock.de/ [Online letzter Zugriff 15.02.2018]

Knoten, der einen passenden Broker als Web-Service publiziert hat, konfiguriert ist, werden die Dienstanfragen an diesen weitergeleitet, wenn sie nicht bereits im Cache verfügbar sind. Neue Diensteintragungen und -verknüpfungen werden ebenso weitergeleitet. So kann das Umstellen auf die Experimentalversion durch das Aufsetzen eines Knotens im Netzwerk erfolgen. Ab dann stehen auch die Dienstempfehlungen des Knotens im Stud.IP zur Verfügung.

Der Web-Service bildet dabei die Verbindung zwischen den Systemen. Auf der Seite des Experimentalsystems kann er unabhängig vom produktiven Stud.IP verändert werden, um z.B. neue Funktionalitäten zu bieten. Dazu kann er deaktiviert werden. Dies stellt für das Produktivsystem keinen kritischen Fehlerfall dar, da dieses dann automatisch nur auf lokale Dienste zurückgreift. Die Abfrage über den Web-Service ist langsamer als eine Datenbankabfrage auf dem gleichen System, weshalb der Web Service einen Flaschenhals im System darstellt. Dieser Unterschied in der Latenz blieb in der Implementierung für das Produktivsystem im Bereich weniger hundert Millisekunden und fiel daher während des Seitenaufbaus nicht auf. In künftigen Entwicklungen sollte hier durch Nutzung von Technologien wie Ajax ein dynamisches Nachladen erfolgen.

Ein weiterer Vorteil dieser Art der Umsetzung ist, dass auch dem Open-Source-Gedanken hinter Stud.IP entsprochen und das Plugin, das auch ohne CASA-Knoten funktioniert, der Gemeinschaft aller Stud.IP-Nutzer zur freien Verfügung gestellt werden konnte.

4.3.2 Erweiterungsklassen mit Domänenbezug Stud.IP

Wie bereits erwähnt wurden für einige Kontextklassen auch domänenspezifische Erweiterungsklassen implementiert. Diese werden in separaten Klassensammlungen gebündelt und erfordern eine Mindestversion des Basiskontextmodells, um verwendet werden zu können. Sie dienen als Grundlage der domänenspezifischen Regeln und sollten daher mit diesen synchron entwickelt und angepasst werden.

Da zu einer domänenspezifischen Erweiterung mehrere Module gehören, wurde mit dem Modul CASA-StudIP exemplarisch ein Modul erstellt, das die zur Erstellung eines Gruppenknotens für das Stud.IP notwendigen Klassen enthält. Dies schließt neben den Importern für die Datenbanken auch den Aggregator, die Regeln, Anfragen und Typen mit ein. Es ist jedoch keine an sich separate Entwicklung, sondern nur eine Zusammenfassung der Stud.IP-spezifischen Erweiterungsklassen, die bereits in den anderen Modulen entwickelt wurden.

4.3.3 Nutzerschnittstellen zur Eingabe und Pflege gruppenöffentlicher Dienste

Das Rechenzentrum und das Juniorstudium stellten die Anforderung, dass Dozenten und eventuell auch Tutoren die einzigen Nutzergruppen mit der Berechtigung zum Hinzufügen und Bearbeiten von Diensten sind. Daher wurde im Stud.IP-Plugin ein Konfigurationsparameter genutzt, der festlegt, welche Nutzergruppen Rechte haben, die über das Nutzen der Dienstempfehlung hinausgehen.

Als Nutzerschnittstelle wird durch das Plugin ein Formular erzeugt, in dem die Tutoren bzw. Dozenten neue Dienste eintragen und die Empfehlungsregeln in Form von Restriktionen festlegen können. Dies ist in Abbildung 4.9 dargestellt.

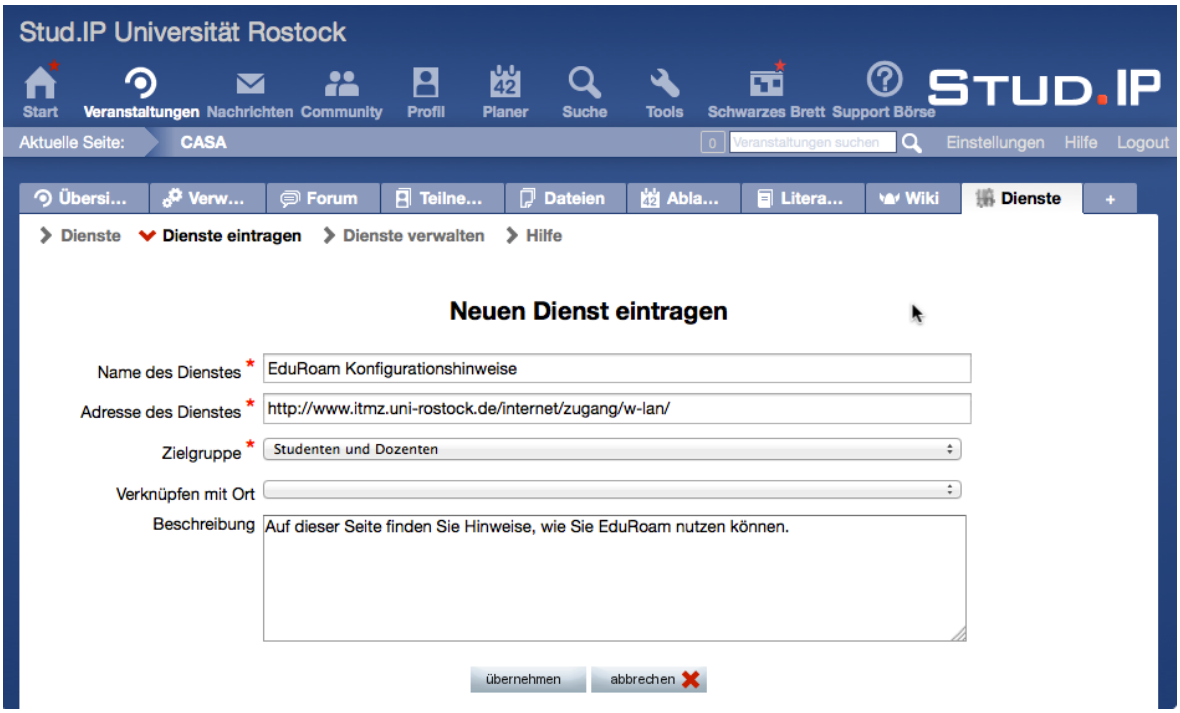

Abbildung 4.9: Formularbasierte Eingabe von Diensten und Verknüpfungen im CASA-Plugin im Stud.IP

Die Restriktionen beziehen sich dabei auf die Identität des Nutzers in Bezug auf seine Nutzerrolle und seine Aktivität, wiedergegeben durch die Veranstaltung. Abschließend kann auch der Ort, dessen potentielle Werte sich aus der Menge der Veranstaltungsorte ergibt, angeben werden.

Die hier umgesetzte Schnittstelle führte zu den Fragen, was den Nutzer zusätzlich motivieren könnte, Dienste mit der Gemeinschaft zu teilen und wie sich ortsbezogene Dienste besser als durch diese Formularstruktur hinzufügen und verwalten lassen. Existierende Systeme, wie zum Beispiel Foursquare, versuchen dies mit einer GPS-aktivierten, mobilen Applikation zu lösen und bieten Nutzern zusätzlich durch Gamification einen Anreiz aktiv zu sein. Für CASA war hier das Ziel zum einen die Entwicklung eines Gamification-Ansatzes, der geeignet ist auch über mehrere Domänen hinweg zu funktionieren und zum anderen die Implementierung einer Dienst- und Ortsverwaltung, die ebenfalls domänen- und nutzerübergreifend arbeitet und gut erweiterbar ist. Zu diesen Fragestellungen wurden studentische Arbeiten konzipiert und betreut. Die Ergebnisse werden in den nächsten beiden Abschnitten vorgestellt.

4.4 Weitere Realisierungen

Im Rahmen dieser Dissertation wurden neben den Kernkonzepten des CASA-Ansatzes auch exemplarisch Fragestellungen betrachtet, die sich durch die Realisierung ergaben. In den folgenden Abschnitten werden zwei von ihnen vorgestellt, die sich zum einen mit der Motivationsförderung der Nutzer und zum anderen mit der Verbesserung der Verwaltung von ortsbasierten Diensten befassen.

4.4.1 Motivation zur Interaktion auf gruppenöffentlichen Knoten: Gamification als CASA-Erweiterung

Die Frage nach der Motivation der Nutzer ist in dem Moment zu stellen, in dem die Eingabe der Dienste und Restriktionen nicht mehr Aufgabe eines Tutors oder Administrators sind. Diese Situation ergab sich in den umgesetzten Produktivumgebungen nicht, jedoch ist sie im Konzept vorgesehen, sobald es einem Nutzer möglich ist seine eigenen Dienste mit anderen zu teilen. Im Besonderen, wenn die Erstellung von Sensoren und eigenen Knoten zum Funktionsspektrum gehört.

Das Szenario für diesen Anwendungsfall war eine über CASA vernetzte Campusumgebung, in der der Nutzer motiviert werden soll, selbst Dienste für andere zu empfehlen. Des Weiteren sollten positive Verhaltensweisen gefördert werden, die sich aus den vorhandenen Kontextinformationen ergaben. Dies schließt zum Beispiel die pünktliche Rückgabe ausgeliehener Medien in der Bibliothek ein. Als Lösung wurde das CASA-System im Rahmen einer Bachelorarbeit um spieletypische Elemente erweitert. Die Ergebnisse wurden in [Wendt 13] publiziert.

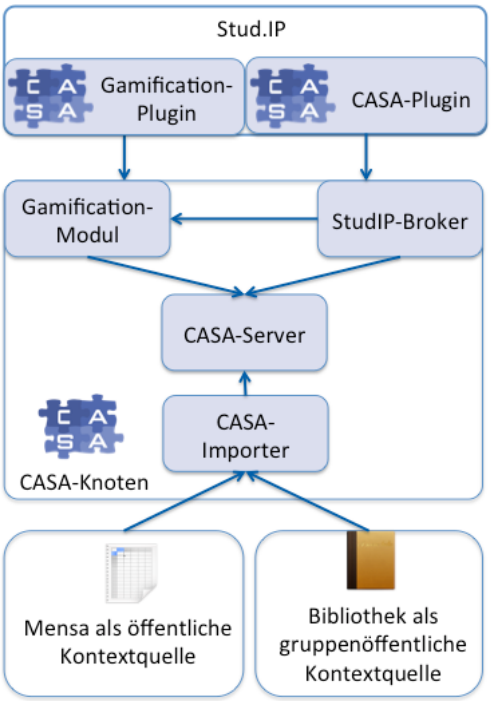

Abbildung 4.10: Bausteinsicht des Gamification-Systems

Realisiert wurde dies als Plugin im Stud.IP und durch ein Modul, das auf der Ebene der Aggregatoren im CASA-Knoten arbeitet. Dies ist in Abbildung 4.10 dargestellt. Das Gamification-Plugin ist dabei ein separates Plugin neben dem bereits vorhandenen CASA-Stud.IP-Plugin. Die Kommunikation zur Dienstnutzung erfolgt regulär zwischen dem CASA-Plugin und dem Stud.IP-Broker. Dieser wurde so erweitert, dass er bei Aktionen, die für die Gamification relevant sind, selbiges Modul informiert. Außerdem verfügt das Gamification-Modul über ei-

ne Schnittstelle zum CASA-Server, um weitere relevante Informationen aufzubereiten, die von Importern geliefert werden.

Abbildung 4.11: Screenshot des Stud.IP-Gamification-Systems mit virtuellen Auszeichnungen. Entnommen aus [Wendt 13]

Das Besondere an dieser Architektur ist, dass es so möglich ist, den Nutzer in einer Anwendung über virtuelle Belohnungen zu informieren, die er für Aktionen in dieser und anderen Domänen bekommen hat. Dies ist in den Abbildungen 4.10 und 4.11 dargestellt, in denen hypothetisch auch die Mensa und die Bibliothek als Kontextquellen genutzt werden. Das Gamification-Modul dient dabei als Broker, der über den eigenen CASA-Knoten als ersten Provider noch weitere Knoten anfragen kann. Somit muss die Logik zur Bewertung der Aktionen nur an einem Knoten existieren. Abseits davon kann natürlich auch auf verschiedenen Knoten ein Gamification-Modul arbeiten, das eigene virtuelle Belohnungen für die Nutzer verwaltet und dem CASA-Netz zur Verfügung stellt.

Quantifizierbare Ergebnisse auf die Frage, wie sehr sich durch diese Methode die Interaktionen mit dem CASA-Knoten steigern lassen, fehlen noch, jedoch unterstützen die nahtlose Integration in das System und die Ergebnisse anderer Studien zu Gamification[Hamari 14] die These, dass dieser Ansatz auch in diesem Umfeld geeignet ist, um die Motivation der Teilnehmer zu steigern.

4.4.2 Nutzerinteraktion bei öffentlichen Knoten: Openstreetmaps und Wikis als CASA-Erweiterung zur Vereinfachung der Nutzung

Sucht ein Nutzer ohne CASA an einem bestimmten Ort nach Diensten, ist er in der Regel auf werbegetriebene Ergebnisse von Suchmaschinen wie Google angewiesen. CASA selbst verfügt nicht direkt über eine Schnittstelle, um nach Diensten an einem Ort zu suchen, da die Organisation der Dienste domänenorientiert ist. Damit kann sich zwar ein Nutzer vom CASA-Knoten der Universität die Dienste des Veranstaltungsraums empfehlen lassen, jedoch bleiben dann eventuell nahe gelegene Dienste aus der außeruniversitären Umgebung unberücksichtigt. Es wurde deshalb zusätzlich zu der domänenorientierten Dienstverwaltung auch eine ortsorientierte Dienstverwaltung erstellt werden.

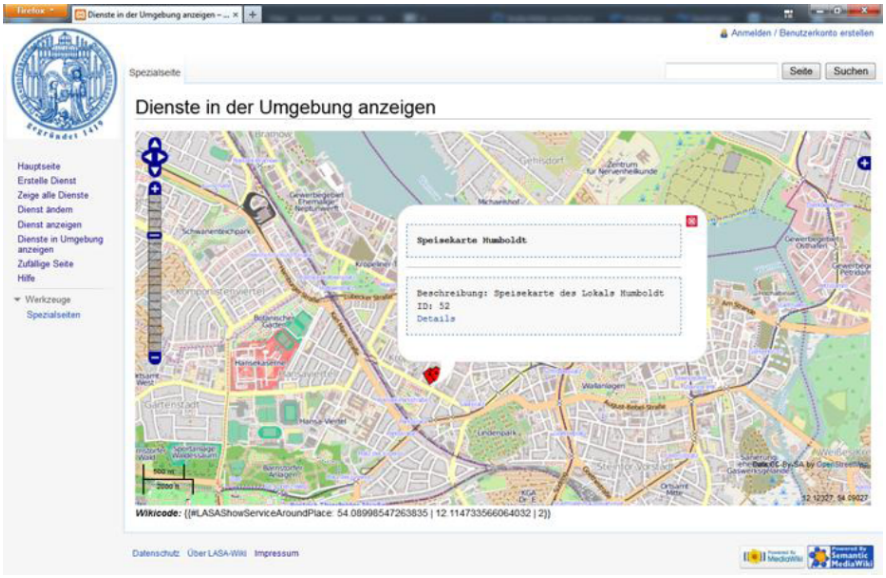

Abbildung 4.12: Screenshot des umgesetzten Systems mit Kopplung von MediaWiki und Openstreetmaps

Ziel dieser Umsetzung war es, ein System und eine Nutzerschnittstelle zu schaffen, die geeignet sind, das CASA-Szenario des öffentlichen Knotens zu unterstützen. Dieser soll von beliebigen Nutzern mit Orten und zugehörigen Diensten versorgt werden können.

Abschließend wurde eine Nutzerbefragung durchgeführt, die analysiert hat, ob Nutzer dieses System zum Suchen und Eingeben ortsbezogener Dienste verwenden würden. Dazu wurde im Rahmen einer Masterarbeit ein Konzept entwickelt, das die freie Wiki-Software MediaWiki und den freien Kartendienst Openstreetmaps (OSM) auf der Anwendungsebene verknüpft und so das Hinzufügen und Suchen von Diensten direkt auf einer Karte für Nutzer ermöglicht. In Abbildung 4.12 ist ein Screenshot der Anwendung wiedergegeben, der zeigt, wie ein Dienstfenster Informationen zu einer Standortabfrage darstellt. Die Umsetzung als CASA-Knoten war aufgrund der begrenzten Zeit im Rahmen der Masterarbeit nicht möglich, jedoch wurde die Architektur so angelegt, dass sich die Verarbeitungsebene in einer Web-Applikation kapselt, die auf der Ebene der Aggregatoren und der Broker arbeiten kann. Die Datenspei-

cherung erfolgt über eine SQL-Datenbank. Das Datenmodell ist so angelegt, dass es sich an dem Basiskontextmodell von CASA orientiert, was eine Integration erleichtert.

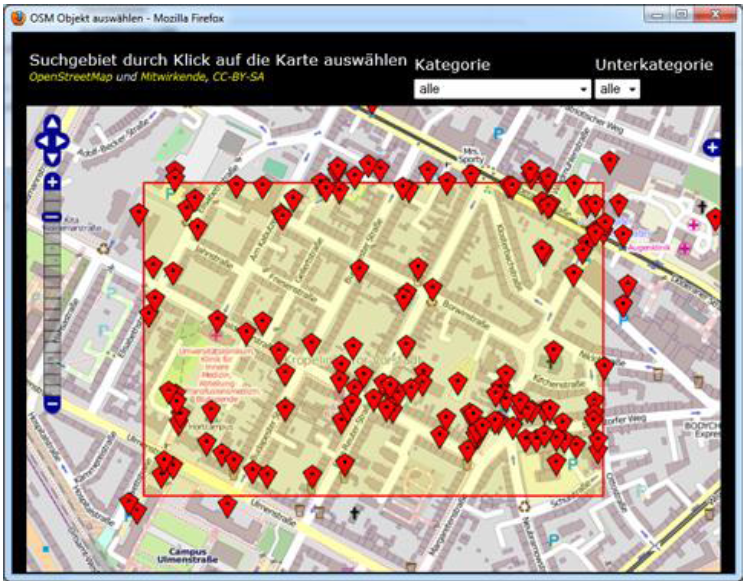

Abbildung 4.13: Screenshot des umgesetzten Systems mit Markierung der möglichen relevanten Orte im Innenstadtbereich von Rostock

Die Verknüpfung von Diensten mit Orten erfordert, dass bereits initial eine große Menge möglicher Orte vorhanden sein muss, denen die Nutzer Dienste hinzufügen können. Des Weiteren ist es notwendig, dass sich die Darstellung der Orte filtern lässt, um nicht die Übersicht zu verlieren. Dazu werden die im System angezeigten Orte auf jene beschränkt, die in Openstreetmaps bereits mit einer Kennzeichnung (engl. Tag) versehen sind, die dafür sprechen, dass sie für Nutzer interessant sind (z.B. Restaurants). In einem Datensatz wie der Karte von Rostock und Umgebung reduziert dies die Anzahl der Openstreetmaps-Objekte von etwa 1,3 Millionen auf etwa 18.000. Zur Illustration zeigt Abbildung 4.13 einen Innenstadtbereich von Rostock, in dem die Orte markiert wurden, die potentiell für Dienstkandidaten in Frage kommen. Dies umschließt Restaurants, Bars und andere Einrichtungen, die in unterschiedlichen Kontexten relevant sein können.

4.5 Zusammenfassung und Bewertung der Resultate der Implementierung

In diesem Kapitel wurde die Umsetzung des in Kapitel 3 konzipierten CASA-Systems beschrieben. Dazu wurde die Systemarchitektur in verschiedenen durch CASA umsetzbaren Varianten vorgestellt. Diese Varianten bilden die Konzepte des privaten, gruppenöffentlichen und öffentlichen Knotens ab. Damit wurde für alle Anwendungsfälle eine geeignete Systemarchitektur inklusive der notwendigen Rahmenbedingungen benannt.

Im Folgenden wurde die Softwarearchitektur vorgestellt. Dazu wurde ausgehend von der Definition von Qin auf die für das Design ausschlaggebenden Richtlinien und Bedingungen eingegangen, die sich durch die Anforderungen aus dem 2. Kapitel ergaben. Daher wurde besonders der Aspekt der Heterogenität der Endgeräte wie auch der Heterogenität der Dienstvielfalt adressiert. Die Skalierbarkeit stand ebenfalls im Fokus der Betrachtungen und wurde bei der Umsetzung an mehreren Stellen als Zielstellung genutzt.

Es wurde die Grundversion des CASA-Knotens mit den verschiedenen Modulen ausführlich beschrieben, wobei ausgehend vom CASA-Server als zentralem Modul die verschiednen Module in ihrer Umsetzung und in ihrer Verbindung zum Server erläutert wurden.

Anschließend wurde mit der technischen Umsetzung des Kontextmodells dargestellt, dass sich durch die Abbildung der Kontextfakten auf serialisierbare Java-Klassen ein erweiterbares und verteilbares Modell schaffen lässt. Exemplarisch wurden dazu auch die umgesetzten Erweiterungsklassen für das Stud.IP näher beleuchtet, die diese Sicht untermauern.

Neben der Grundvariante wurde auch eine Variante mit konkretem Anwendungsbezug realisiert. Die erfolgte Umsetzung als Gruppenknoten für das Stud.IP wurde ausführlich beschrieben, wobei hier mit der hybriden Architektur aus Produktiv- und Experimentalumgebung eine Struktur und Organisation geschaffen wurde, die im besonderen Maße für künftige Forschungsvorhaben einsetzbar ist, ohne auf separate Installationen zurückgreifen zu müssen. Zusätzlich wurden in studentischen Arbeiten weitere Realisierungen mit Fragen nach der Nutzermotivation und der Schnittstelle für die Eingabe ortsbasierter Dienste beschrieben.

Die dargestellten Realisierungen zeigen, dass das vorgeschlagene System der CASA-Knoten technisch umsetzbar ist. Besonders durch die Umsetzung für das Stud.IP konnte gezeigt werden, dass das CASA-System gut adaptierbar und im Feld von Campus-Anwendungen mit den verschiedenen Nutzerrollen anwendbar ist. Mit der Gamification-Erweiterung wurde ein Weg aufgezeigt, der die domänenübergreifende Dienstvermittlung durch ein domänenübergreifendes Motivationswerkzeug erweitert. Die Bewertung, wie sehr sich die Motivation beeinflussen lässt, steht noch aus, jedoch ist das Potential einen positiven Einfluss auf den Nutzer zu haben bereits über CASA hinaus nachgewiesen. Die Entwicklung zur ortsbasierten Empfehlung von Diensten zeigte, dass die Verwendung eines öffentlichen CASA-Knotens durchaus einen Bedarf der Nutzer darstellt, der bisher noch nicht gedeckt wurde. Die Möglichkeit, anonym auf ortsverknüpfte Dienste zuzugreifen, ist ebenfalls neu.

Nachdem in diesem Kapitel die Realisierungen vorgestellt wurden, wird im nächsten Kapitel genauer auf die Evaluation des umgesetzten CASA-Systems eingegangen, wobei der Fokus auf der Nutzerbefragung im Zusammenhang mit der Stud.IP-Umsetzung für das Juniorstudium liegt.

Kapitel 5

Evaluation der nutzergetriebenen Dienstverteilung mit CASA

Die im letzten Kapitel dargestellte Realisierung des CASA-Systems erfolgte sowohl unter dem Aspekt der Implementierung des Konzeptes in Form eines Demonstrators als auch unter dem Aspekt einer Realisierung für eine Produktivumgebung. Dabei wurde gezeigt, dass die Konzepte umsetzbar sind und sich im Anwendungsfall der Universität praktisch nutzen lassen.

Das Ziel der Evaluation ist es zu zeigen, dass die in dem Konzept und der Implementierung vorgestellte Lösung die in Abschnitt 2.6 formulierten Anforderungen adressiert. Dabei wird im Einzelnen diskutiert, wie dies hier erfolgt und wo Potential für weitere Entwicklungen und Erweiterungen liegt. Da die Beteiligung der Nutzer für dieses System essentiell ist, wird deren Haltung gegenüber dem System und dem Konzept sowohl durch Befragungen als auch durch Auswertung von anonym erhobenen Nutzerdaten ermittelt.

Das Kapitel beginnt mit einer Ausführung, warum das Konzept des gruppenöffentlichen Knotens als Demonstrator umgesetzt wurde und welche Vor- und Nachteile sich dadurch ergaben. Anschließend werden die technischen, konzeptionellen und sozialen Anforderungen überprüft. Im nächsten Komplex werden die durchgeführten Fallstudien vorgestellt und es wird diskutiert, welche Schlussfolgerungen sich daraus für andere Anwendungsfälle ergeben. Abschließend werden in einem Ausblick weitergehende Evaluationsmöglichkeiten diskutiert und es wird gezeigt, in welchen Bereich diese noch weitere Einsichten liefern können.

5.1 Evaluation entsprechend der Anforderungsanalyse

Die Evaluation des CASA-Konzepts untersucht verschiedene Bereiche und Aspekte des Systems. Diese ist nicht auf eine rein konzeptionelle oder theoretische Analyse beschränkt, sondern berücksichtigt auch die Eigenschaften, die sich erst durch die Implementierung ergaben. Zudem wurden zwei Fallstudien durchgeführt und bewertet, die sich im Speziellen mit der

Motivation der Nutzer zur passiven Verwendung und zur aktiven Teilnahme an einem System befassen, das auf dem CASA-Konzept aufbaut.

Zunächst wird überprüft, welche der drei Varianten (A, B oder C entsprechend Abschnitt 4.1.1) sich für eine Evaluation unter den beschränkten Ressourcen an Zeit und Aufwand eignen. Anschließend werden die Anforderungen aus 2.6 mit dem Konzept und der Umsetzung verglichen.

5.1.1 Auswahl eines Knotenkonzeptes für die Demonstratoren

Um das CASA-Konzept unter realistischen Bedingungen evaluieren zu können, musste ein geeignetes Szenario gefunden werden. Mit dem Juniorstudium der Universität Rostock wurde sowohl eine für das System interessante Plattform als auch eine passende Zielgruppe für das System gefunden. Die Mitarbeiter des Projektes unterteilen sich in die Koordinatoren, die die Verantwortung für das Projekt auf inhaltlicher und technischer Ebene haben, und die Tutoren, die als studentische Hilfskräfte aus verschiedenen Fachrichtungen Veranstaltungen aufzeichnen, in das System einstellen und auch Präsenzveranstaltungen durchführen. Teilnehmer des Projektes sind Schüler aus ganz Deutschland, die neben der gymnasialen Oberstufe bereits Hochschulkurse absolvieren. Die Kurse werden nach dem Modell des Blendet Learning angeboten und verknüpfen so den regelmäßigen Konsum von Vorlesungsaufzeichnungen mit der tutorgestützten Lösung von Übungen und einer geringen Zahl an Präsenzveranstaltungen. Mit diesem Lern-Szenario aus dem universitären Bereich wurde eine heterogene Nutzermenge gefunden, da die Tutoren und Lernenden aus verschiedenen Fachrichtungen stammen. Auch eine Motivation der Nutzer zur Interaktion mit dem System wurde sichergestellt, da es eine technische Innovation mit gesteigertem Funktionsumfang zu den Vorjahren darstellte, in denen die Videos lediglich als Verlinkung in einem Forumsbeitrag veröffentlich wurden. Mit dem CASA-System bot sich nun die Möglichkeit Videos direkt in das Stud.IP einzubinden und dort abzuspielen. Außerdem konnten durch die Offenheit des Systems eine Vielzahl an weiteren Diensten eingebunden werden. Offen blieb dabei jedoch der Umfang des umzusetzenden CASA-Systems.

Gegen die Evaluation des Konzeptes des privaten Knotens sprachen verschiedene Gründe. Die wichtigsten waren die Anforderungen an den Daten- und Privatsphärenschutz, bei gleichzeitiger Aufzeichnung von Daten zur Auswertung. Der Aufwand, um die Sicherheit zu gewährleisten, stand in keinem Verhältnis zu der zur Verfügung stehenden Zeit. Außerdem wäre ein solcher Test nicht mit den häufigen Entwicklungs- und Wartungszyklen eines Demonstrators zu vereinen gewesen. Abgesehen davon wäre fraglich gewesen, ob für die Anwender ohne zur Verfügung stehende gruppenöffentliche und öffentliche Knoten ein Nutzen bestanden hätte.

Der Evaluationsnutzen eines reinen öffentlichen Knotens ist ebenfalls begrenzt, da hier die Nutzer in erster Linie konsumieren und so der CASA-Aspekt hinsichtlich der nutzergetriebenen Produktion und Verteilung von Dienstempfehlungen nicht zur Geltung kommt.

Das Argument, dass ein Nutzen ohne private und öffentliche Knoten nur sehr begrenzt ist, trifft auf gruppenöffentliche Knoten auch zu. Jedoch ist hier eine Nutzung durch die Einbettung in eine Mehrbenutzeranwendung möglich. Im vorliegenden Fall kann durch eine

Anwendung im Stud.IP auch von weiteren Vorteilen profitiert werden. Die Sicherheitsbedenken durch böswillige Manipulation des Systems sind weniger kritisch, da die Nutzer sich gegenüber dem Stud.IP bereits eindeutig mit ihrem Nutzerkonto identifizieren müssen.

Über eine kurze Einweisung konnten alle Tutoren des Juniorstudiums für das System geschult werden und waren so in der Lage es schneller zu nutzen. Außerdem liegt hier auch der Vorteil in der Notwendigkeit zur Verwendung des Systems vor, da das Videoangebot des Juniorstudiums nur über das CASA-System zu erreichen war, was die Nutzerbeteiligung und die Intensität der Auseinandersetzung mit dem System erhöhte. So können die Nutzergruppen durch die im Stud.IP verankerten Rollen gut getrennt werden. Des Weiteren war es möglich im Juniorstudium differenzierte Befragungen durchzuführen.

Die Gegenargumente sind zum einen, dass die Nutzung sich an der Domäne des Juniorstudiums orientiert und somit von Seiten der Dozenten und Tutoren eine einmalige Aktivität in der Woche zu erwarten ist. Auch bei den Studenten und Schülern wird die Nutzung einmal die Woche kaum übersteigen, da hier das Betrachten der Vorlesungsaufzeichnungen das zentrale Element ist. Es ist anzunehmen, dass sich die Art der zur Verfügung gestellten Dienste auf die Videos konzentrieren wird. Ein weiterer Aspekt ist hier, dass auch die Regelkomplexität unter den Möglichkeiten bleiben wird, die CASA bieten kann, da die Zuordnung eindeutig nach den Kontexten Aktivität und Nutzergruppe erfolgt. Vom technischen Aspekt her erfordert diese Methode eine durchdachte Architektur, die Ausfallsicherheit und mindestens eine CASA-Emulation bietet.

Basierend auf diesen Überlegungen wurde die Entscheidung getroffen einen Stud.IP Gruppenknoten zu realisieren. Neben dem Juniorstudium wurde mit dem Rechenzentrum der Universität ein weiterer kompetenter und kritischer Partner gewonnen, der die Entwicklung aus technischer und konzeptioneller Sicht begleitet hat. Das Hauptszenario für diese Evaluation ist daher die nutzergetriebene Vermittlung von Diensten in einem Campusumfeld. Dies wurde mit zwei Stud.IP-Installationen durchgeführt, zum einen dem Stud.IP des Juniorstudiums und zum anderem dem offiziellen Stud.IP der Universität Rostock. Neben diesem Szenario wurden noch weitere Evaluationen durchgeführt, die sich in erster Linie auf die Nutzerakzeptanz und Motivation bezüglich eines solchen Systems beziehen. Diese werden in Abschnitt 5.2.1 erläutert. In den nächsten Abschnitten werden zunächst die konzept- und die umsetzungsspezifischen Systemeigenschaften diskutiert.

5.1.2 Evaluation der Systemeigenschaften des CASA-Konzeptes

Der Fokus der konzeptionellen Anforderungen an das System liegt auf dem Kontextmodell. Dieses soll nicht nur skalierbar sein, sondern auch robust gegenüber Erweiterungen und Veränderungen. Neben dem Kontextmodell entscheidet die Struktur des Systems darüber, wie austauschbar die Sensormodelle sind. Diese Struktur bestimmt auch, ob die Idee des Privacy-by-Design umgesetzt wird, indem ein monodirektionaler Datenfluss private Daten schützt.

Verteiltes, skalierbares Kontextmodell mit Unterstützung dezentraler Entwicklung

Das im Konzept vorgeschlagene und in der Implementierung umgesetzte Kontextmodell unterstützt eine dezentrale Weiterentwicklung. Durch die Aufteilung in einen Basissatz mit Kontextklassen, von denen sich alle anderen ableiten lassen, wird den Entwicklern ein gemeinsames Fundament gegeben. Dies sichert eine Kommunikation auch über Domänengrenzen hinweg, indem die Objekte über ihre Basisklassen interpretiert werden können. Die Aufteilung erlaubt es für jede Domäne ein eigenes Kontextmodell zu erstellen, das innerhalb der Domäne an alle Anforderungen angepasst ist. Werden diese Klassen in einem Knoten außerhalb der Domäne benötigt, so lässt sich entweder das Kontextmodell importieren, um es dort nativ zu nutzen, oder es wird ein Importer benötigt, der die Transformation in Fakten des vorhandenen Kontextmodells realisiert.

Nachhaltige Unterstützung logischer, virtueller und physischer Sensoren

Das Konzept der Importer ist der Ansatz zur Unterstützung von Sensoren. Diese als Software umgesetzten, kleinen Programme sind sensorspezifisch und konvertieren die Sensordaten in eines der Basisformate. Anschließend werden diese Basisformate in das domänenspezifische Kontextmodell überführt. Dies ermöglicht die Austauschbarkeit der Sensorschnittstellen zwischen verschiedenen Domänen. Die Lizenzierung von CASA unter der Apache 2.0-Lizenz unterstützt ebenfalls die nachhaltige Verbreitung.

Monodirektionaler Datenfluss zur Sicherung schutzwürdiger Daten nach dem Konzept des Privacy-by-Design

Die in Abschnitt 3.3.3 konzipierte Struktur verlangt, dass die Aggregation von Daten stets auf dem Knoten erfolgt, der den höheren Grad an Privatsphäre hat. Zwischen einem öffentlichen und einem gruppenöffentlichen Knoten wäre dies der gruppenöffentliche. Aus einem privaten Knoten werden nur Daten weitergegeben, die notwendig sind, um einen Nutzer zu identifizieren, oder die für eine Suchanfrage benötigt werden. Fragt ein privater Knoten externe Knoten an, so werden die Ergebnisse erst an den privaten Knoten übermittelt und von diesem dort aggregiert. Der Nutzer konfiguriert in privaten Knoten selbstständig seine Regeln und entscheidet dabei mit, welche Anfragen von seinem Knoten ausgehen. In gruppenöffentlichen Knoten werden Anfragen unabhängig von der Person gestellt und erlauben so nur begrenzt Rückschlüsse auf die Identität.

5.1.3 Evaluation der Eigenschaften der CASA-Umsetzungen

Technische Anforderungen an CASA sind abhängig von der konkreten Umsetzung. Daher werden hier nur Eigenschaften bewertet, die sich durch die Umsetzung der Demonstratoren ergaben. Mit weiteren Umsetzungen, die sich ebenfalls nach dem CASA-Konzept richten, könnten auch andere Eigenschaften erreicht werden.

Zentrale Vereinheitlichung auf Ebene der Dienste

Die im Rahmen der Umsetzung angestrebte Vereinheitlichung der Dienste auf einer Ebene konnte nur bedingt erreicht werden. Auf Grund der begrenzten zeitlichen Ressourcen dieser Arbeit wurde der Dienstbegriff auf die als Webseite darstellbaren Schnittstellen beschränkt. So konnte für die Demonstratoren ein Kompromiss aus Funktionalität und Aufwand erreicht werden. Jedoch geht das Konzept weiter und würde hier gerade auf den mobilen Endgeräten auch die Einbindung von Apps und Systemfunktionalitäten, wie der Steuerung der Netzwerkschnittstelle, einschließen. Auf dem stationären Endgerät wäre hier neben der installierten Software auch die Peripherie-Hardware zu nennen, die Dienste für CASA zur Verfügung stellen könnte. Dies ist jedoch noch nicht Teil der Umsetzungen von CASA und daher nur Gegenstand potentieller Weiterentwicklungen.

Plattformunabhängigkeit durch Transformation zu einer browserfähigen, graphischen Schnittstelle

Die Erstellung von Schnittstellen in Form von Webseiten für die Dienste kann als universell anwendbarer Weg bezeichnet werden. Die Probleme, die hier auftraten, waren jedoch teilweise unerwartet schwerwiegend. Zentral ist hier im Umfeld des Stud.IP-Knotens die „Same Origin Policy“ zu nennen, die dem Konzept des Einbindens externer Inhalte die Hürde auferlegt, dass diese mindestens dasselbe Protokoll, teilweise auch denselben Port, Hostnamen und dieselbe Subdomain nutzen müssen. So kam es im SSL gesicherten Stud.IP der Universität zu Problemen, wenn Inhalte von anderen, nicht gesicherten Quellen kamen, also zum Beispiel mit dem Protokoll HTTP statt HTTPS.

Umsetzung als verteiltes System mit effizienter Kommunikation innerhalb von Domänen und mit nutzereigenen Knoten

Die Strukturierung des Systems mit domänenspezifischen und privaten Knoten erlaubt eine sinnvolle Organisation innerhalb von Domänen. Dabei kann durch das in Java modular organisierte Kontextmodell auch ein komplexes System mit Domänen und Subdomänen erstellt werden. Es können einzelne Knoten zusätzlich zu externen auch weitere, eigene Kontextmodellerweiterungen nutzen. Dabei werden lediglich die Java-Bibliotheken der jeweiligen Kontextmodelle und Erweiterungen im Knoten verfügbar gemacht, sodass dieser in der Lage ist, Regeln auf die Fakten dieser Modelle anzuwenden. Dies ist für die Umsetzung hinreichend effizient, jedoch erfordert es ein manuelles Eingreifen bei Aktualisierungen. Daher wäre für künftige Entwicklungen zu überprüfen, ob sich Konzepte wie OSGI zur dynamischen Einbindung von Bibliotheken in diesen Domänen eignen würden.

Implementierung unter Nutzung von Systemen und Softwarekomponenten mit geeigneten Lizenzen und einer aktiven Entwicklergemeinde

Die Umsetzung erfolgte ausschließlich mit Softwarekomponenten, die unter einer freizügigen Lizenz stehen. So findet die freizügige Apache 2.0-Lizenz Anwendung beim wissensbasierten System Drools[1], ebenso wie bei JDOM[2], Apache Camel[3] oder auch bei Vaadin[4], dem Framework zur Erstellung der Oberflächen. Lediglich für die Gestaltung der mobilen Oberfläche wurden in der Implementierung Abstriche gemacht, da die Bibliothek Vaadin TouchKit für den nicht kommerziellen Einsatz eine AGPL vorsieht[5]. Diese ist in dem Punkt restriktiver als die Apache 2.0-Lizenz, dass sie ein strenges Copyleft enthält. Dies bedeutet, dass für das Modul CASA-Mobile andere Weiterverbreitungsbedingungen gelten als für die anderen Module und dass hier eine Lizenzierung unter Apache 2.0 nicht möglich ist.

Nutzung einer Kommunikationsinfrastruktur, die geeignet ist über Netzwerkgrenzen hinweg effizient Daten zu übertragen

Die Verwendung einer standardisierten, nachrichtenorientierten Middleware, hier Apache Camel, hat sich als sinnvoll erwiesen. Die Kommunikation über Netzwerkgrenzen hinweg ist bei entsprechender Konfiguration des Netzwerkes möglich und die Verknüpfung der Module zu einem Knoten erfolgte problemlos. Gleichzeitig können durch diese Infrastruktur auch Komponenten angebunden werden, die in einer anderen Programmiersprache umgesetzt wurden.

Einbindung der Dienstempfehlungen in den Software Workflow des Nutzers

Um die Einbindung in den Workflow zu ermöglichen, ist die vorgeschlagene Methode mit der Einbettung von iFrames ein möglicher Weg. Dieser verlangt jedoch, dass die Anwendung des Nutzers eine Schnittstelle dafür vorsieht oder in diesem Aspekt erweiterbar ist. Für andere Anwendungsfälle sind andere GUIs notwendig, die dann z.B. als native Software auf dem Endgerät des Nutzers ausgeführt werden und den Nutzer über mögliche Assistenz für seine Situation informieren. Diese Plugin-Erstellung lässt sich nicht auf ein Framework oder eine Technik begrenzen. Jedoch bietet der Weg der iFrame-Einbettung einen geeigneten Ausgangspunkt.

Unterstützung mobiler Geräte als Endpunkte der Dienstempfehlung

Die erstellte mobile Schnittstelle zeigt, dass die Umsetzung für Smartphones und Tablets durch bereits existierende Bibliotheken möglich ist. Wenn die Lizenz des verwendeten Vaadin-Frameworks nicht für den jeweiligen Anwendungsfall geeignet ist, kann auch eine andere

[1] `http://www.drools.org/code/license.html` [Online letzter Zugriff 15.02.2018]
[2] `http://www.jdom.org/docs/faq.html` [Online letzter Zugriff 15.02.2018]
[3] `http://camel.apache.org/what-is-the-license.html` [Online letzter Zugriff 15.02.2018]
[4] `https://vaadin.com/license` [Online letzter Zugriff 15.02.2018]
[5] `https://vaadin.com/add-ons/touchkit` [Online letzter Zugriff 15.02.2018]

Technik, wie z.B. PhoneGap[6], in Erwägung gezogen werden. Die kompakte Datenübertragung und die Entkopplung von GUI und Server erwiesen sich als gut geeignet.

5.1.4 Evaluation der Systemeigenschaften in Bezug auf die sozialen und organisatorischen Anforderungen

Diese sozialen und organisatorischen Anforderungen umfassen jene, die einen Einfluss darauf haben, wie das System durch den Nutzer wahrgenommen wird und ob eine Weiterentwicklung stattfinden kann.

Kontrolle des Datenflusses durch den Nutzer

Dem Nutzer die Kontrolle über seine Daten zu geben war eines der zentralen Ziele von CASA. Dies ist gelungen, indem der Nutzer die personenbezogenen Daten auf seinem privaten Knoten hält. Das Konzept sieht explizit vor, dass die Aggregation von personenbezogenen Dienstkandidaten nur auf dem privaten Knoten erfolgen darf. Da ein Datenabfluss nur bedingt verhindert werden kann, soll der Nutzer selbst die Datenfreigabe für Anfragen an gruppenöffentliche und öffentliche Knoten erteilen. Es ist Teil der Selbstbestimmung des Nutzers, dass dieser weiß, welche Informationen übertragen werden, wenn eine Anfrage gestellt wird.

Dennoch ist der Nutzer darauf angewiesen, dass bei der Wahl der gruppenöffentlichen Knoten seine Daten schützt, indem er auf vertrauenswürdige Betreiber und Datenschutzerklärungen achtet. Das CASA-Konzept kann böswillig herbeigeführte Datendiebstähle nicht verhindern. Daher ist hier eine stete Weiterentwicklung und Stärkung der konzipierten Ideen essentiell. So sind verschlüsselte Verbindungen zwischen den Knoten nur ein erster Schritt. Auch Konzepte zur Zertifizierung von vertrauenswürdigen Knoten und zur Authentifizierung zwischen den Knoten selbst sowie zwischen den Knoten und Nutzern sind noch nicht Teil des CASA-Konzeptes.

Verteilung des Wartungsaufwandes auf verschiedene Nutzerrollen

Die Wartung des Systems umfasst verschiedene Aufgaben. Während die rein technische Instandhaltung der Knoten den Administratoren des jeweiligen Knotens obliegt, kann die Qualitätssicherung der Daten von den jeweiligen Nutzern übernommen werden. Da es sich dabei nicht um ein klassisches Verhältnis zwischen einem Anbieter und vielen Nutzern, sondern um ein Verhältnis zwischen verschiedenen Prosumern, also der Einheit von Producer und Consumer, handelt, ist der Aufwand nicht bei einer Person konzentriert.

[6] `https://phonegap.com/` [Online letzter Zugriff 15.02.2018]

Integration von geeigneten Motivationskonzepten zur Beteiligung der Nutzer

Die Motivation der Nutzer zur Beteiligung am System lässt sich bei CASA auf zwei Ziele aufteilen, zum einen das System aktiv zu erweitern und neue Inhalte hinzuzufügen, zum anderen das System als solches zu nutzen und zu überprüfen, wie sich der eigene Arbeitsablauf mit CASA verbessern lässt. Für das erste Ziel wurde mit dem Gamification-Konzept eine Methode erarbeitet, die sich innerhalb einer Domäne über mehrere Knoten hinweg umsetzen lässt und so ein umfassenderes Nutzererlebnis bieten kann als eine einzelne Aktion [Wendt 13]. Um den Nutzer zu motivieren seinen Arbeitsfluss mit CASA zu unterstützen, ist es notwendig, dass CASA einen Mehrwert gegenüber der nicht kontextuell unterstützen Arbeitsweise liefert. Dies ist jedoch bedingt dadurch, dass es ausreichend viele aktuelle und adequate Inhalte gibt, die den Nutzer in seinem Arbeitsprozess unterstützen können. Hier kann nur indirekt auf die Nutzer eingewirkt werden, indem z.B. der Autor eines Dienstes für die häufige Nutzung durch Dritte belohnt wird. So kann dieser motiviert werden, sich weiterhin aktiv zu beteiligen und weitere relevante Inhalte zur Verfügung zu stellen.

Veröffentlichung des Systems und seiner Schnittstellen unter Lizenzen, die zur Erweiterung einladen und den Schutz von Eigenentwicklungen bieten

Die Wahl der Lizenz, unter der dieses System veröffentlicht wurde, fiel mit der Entscheidung für die Apache 2.0 Lizenz auf eine sehr freizügige Lizenz. Diese erlaubt es externen Entwicklern und interessierten Nutzern das System für eigene Zwecke anzupassen, ohne dass sie damit weitergehende Verpflichtungen eingehen. Lediglich die Namensnennung ist bei Weiterverbreitung zwingend. So können Eigenentwicklungen geschützt werden, ohne dass für die Weiterentwicklung Hürden aufgebaut werden.

5.2 Bewertung des Konzeptes und der Realisierung anhand durchgeführter Fallstudien

Die folgenden Fallstudien beleuchten genauer die Evaluationsaspekte, die sich erst durch eine Realisierung und den Kontakt mit Nutzern ergaben. Sie zeigen, dass CASA ein Ziel bearbeitet, das durch die Nutzer sowohl nachgefragt als auch gerne aktiv genutzt wird. Dabei wurden anhand von anonym erhobenen Nutzerdaten und Befragungen Rückschlüsse auf die Eignung von CASA für den Einsatz in verschiedenen hochschulnahen Szenarien gegeben.

5.2.1 CASA@StudIP: Von der Integration von Smart-Environment-Diensten in Altanwendungen zur nutzergetriebenen Integration von beliebigen Diensten in eine Lehr- und Lernumgebung

Ein Anwendungsfall von CASA ist die nachträgliche Integration von Diensten in bestehende Anwendungsumgebungen. Diese Altanwendungen sind heute allgegenwärtig und feste Be-

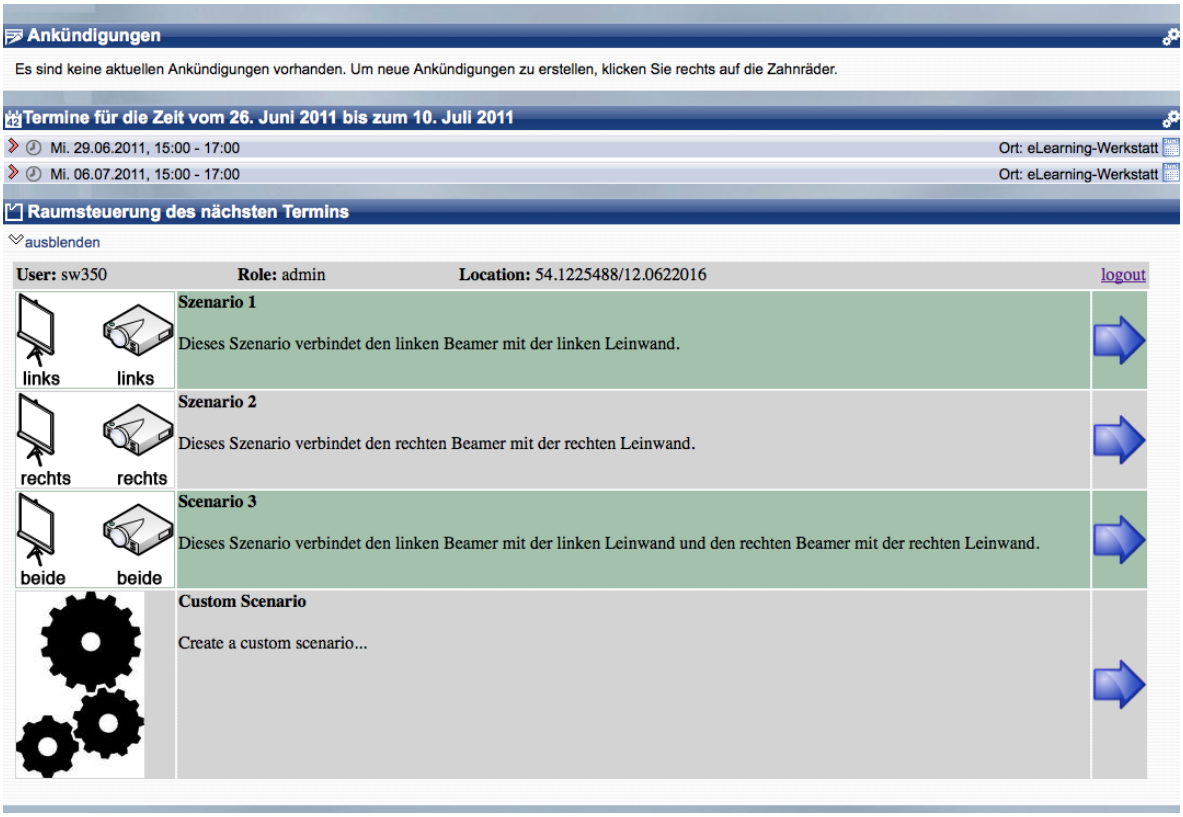

Abbildung 5.1: Screenshot mit der Integration einer intelligenten Lehrumgebung in das Stud.IP als Altsystem in einem Campusumfeld. Entnommen aus [Lehsten 11a]

standteile der Arbeitsabläufe ihrer Nutzer. Sie zeichnen sich im Besonderen dadurch aus, dass sie sehr langlebig sind und nicht schnell ersetzt werden können. Dies ist dadurch bedingt, dass sie einerseits essentielle, komplexe Aufgaben erfüllen und gleichzeitig einen hohen Einarbeitungsaufwand sowohl durch die Administratoren als auch durch die Nutzer erfordern. Neue Dienste und Geräte bedürfen ihrerseits häufig einer Konfiguration, die die Nutzung erst ermöglicht, was zu einer Vielzahl von separaten Applikationen nebeneinander führt, wobei jede nur einen Aspekt der Altanwendung unterstützt.

In einem ersten Prototypen bestand der Ansatz darin, mit dem Stud.IP eine typische Altanwendung im universitären Umfeld so zu erweitern, dass sie eine intelligente Raumsteuerung situationsbezogen einbinden kann. So sollte der Konfigurationsaufwand, der sonst abschreckend wirkt, verringert und nebenbei Zeit gespart werden, da die Umgebung direkt aus dem Stud.IP heraus mit dem Computer des Dozenten gesteuert werden kann. Das Ergebnis war ein Demonstrator, der einige Konzepte von CASA bereits vorwegnahm und so eine Basis für weitere Entwicklungen lieferte [Lehsten 11a]. Abbildung 5.1 illustriert einen Ausschnitt des umgesetzten Systems, das es am Ende ermöglichte, in einer speziellen Laborumgebung, der E-Learning-Werkstatt, über das Stud.IP die Geräte wie Beamer und Leinwände zu steuern.

Die Einbettung von CASA in das Stud.IP, die bereits unter dem Aspekt der Implementierung in Abschnitt 4.3 beschrieben wurde, stellte den größten Evaluationskomplex dar. Durch die

Aufteilung in die Produktiv- und Experimentalumgebung konnte einerseits ermittelt werden, wie sich die technischen Anforderungen bei einer vollständigen CASA-Implementierung umsetzen lassen und andererseits, wie ein solches System von seinen Nutzern wahrgenommen und verwendet wird. Da der technische Aspekt bereits in Abschnitt 5.1.3 erläutert wurde, wird er im Folgenden ausgespart.

Im Rahmen des Juniorstudiums war es möglich zu beobachten, wie die Tutoren Dienste in das System einstellen. Dazu besuchen die Tutoren die Vorlesungen an der Universität und zeichnen den Dozenten und seine Folien synchron auf. Diese Aufnahmen werden dann von ihnen in ein einzelnes Video konvertiert und auf einem Server der Universität bereitgestellt. Abschließend werden die Videos als Dienste im Stud.IP über CASA verfügbar gemacht. Vor dieser Umsetzung wurden lediglich Links in eine Liste im Stud.IP eingefügt. Mit CASA erfolgt eine Einbettung der Videos direkt in den Seiten der Veranstaltung, was das Anschauen direkt in dem Browserfenster ermöglicht.

Die Verwendung des Systems erfolgte intensiv. Im Evaluationszeitraum von 98 Tagen wurden die Interaktionen mit dem System anonymisiert protokolliert und neben dem Zeitpunkt der Interaktion wurde auch erfasst, mit welchen Diensten interagiert wurde und zu welcher Rolle der Nutzer gehörte. Insgesamt wurde von den Tutoren 170 mal ein Dienst hinzugefügt, 49 mal wurden Dienste geändert und 27 mal gelöscht. Die Studenten erzeugten 1369 Dienstaufrufe. Basierend auf diesen Zahlen kann gesagt werden, dass die Nutzer das System ausgiebig nutzten und sich daher über das Stud.IP-Plugin und diesen Aspekt der Integration von externen Diensten in eine gruppenöffentliche Anwendung eine fundierte Meinung bilden konnten.

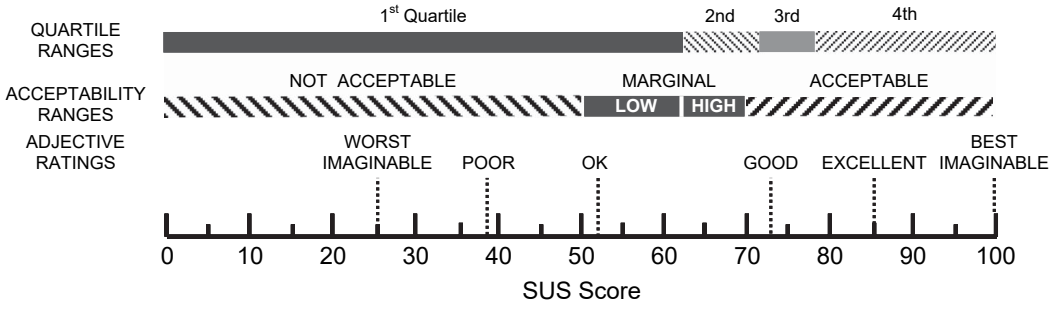

Abbildung 5.2: Quantitative Verteilung von SUS Scores mit möglichen Interpretationen und Grenzwerten. Entnommen aus [Bangor 08]

Die empfundene Nutzerfreundlichkeit einer Webseite bedingt die Zufriedenheit eines Nutzers mit der Webseite und letztendlich auch das Vertrauen, das ein Nutzer einer Webseite entgegenbringt[Flavián 06]. Daher musste für die Bewertung der Aussagen die empfundene Nutzbarkeit zumindest ansatzweise ermittelt werden, um sicherzustellen, dass die Aussagen nicht durch eine sehr niedrige Nutzerfreundlichkeit übermäßig beeinflusst werden. Um die Nutzerfreundlichkeit des umgesetzten Systems zu ermitteln, wurde mit den Tutoren eine Befragung zum CASA-Plugin im Stud.IP durchgeführt. Dabei wurde unter anderem der Software Usability Scale Score (SUS Score) ermittelt [Brooke 96]. Dieser basiert auf 10 Fragen zur Anwendung, die die Nutzer mit Hilfe einer fünfstufigen Liekert-Skala (Stimme gar nicht zu (0) - Stimme vollständig zu(4)) beantworten. Die Antworten werden anschließend

in Punktwerte (0 - 4) umgerechnet, addiert und mit dem Faktor 2,5 multipliziert, um am Ende einen SUS Score zwischen 0 und 100 zu ergeben. Abbildung 5.2 aus [Bangor 08] zeigt eine mögliche Interpretation sowie die Verteilung des SUS Scores über 2324 Umfragen in 206 Studien hinweg. Ein SUS Score von über 52% markiert den unteren Grenzwert und spricht damit für eine Anwendung ohne massive Mängel. Dies war das Mindestziel, das zu erreichen war. Leider ist die Zahl der Tutoren insgesamt relativ gering, und da auch hier nicht alle an dem Test teilgenommen haben, ergab die Auswertung der 4 Fragebögen kein belastbares Ergebnis, sondern nur eine Tendenz. So reichen die ermittelten SUS Werte von 45 bis 80. Im Median, der etwas die Wirkung der Ausreißer vermindert, ergab sich ein Wert von 63,75.

Von Seiten der Administratoren des Juniorstudiums war vorgegeben, dass die Tutoren regelmäßig die Videos der Veranstaltungen einpflegen sollten. Darüber hinaus wurde ihnen auf einer Demonstrationsseite gezeigt, dass sich neben Videos auch andere Dienste, wie webbasierte Compiler, Wikipedia-Artikel und Online-Atlanten einbinden lassen. Über den Evaluationszeitraum von 98 Tagen zeigte sich, dass nach den Demonstrationsdiensten lange nur noch Videos eingetragen wurden. Zum Ende hin wurde die Vorlesungsevaluation über Google-Docs durchgeführt, das sich ebenfalls über CASA direkt in den Veranstaltungen integrieren lässt. Dies ist ebenfalls ein Beispiel für die Integration in eine Altanwendung. So verfügt das Stud.IP durchaus über eine umfangreiche und komplexe Evaluationskomponente. Jedoch wurde durch die Anwender die Nutzung von CASA und damit die Integration eines aus Nutzersicht besser geeigneten Dienstes in den Arbeitsablauf hier vorgezogen.

Das Plugin begrenzte die möglichen Kontextbedingungen für das Einbinden eines Dienstes auf die Veranstaltung, die Rolle der Nutzer (Dozent/Juniorstudent) und den Veranstaltungsort. Letzterer war nicht relevant, da es sich ausschließlich um Online-Studenten handelte. Damit wurden drei der vier möglichen Kontextdimensionen von Dey abgedeckt. Befragt nach ihrem Wunsch für weitere mögliche Kontextbedingungen antworteten drei der vier Befragten, dass sie keinen weiteren Bedarf sehen. Einer sprach sich für eine zeitliche Beschränkung aus, um Dienste zeitgesteuert freizuschalten. Dies zeigt, dass die Wahl der Bedingungen sinnvoll war und dass mit dem Hinzufügen der zeitlichen Komponenten in einer künftigen Version der aktuelle Bedarf vollständig gedeckt werden kann.

Das Juniorstudium führt jedes Semester eine Evaluation mittels eines Fragebogens durch. Für den Zeitraum, in dem CASA hier verwendet wurde, um die Videos und andere Werkzeuge einzubinden und darzustellen, wurde der Fragebogen um CASA-spezifische Fragen erweitert. Die zusätzlichen Fragen waren mit einer sechsstufigen Liekert-Skala (1=trifft völlig zu - 6=trifft gar nicht zu) versehen. Insgesamt haben 24 Teilnehmer an dieser Evaluation teilgenommen. Auf die Frage, ob sie die Nutzung der Videos und Werkzeuge als einfach empfanden, stimmten 83% zu (1=29%, 2=46%, 3=%8, 4=%8, 5=%8, 6=0%), was die Tendenz des SUS stützt, und für eine einfache Nutzung in einer akzeptablen Umsetzung spricht. Befragt, ob außer den Vorlesungsaufzeichnungen noch weitere Werkzeuge und Dienste angeboten wurden, antworteten 68% negativ (1=4%, 2=8%, 3=21%, 4=17%, 5=13%, 6=38%). Es zeigt sich, dass es zwar in einigen Fällen genutzt wurde, in der überwiegenden Anzahl an Fällen, wurden durch den Tutor jedoch keine oder kaum weiterführenden Dienste und Werkzeuge eingebunden. Dies kann an den durch CASA begrenzten Möglichkeiten liegen, jedoch auch am Mangel an geeignetem Diensten zu den Lerninhalten der Vorlesung oder am Engagement der Tutoren. Ein ähnliches Bild zeigt sich bei der Frage, ob sie die Medien und Werkzeuge gern genutzt haben. Hier

antworteten 64% negativ (1=4%, 2=8%, 3=21%, 4=17%, 5=13%, 6=38%), wobei hier auch der eventuell als trocken empfundene Lerninhalt einen Einfluss gehabt haben kann. Befragt nach ihren Wünschen für weitere Dienste antwortete die Hälfte der Befragten mit eigenen Vorschlägen, 11 gaben an keine weiteren Wünsche zu haben und ein Teilnehmer gab an, dass er die Frage nicht verstanden habe. Die geäußerten Vorschläge reichten dabei von erweiterten Kommunikationsmöglichkeiten, über das Einbinden von Literatur, der Möglichkeit Vorbereitungstests abzulegen, bis zu erweiterten Interaktionsmöglichkeiten. Dies zeigt, dass CASA noch Potential hat mehr auf die Bedürfnisse der Nutzer einzugehen, indem einerseits die Integration in das Stud.IP vertieft wird, und andererseits die Menge der möglichen Dienste um die Integration von Skype oder anderen Kommunikationssystemen erweitert wird.

Als Ergebnis dieser Fallstudie kann gesagt werden, dass die nutzergetriebene Integration mit CASA einen existierenden Bedarf erfüllt hat und die kontextuelle Verknüpfung der Dienste mit Situationen der Nutzer über alle vier Kontextdimensionen sinnvoll und erwünscht ist. Die Administratoren bezeichneten die wartungsarme Einbettung in die Anwendung selbst als besonders positiv und haben ihr Interesse an Weiterentwicklungen für die kommenden Stud.IP-Versionen bekundet. Gleichzeitig äußerten sich die Nutzer positiv zur Integration der Inhalte in das Stud.IP und gaben wertvolle Hinweise für künftige Entwicklungen.

Das CASA-Plugin für Stud.IP ist als quelloffen unter der Apache 2.0 Lizenz auf GitHub verfügbar. Damit kann es auch nach Abschluss dieser Arbeit produktiv in den Stud.IPs der Universität Rostock und dem Stud.IP des Juniorstudiums in Rostock eingesetzt werden.

5.2.2 LASA - Dissemination von Diensten durch den Nutzer

Die Dissemination von Diensten, also die Verteilung und Verbreitung von Diensten durch den Nutzer, ist ein weiterer Aspekt, der durch diese Arbeit realisiert werden sollte. Als Umsetzung des Konzeptes, dass voneinander unabhängige Nutzer sich gegenseitig Dienstempfehlungen geben, wurde die nächste Fallstudie realisiert. Dabei wird neben der Bereitschaft der Nutzer zur Dienstempfehlung und zur entsprechenden Dienstsuche in öffentlichen Quellen auch evaluiert, wie der Aspekt der Ortsverknüpfung nutzerfreundlich und komfortabel umgesetzt werden kann.

Dazu wurde ein Konzept entworfen, das die Verteilung der Dienstempfehlung über öffentliche und nutzergetriebene Plattformen realisiert. In der Implementierung, die bereits in Abschnitt 4.4.2 beschrieben wurde, entstand ein System, das OpenStreetMaps mit einem MediaWiki verknüpft. Neben diesem Nachweis der Machbarkeit besteht der Nutzen dieser Umsetzung in der anschließend durchgeführten Nutzerstudie. Dabei wurde das System 14 Teilnehmern vorgestellt. Anschließend wurden diese vor eine Reihe von Aufgaben gestellt, bei denen sie das System verwenden sollten.

Im Nutzertest dauerte die Suche nach einem bestimmten Dienst an einem Ort mit durchschnittlich 162 Sekunden signifikant länger als die durchschnittlich 89 Sekunden mit Google. Befragt nach der empfundenen Geschwindigkeit bewerteten 9 der 14 Teilnehmer diese als gleich gut oder besser im Vergleich zu Google. Der Aspekt der Nutzerfreundlichkeit wurde hier vom Großteil der Nutzer im Punkt der Reaktionsgeschwindigkeit also als positiv bewertet. Dies lässt sich damit erklären, dass es nicht um eine einfache Suchanfrage ging, die

Google innerhalb von Sekundenbruchteilen beantworten kann, sondern um die Suche nach der Menükarte eines kleinen Restaurants. Es ist zu vermuten, dass den Nutzern die Zeit für diese Aufgabe als angemessen erschien.

Im Abschluss der Befragung antworteten 10 von 14 Nutzern, dass sie sich vorstellen können, dieses System für die Suche nach ortsbezogenen Diensten zu verwenden. Für CASA konnte in dieser Fallstudie zweierlei gezeigt werden. Einerseits gibt es ein Interesse der Nutzer selbst ortsbezogene Dienste zu nutzen bzw. die selbstgenutzten mit anderen zu teilen, andererseits ist die vorgestellte Nutzungsschnittstelle von Nutzern positiv und als geeignet bewertet worden. Dabei zeigt sich, dass für verschiedene Domänen verschiedene Nutzerschnittstellen als geeignet empfunden werden und daher ein Konzept wie CASA, im Besonderen mit seiner Entkopplung von Schnittstelle und Datenverarbeitung, Nutzungspotential hat.

5.2.3 Anwendungs- und domänenübergreifende Gamification mit CASA - Nutzermotivation in nutzergetriebenen Systemen

Wie bereits in Abschnitt 4.4.1 beschrieben, wurde eine Fallstudie zur Nutzermotivation durchgeführt. Dabei wurde eine Erweiterung für CASA umgesetzt, deren Ziel es einerseits ist, dem Nutzer ein direktes Feedback bei Aktionen im Zusammenhang mit CASA zu geben und ihn andererseits zu motivieren weitere Aktion mit CASA durchzuführen [Wendt 13]. Dabei ist zu beachten, dass dieses Feedback in erster Line dazu dienen soll, den Nutzer zu unterstützen, das System kennenzulernen und auszuprobieren. Als Nebeneffekt sollte so die Menge der produzierten Dienste gesteigert und für alle Nutzer ein Mehrwert erzeugt werden.

Die Umsetzung, die auch für Stud.IP verfügbar ist, erlaubt es dabei konzeptionell verschiedene Anwendungen und Domänen zu verknüpfen. Ein Nutzer kann somit seine Reputation aus verschiedenen Systemen bündeln und in anderen Umgebungen präsentieren. Dieses Konzept ist als solches neu und bietet daher Raum für viele Erweiterungen. Dabei sind Forschungsfragen bei der Authentifizierung gegenüber den Anwendungen und der Authentizität von Auszeichnungen von externen Anwendungen ebenso zu stellen wie die Frage nach einer sinnvollen Standardisierung von virtuellen Auszeichnungen.

Eine belastbare Antwort auf die Frage, ob und wenn ja wie sehr sich durch diese Methode die Interaktionen mit dem CASA-Knoten steigern lassen, fehlt, da es sich hierbei nur um einen Demonstrator gehandelt hat, der im Rahmen einer Bachelorarbeit umgesetzt wurde. Jedoch unterstützen die nahtlose Integration in das System und die Ergebnisse anderer Studien zu Gamification[Hamari 14] die These, dass dieser Ansatz auch in diesem Umfeld geeignet ist, um die Motivation der Teilnehmer zu steigern.

5.3 Zusammenfassung der Evaluation und Ausblick

In diesem Kapitel erfolgte die Evaluation des CASA-Konzeptes und der Umsetzung auf Basis der Anforderungen und Nutzerbefragungen. Vorangestellt wurde erläutert, weshalb die Evaluation am Beispiel eines Gruppenknotens am Besten geeignet ist, um die Anforderungen an

das System zu überprüfen. Dabei liegen die Vorteile in der aktiven Nutzung durch Gruppen an der Hochschule ebenso wie in den Synergieeffekten, die sich durch die Einbettung in ein Campusmanagementsystem wie Stud.IP ergeben. So konnten auf realistische Anforderungen im Produktivbetrieb reagiert und gleichzeitig aussagekräftige Daten zur Nutzung erhoben werden. Die Gegenargumente zu dieser Art der Umsetzung betrafen in erster Linie die Einschränkung des Anwendungsfalls, da durch das Juniorstudium einerseits die zu verknüpfenden Inhalte fest vorgegeben waren und andererseits nur begrenzt eine Dynamik entstand, da es eine strikte Trennung zwischen Dienstautoren und Dienstnutzern gab.

Im Weiteren wurde bezugnehmend auf die konzeptionellen Anforderungen festgestellt, dass diese erfüllt wurden. Besonders das verteilte Kontextmodell und die Privacy-Eigenschaften stellen einen konzeptionellen Mehrwert gegenüber existierenden Konzepten für kontextbewusste Systeme dar. Die verfolgte Strategie mit dem offenen Austausch der Kontextmodelle und der Schnittstellen zu Sensoren mit dem ausgewählten Apache 2.0-Lizenzmodell dient darüber hinaus der Unterstützung der Weiterentwicklung von CASA. Diese ist notwendig, denn wie sich allein aus den Vorschlägen der Nutzer für weitere Dienste ergibt, ist eine geschlossene Entwicklung für die Modelle nicht sinnvoll.

Die Evaluation der realisierten Umsetzung zeigte anschließend, dass die zentrale Vereinheitlichung auf Ebene der Dienste eine große Herausforderung darstellt. So ist zwar mit der Kappselung in Webseiten eine erhebliche Anzahl an möglichen Diensten realisierbar, jedoch zeigen die von Nutzern geäußerten Dienstwünsche, dass hier die Kappselung nur einen ersten Schritt darstellen kann und dass die Integration nativer Anwendungen sowohl von mobilen als auch von stationären Computern eine von Nutzern gewünschte Zielvorgabe darstellt. Mit der Umsetzung als verteiltes System unter Verwendung modularer, unabhängiger Bestandteile konnte sowohl die Kopplung zwischen der Produktiv- und der Experimentalversion effizient gestaltet als auch die Ausfallsicherheit erhöht werden. Außerdem ließ sich so die Heterogenität bei den Lizenzen der verwendeten Bibliotheken von Drittanbietern handhaben. So können in den verschiedenen Modulen auch Bibliotheken mit starkem Copyleft zum Einsatz kommen, ohne dass dies einen Einfluss auf das Gesamtsystem hat, da die Module alle separat für sich ausführbar sind und keine direkten Abhängigkeiten existieren.

Die Evaluation der sozialen und organisatorischen Anforderungen ergab, dass zwar die Datenflusskontrolle durch den Nutzer eine sinnvolle Eigenschaft ist, dies jedoch kein umfassender Schutz für die Nutzerdaten ist. Es wurde explizit darauf hingewiesen, dass die Nutzer mit der Wahl der gruppenöffentlichen Knoten selbst verantwortlich für ihre Daten sind. Es wurde ebenfalls klargestellt, dass weitergehende Sicherheitskonzepte für CASA in Form von verschlüsselten Verbindungen und die Zertifizierung von Knoten ein essentieller Bestandteil der weiteren Entwicklungen sein müssten. Gleichzeitig wurde mit der Gamification-Erweiterung ein prototypischer Baustein erstellt, der zeigt, dass auch die wichtige Nutzermotivation viele Ansatzpunkte für zukünftige Entwicklungen hat.

Anschließend wurde in Form von Fallstudien das CASA-System evaluiert, um auch durch Nutzerbefragungen die Eignung von Konzept und Realisierung in Bezug auf die Integration und Dissemination von Diensten zu bewerten. In der Umsetzung für das Stud.IP erfolgte eine intensive Nutzung durch die Studenten und Tutoren des Juniorstudiums. So konnten im Evaluationszeitraum von 98 Tagen insgesamt 1369 Dienstaufrufe registriert werden. Unter Anwendung eines standardisierten Software-Usability-Score-Fragebogens wurde versucht die

Nutzerfreundlichkeit des entwickelten Plugins zu ermitteln, um sicherzustellen, dass die Aussagen zum System nicht durch eine prototypische Umsetzung verzerrt werden. Leider war die Zahl der Dozenten, die in diesem Zeitraum das Plugin nutzten, gering, und auch von diesen beantwortete nicht jeder den Fragebogen. Der ermittelte Wert aus den 4 Befragungen lag im Median bei 63,75 und platziert das getestete CASA-Plugin für Stud.IP in einem akzeptablen Bereich für eine prototypische Entwicklung. Die Befragung der Tutoren ergab, dass die verwendeten Kontextarten zur Verknüpfung von Diensten geeignet waren, zusätzlich wurde die Frage nach einer Erweiterung für eine zeitliche Verknüpfung von Diensten gestellt. Diese lässt sich integrieren und sollte den Bedarf nach einer Integration abschließen. Die Befragung der Teilnehmer des Juniorstudiums stützte die positive Tendenz aus dem SUS, da hier 20 der 24 Teilnehmer angaben, dass sie der Aussage, die Nutzung sei einfach, zustimmten. Außerdem zeigte sich, dass die Juniorstudenten noch eine Vielzahl an weiteren Ideen für Dienste hatten, die eventuell in CASA integriert werden könnten. Das Ausprobieren der Tutoren führte ebenfalls zu interessanten Ergebnissen und weiterführenden Ideen. So wurde durch einen Tutor ein Google-Docs-Dokument eingebunden, um mit den Juniorstudenten eine Evaluation der Veranstaltung durchzuführen. Diese Möglichkeit war vorher nicht bekannt und wurde auch von den Administratoren des Juniorstudiums anschließend intensiv genutzt.

Insgesamt hat sich gezeigt, dass die Nutzung und die positive Rückmeldung durch die Nutzer für das Konzept der nutzergetriebenen Integration und seine Umsetzung mit CASA sprechen. Es muss jedoch auch deutlich gesagt werden, dass die Anzahl der Rückmeldungen nicht statistisch relevant ist und daher nur als Tendenz wahrgenommen werden kann.

Abschließend wurden im Rahmen der Fallstudien zur Dissemination von Diensten mit LASA und der Gamification-Erweiterung weitere Aspekte diskutiert. So ergab die Befragung der Nutzer zum Konzept der nutzergetriebenen Dissemination von Diensten, dass die überwiegende Zahl der Nutzer sich vorstellen könnte, ein solches System zu nutzen. Gleichzeitig zeigte sich, dass auch die Reaktionsgeschwindigkeit als weniger relevant empfunden wurde, denn trotz einer anderthalb mal so langen Suchdauer im Vergleich zu Google bewertete die Mehrzahl von Benutzern dies dennoch als akzeptabel. Für den Aspekt der Gamification fehlen leider aussagekräftige Nutzermeinungen, da diese nur als Demonstrator umgesetzt und nicht in das Produktivsystem des Stud.IP überführt wurde. Jedoch lassen Ergebnisse zu Studien hier vermuten, dass eine Steigerung der Motivation durch Gamification zu erwarten ist.

Im Ausblick lässt sich feststellen, dass es noch viel Raum für Erweiterungen in den CASA-Umsetzungen gibt. Diese sollten im Besonderen den Bereich des Schutzes der Kommunikation der Nutzer mit dem System betreffen, jedoch auch die Nutzerfreundlichkeit der graphischen Schnittstellen verbessern. Die Integration weiterer graphischer Schnittstellen sollte hier auch Weiterentwicklungen im Bereich der Dienstintegration fördern und so auch die Einbindung von nativen Applikationen ermöglichen.

Kapitel 6

Zusammenfassung und Ausblick

Die allgegenwärtige Nutzung von Software-Diensten und -Anwendungen ist bereits Bestandteil des Alltags an Hochschulen. Dem entgegen stehen viele, durch komplexe Altanwendungen realisierte Arbeitsabläufe, deren Modernisierung oder Austausch nicht ohne Weiteres möglich ist.

Die situationsabhängige Integration aktueller Dienste in die existierenden Arbeitsabläufe und Prozesse von Lehre, Forschung und Verwaltung an Hochschulen ist dabei ein sinnvoller, nächster Schritt in der Entwicklung hin zu einer effizienten und modernen Hochschule mit Service-Orientierung. Dabei bietet die Beteiligung der Nutzer an der Identifikation relevanter Dienste sowie an der Zuordnung dieser zu relevanten Situationen und Prozessen eine Lösung für mehrere Herausforderungen. Diese Methode adressiert sowohl die Heterogenität der Dienste und Nutzer, als auch die Dynamik der Prozesse und Situationen. Gleichzeitig wird durch die Integration in existierende Arbeitsumgebungen und -prozesse der Wartungs- und Eingewöhnungsaufwand verringert, was die Akzeptanz sowohl auf Nutzer- als auch auf Administrationsebene fördert.

6.1 Zusammenfassung und Ergebnisse

Das Graduiertenkolleg MuSAMA befasst sich allgemein mit der Interaktion und Kommunikation in intelligenten Umgebungen. Diese Arbeit adressiert dabei im Speziellen die Frage, wie in einer physisch verteilten und komplexen Umgebung mit heterogenen Nutzergruppen und ihren unterschiedlichen Endgeräten eine Empfehlung und Integration von unterstützenden Diensten passend zur jeweiligen Nutzersituation erfolgen kann. Als Rahmenszenario wurde mit der pervasiven Hochschule ein universitäres Umfeld gewählt, das heute bereits in Ansätzen realisiert ist. Dabei wurde die Heterogenität als zentrale Herausforderung für diese großflächige, pervasive Umgebungen identifiziert, da sie nicht nur auf technischer Ebene, sondern auch auf organisatorischer und sozialer Ebene vorliegt.

Ausgehend von einer Betrachtung der Rahmenbedingungen, welche sich durch die Erhebung und Verwendung von Kontextinformationen ergeben, wurde das universitäre Umfeld darauf untersucht, welche Kontextquellen und -informationen sich hier eignen, um der Heterogenität zu begegnen. Dabei wurde festgestellt, dass einerseits die bereits in [Dey 00] benannten Kontextdimensionen von Identität, Ort, Zeit und Aktivität relevant sind, dass jedoch andererseits die Beschreibung der Identität nicht unbedingt eine Zuordnung zu einer konkreten Person beinhalten muss. Basierend darauf wurde eine Klassifikation nach Domänen und der Reichweite der Dienste mit den Eigenschaften privat, gruppenöffentlich und öffentlich erstellt.

In einem nächsten Schritt wurde nach einer Methode gesucht, die geeignet ist effizient die Beschreibungen von Diensten und Situationen sowie die Zuordnung zwischen diesen zu erstellen und zu pflegen. Gleichzeitig sollte diese Methode auch die Weiterentwicklung des Systems auf technischer Ebene ermöglichen, da durch die Vielzahl der möglichen Domänen auch davon ausgegangen werden musste, dass Anpassungen in den technischen Modulen zur Kontexterhebung und -verarbeitung regelmäßig notwendig werden. Da eine vollständig zentrale Administration und Redaktion des Systems auf Grund der Komplexität sehr aufwändig wäre, wurde mit Crowdsourcing eine Methode untersucht, die auf eine Verteilung der Aufgaben an interessierte Nutzer abzielt. Mittels einer Betrachtung der Evolutionstheorie wurden Anforderungen an die Art des Crowdsourcings definiert, die geeignet sind einerseits eine Anpassung des Systems innerhalb einer Domäne zu gewährleisten und die andererseits erlauben, Innovationen auch auf andere Domänen zu übertragen.

Da somit eine geeignete Methodik zur Erstellung und Organisation von Dienstempfehlungen identifiziert war, wurde im Weiteren die technische Perspektive betrachtet. Es wurde festgestellt, dass neben der Unterstützung heterogener Endgeräte, Anwendungen und Dienste auch die Eignung zur Integration in Altsysteme eine Anforderung darstellt, da diese in besonderem Maße an Hochschulen genutzt werden und Teil vieler Arbeitsabläufe sind. In einer Analyse existierender Systeme deren Fokus auf die bereits identifizierten Anforderungen und ihrer Eignung zur nutzergetriebenen Dienstvermittlung in Umgebungen mit heterogenen Systemen und Nutzergruppen lag, wurden besonders im Bereich Nutzereinbindung, Erweiterbarkeit und Datenschutz noch Fehlstellen identifiziert. Diese wurden sowohl im Konzept als auch in der Umsetzung dieser Arbeit adressiert.

Darauf aufbauend wurde mit CASA ein Konzept für ein kontextgestütztes Dienstempfehlungssystem entwickelt, das modular und durch den Nutzer erweiterbar ist. Das hier vorgestellte Konzept zeichnet sich im Besonderen durch eine Separation von privaten, öffentlichen und gruppenöffentlichen Daten und Diensten aus. Gleichzeitig wurde das System so angelegt, dass die Integration in Altanwendungen durch die Architektur und Schnittstellen effizient möglich ist und im Rahmen von Plugins realisiert werden kann [Lehsten 11b]. Dies stellt zusammen mit der modularen Struktur und der Nutzerorientierung einen neuen Ansatz bei Assistenzsystemen dar, der sich durch die Veröffentlichung der Quellen unter der freizügigen Apache 2.0 Lizenz für die Adaption durch andere Nutzer und Organisationen anbietet.

Die Umsetzbarkeit des CASA-Konzeptes wurde durch eine exemplarische Realisierung bestätigt und auch die Anwendbarkeit auf die universitäre Umgebung gezeigt. Dazu wurde sowohl für das Rechenzentrum als auch für das Juniorstudium der Universität Rostock eine Umsetzung implementiert, die direkt im Produktivsystem eingesetzt wurde. Die entwickelte Lösung wurde dabei auch über die Dauer dieser Arbeit hinaus Teil des Systems und möglicher An-

knüpfungspunkt für weitere Forschungsarbeiten. Ferner wurde durch weitere prototypische Implementierungen gezeigt, wie eine Integration von Ortsdaten durch Nutzer effizient erfolgen kann und wie die Motivation, sich an diesem System redaktionell zu beteiligen, gesteigert werden kann. Gleichzeitig wurde mit der konzeptionellen Integration von Gamification in CASA ein Ausblick darauf gegeben, dass CASA sich auch als Basis für anwendungsübergreifende Systeme zur Nutzerassistenz eignet.

In der Evaluation wurde durch den Vergleich mit den Anforderungen gezeigt, dass das CASA-System in seiner ersten Umsetzung geeignet ist als Assistenzsystem für Dienstempfehlungen im Campus-Umfeld eingesetzt zu werden. Für weitergehende Anwendungen entsprechend des Konzeptes wurde der Aspekt der Sicherheit in der Kommunikation benannt, der hier noch erweitert werden muss. Dies ist durch die Veröffentlichung des CASA-Systems unter der Apache 2.0 Lizenz möglich. Abschließende Nutzerbefragungen ergaben, dass die Problematik, die durch CASA adressiert wird, vorhanden ist, und dass es einen Bedarf nach dem vorgestellten System gibt.

Die Ergebnisse dieser Arbeit lassen sich daher wie folgt zusammenfassen:

Ergebnis 1: Analyse der Anforderungen für ein Dienstempfehlungssystem im universitären Umfeld

In dieser Arbeit wurde eine Analyse von nutzbaren Kontextrelationen durchgeführt, die zur Empfehlung von Diensten im universitären Umfeld verwendet werden können. Der darauf aufbauende Anforderungskatalog umfasst sowohl konzeptionelle, technische als auch soziale Aspekte. So sind neben der Modularität und Erweiterbarkeit auch die Eignung zur Integration in existierende Systeme und Strukturen wichtige Elemente, die jedoch von einer Einbindung des Nutzers in den Redaktions- und Weiterentwicklungsprozess begleitet werden sollten. Unter Berücksichtigung dieser Aspekte ist die Konzeption und Umsetzung von vielseitigen Systemen möglich, deren Nutzer sich aktiv an der Weiterentwicklung beteiligen. Indem hier ein Privacy-by-Design-Ansatz verfolgt wird, kann ein Schutz der persönlichen Daten auf verschiedenen Ebenen gewährleistet werden.

Ergebnis 2: Konzeption und Implementierung eines verteilten, nutzergetriebenen Dienstempfehlungssystems

Mit dem in dieser Arbeit entwickelten CASA-System wurde ein innovatives Konzept vorgestellt, das durch seine verteilte Organisation mit domänenspezifischen Knoten eine Alternative zu existierenden Ansätzen mit zentraler Verwaltung darstellt. Die Verteilung erlaubt eine Aggregation von Dienstempfehlungen über verschiedene Anwendungen und bietet gleichzeitig eine plattformunabhängige Integration in Altanwendungen. Des Weiteren zeigt die mehrschichtige, am Konzept des Crowdsourcings orientierte Integration der Nutzer von

der Redaktion der Dienstbeschreibungen und Situationen über die Weiterentwicklung einzelner Module bis zum Betrieb eigener Knoten einen neuen Weg auf, wie mit der Heterogenität und Dynamik in dieser Umgebung umgegangen werden kann [Lehsten 14].

Ergebnis 3: Demonstration der Umsetzbarkeit des CASA-Systems und dessen Eignung zur Integration in Produktivumgebungen

Im Rahmen der Arbeit erfolgte die Umsetzung eines prototypischen Demonstrators, der als regelbasiertes System auf Basis der durch die Nutzer zur Verfügung gestellten Daten deren Situationen abbildet. In einer weiteren, angepassten Implementierung wurde gezeigt, wie sich das vorgestellte System in existierende Umgebungen des universitären Lernmanagementsystems Stud.IP und in das Stud.IP des Juniorstudiums einbetten lässt. In diesen Produktivsystemen wurden durch die wartungsarme Integration von Zusatzdiensten und die nutzerabhängige Verteilung an die Anwender bleibende Mehrwerte erzielt [Lehsten 13]. Insgesamt ergab sich eine sinnvolle Integration der neuen Funktionalität in die Arbeitsabläufe der Nutzer. Die aufgezeigten Erweiterungen des Kontextmodells illustrieren die Anpassbarkeit und mit den vorgestellten Fallstudien zu den Bereichen Nutzermotivation und ortsbasierte Dienstverknüpfung wurde auch das weitere Potential aufgezeigt.

6.2 Ausblick und weiterführende Forschungsfragen

In dieser Dissertation wurde ein Weg aufgezeigt, wie sich die Heterogenität bei der Dienstvermittlung und -empfehlung nachhaltig adressieren und gleichzeitig die Privatsphäre der Nutzer schützen lassen. Dabei wurden jedoch auch konzeptionelle Fragestellungen und technische Aspekte offen gelassen. Daher sollen hier die wichtigsten benannt werden und Vorschläge für weitere Schritte in der Forschung und Entwicklungen gegeben werden.

In der Arbeit wurde sich auf jene Dienste beschränkt, die sich in Webseiten kapseln lassen. Dies schließt sowohl native Anwendungen als auch andere Dienste aus, wie den Zugang zu Netzwerken oder Geräten. Daher sollte in künftigen Arbeiten überprüft werden, wie sich Anwendungen auf mobilen und stationären Endgeräten so in die Systemstruktur integrieren lassen, dass sie auch das Ausführen von anderen Anwendungen oder Aktionen in diesen Anwendungen als Dienst nach außen anbieten können. Gleichzeitig sollte auch der Aspekt der zentralen Vereinheitlichung weiter untersucht werden. Dabei könnte durch die Entwicklung von zur Laufzeit einbindbaren Bibliotheken (z.B. über OSGi) auch die Verwendung von zuvor unbekannten Geräten gestattet werden.

Der Aspekt der Sicherheit wird von dieser Arbeit nicht fokussiert, weshalb gerade hier noch einige Problemfelder offen sind. So sind Konzepte zur Authentifizierung von Knoten notwendig, um sicherzustellen, dass sich durch manipulierte CASA-Knoten nicht zum Beispiel die Zugangsdaten zu sozialen Netzwerken auslesen lassen. Ein praktikabler Weg könnte hier die

Nutzung von dem in RFC 6749 spezifizierten OAuth-Framework[1] sein. Dies ersetzt jedoch nicht die ebenfalls notwendige, verschlüsselte Kommunikation. Für eine tiefergehende Integration von CASA in mobile Anwendungen sind daher in erster Linie noch weitere Arbeiten auf technischer Seite notwendig.

Die Fallstudie zur Dissemination der Dienste über die Nutzer hat gezeigt, dass ein grundlegendes Interesse an diesem Bereich besteht. Daher sollte auch der Aspekt der Integration von externen Informationsquellen, wie Kartendiensten und Online-Enzyklopädien, weiter untersucht werden. Mit Einbindung dieser Datenquellen und ihrer strukturierten Daten könnten Nutzer auch Anfragen erstellen, die dynamisch neue Dienste einbinden wie zum Beispiel die Integration von als lesenswert markierten Wikipedia-Artikel für Sehenswürdigkeiten in der näheren Umgebung des Nutzers. Dabei können Systeme wie DBpedia genutzt werden, in denen bereits über vier Millionen Objekte mit über 500 Millionen Relationen verknüpft wurden[2]. Eine effiziente Methode zur Auswertung und Strukturierung dieser Daten ist dabei unerlässlich, da ein Abrufen aller verfügbaren Informationen zu jedem Zeitpunkt keine Option darstellt.

Neben der Erschließung derart strukturierter, nicht nutzerbezogener Kontextdaten wäre auch die Erschließung von nicht nutzergenerierten Diensten eine mögliche Erweiterung. Dazu könnte eine anonyme Erhebung von Nutzerdaten ebenso genutzt werden wie auch eine tiefergehende Klassifikation der verwendeten Dienste. Dabei ist jedoch darauf zu achten, dass die Privatsphäre des Nutzers nicht kompromittiert wird und die Dienste keinerlei Rückschlüsse auf die Identität zulassen. Mit der Erschließung derart großer Kontext- und Dienstmengen stellt sich auch die Frage nach einer effizienteren Gewichtung bei großen Resultatmengen. Dazu könnten weitere Metadaten Anwendung finden, wie z.B. die Nutzung von Abrufstatistiken.

Neben diesen technischen Weiterentwicklungen stellt eine effiziente Anwendung von CASA auch organisatorische Anforderungen an die Infrastrukturen. So ist eine Ursache für starke Heterogenitäten häufig die Existenz einer Vielzahl an verschiedenen Hierarchien für die technischen Systeme innerhalb einer Domäne, wie zum Beispiel innerhalb einer Universität. Die Knoten-Charakteristik erlaubt zwar ein weitgehendes Nebeneinander verschiedener CASA-Systeme, jedoch wäre hier eine Verbindung über verschiedene Systeme sinnvoll, um Synergien, wie in der anwendungsübergreifenden Gamification-Fallstudie, zu nutzen. Gleichzeitig stellen die proprietären und nicht erweiterbaren Systeme in den Universitäten eine weitere Herausforderung dar. Hier ist mit der Service-orientieren Struktur des Hochschulverwaltungssystems HISinOne bereits ein System vorhanden, dessen Struktur eine Kopplung und Integration über die verschiedenen Säulen einer Hochschule hinweg erlauben könnte. Gerade für die effiziente Nutzung von CASA sind ebensolche Systeme ideal, die mit offenen Schnittstellen ihre eigene Funktionalität in Dienste kapseln.

[1] https://tools.ietf.org/html/rfc6749 [Online letzter Zugriff 15.02.2018]

[2] http://wiki.dbpedia.org/about - Die Zahlen beziehen sich auf den Stand von 2014 [Online letzter Zugriff 15.02.2018]

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Adomavicius 05] Gediminas Adomavicius, Alexander Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6):734–749, jun 2005.

[Baldauf 07] Matthias Baldauf, Schahram Dustdar, Florian Rosenberg. A survey on context-aware systems. Int. J. Ad Hoc and Ubiquitous Computing, 2(4):263–277, 2007.

[Bangor 08] Aaron Bangor, Philip T. Kortum, James T. Miller. An Empirical Evaluation of the System Usability Scale. International Journal of Human-Computer Interaction, 24(6):574–594, jul 2008.

[Bass 03] Len Bass, Paul Clements, Rick Kazman. Software Architecture in Practice, Band 2nd of SEI series in software engineering. Addison-Wesley Professional, 2003.

[Bellavista 13] P. Bellavista, A. Corradi, M. Fanelli, L. Foschini. A Survey of Context Data Distribution for Mobile Ubiquitous Systems. ACM Computing Surveys, 45(1), 2013.

[Berwanger 18] Jörg Berwanger, Marion Steven, Werner Krommes, Eggert Winter. Gabler Wirtschaftslexikon, Stichwort: Prozess. http://wirtschaftslexikon.gabler.de/Archiv/12416/prozess-v13.html. zuletzt abgerufen am 09. Februar 2018.

[Bettini 10] Claudio Bettini, Oliver Brdiczka, Karen Henricksen, Jadwiga Indulska, Daniela Nicklas, Anand Ranganathan, Daniele Riboni. A survey of context modelling and reasoning techniques. Pervasive and Mobile Computing, 6(2):161–180, apr 2010.

[Brooke 96] J Brooke. SUS - A quick and dirty usability scale. In P. Jordan, B. Thomas, B Weerdmeester, Hrsg., Usability evaluation in industry, Seiten 189–194. Taylor & Francis, London (UK), 1996.

[Cantino 13] Andrew Cantino. Huginn. https://github.com/cantino/huginn. zuletzt abgerufen am 23. August 2017.

[Chen 00] Guanling Chen, David Kotz. A Survey of Context-Aware Mobile Computing Research. Technischer Bericht TR2000-381, Dartmouth College, Computer Science, 2000.

[Chen 04a] H. Chen, F. Perich, T. Finin, A. Joshi. SOUPA: standard ontology for ubiquitous and pervasive applications. In Imrich Chlamtac, Fausto Giunchiglia, Hrsg., Tagungsband:

The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004., Seiten 258–267. IEEE, 2004.

[Chen 04b] Harry Chen, Tim Finin, Anupam Joshi. A context broker for building smart meeting rooms. In Craig Schlenoff, Michael Uschold, Hrsg., Tagungsband: Proceedings of the AAAI Symposium on Knowledge Representation and Ontology for Autonomous Systems Symposium, 2004 AAAI Spring Symposium, Seiten 53–60. AAAI Press, Menlo Park, CA, 2004.

[Chen 76] Peter Pin-Shan Chen. The entity-relationship model-toward a unified view of data. ACM Transactions on Database Systems (TODS), 1(1):9–36, 1976.

[Cheshire 13] Stuart Cheshire, Marc Krochmal. Multicast DNS. RFC 6762, Internet Engineering Task Force, Februar 2013.

[Compton 09] Michael Compton, Cory Andrew Henson, Laurent Lefort, Holger Neuhaus, Amit P. Sheth. A Survey of the Semantic Specification of Sensors. In Kerry Taylor, Arun Ayyagari, David De Roure, Hrsg., Tagungsband: 2nd International Semantic Sensor Networks Workshop, Seiten 17–32, Wahington, DC, 2009. CEUR-WS.org.

[Cook 07] Diane J. Cook, Sajal K. Das. How smart are our environments? An updated look at the state of the art. Pervasive and Mobile Computing, 3(2):53–73, mar 2007.

[Cook 12] Diane J. Cook, Sajal K. Das. Pervasive computing at scale: Transforming the state of the art. Pervasive and Mobile Computing, 8(1):22–35, feb 2012.

[Crossland 08] Dave Crossland, Francis Irving, Thorsten Glaser, Rufus Pollock, Evan Prodromu, Kragen Sitaker, Luis Villa. Open Software Service Defenition. http://opendefinition.org/ossd/. zuletzt abgerufen am 23. August 2017.

[Darwin 58] Charles Darwin, Alfred Wallace. On the tendency of species to form varieties; and on the perpetuation of varieties and species by natural means of selection. Journal of the Proceedings of The Linnean Society, 3, 1858.

[Darwin 59] Charles Darwin. On the Origin of the Species by Means of Natural Selection: Or, The Preservation of Favoured Races in the Struggle for Life. John Murray, 1859.

[Denham 13] Alex Denham. Huginn - An Information Collection Agent. `http://www.i-programmer.info/news/136-open-source/5684-huginn-an-information-collection-agent.html`. zuletzt abgerufen am 23. August 2017.

[Dey 00] Anind K. Dey, Gregory D. Abowd. Towards a better understanding of context and context-awareness. Tagungsband: CHI 2000 workshop on the what, who, where, when, and how of context-awareness, Band 4, Seiten 1–6, 2000.

[Dressler 10] Enrico Dressler. Allgegenwärtige Kommunikation heterogener Systeme in Smart Ensembles. Dissertation, Universität Rostock, 2010.

[Erl 08] Thomas Erl. SOA - Studentenausgabe: Entwurfsprinzipien für serviceorientierte Architektur. Programmer's Choice. Addison Wesley in Pearson Education Deutschland, 2008.

[Flavián 06] Carlos Flavián, Miguel Guinalíu, Raquel Gurrea. The role played by perceived usability, satisfaction and consumer trust on website loyalty. Information & Management, 43(1):1–14, jan 2006.

[Fowler 10] Martin Fowler. Domain Specific Languages. Addison-Wesley Professional, 1. Auflage, 2010.

[Free Software Foundation 17] Free Software Foundation. What is free software? - The Free Software Definition. `https://www.gnu.org/philosophy/free-sw.en.html`. Versionsstand 1.153 vom 4. April 2017, zuletzt abgerufen am 23. August 2017.

[Gab 13] Gabler Kompakt-Lexikon Wirtschaft. Springer Fachmedien Wiesbaden, Wiesbaden, 2013.

[Gatti 15] Francesco Gatti. Yahoo Shuts Down Pipes, The First Service To Make APIs For Everyone. http://readwrite.com/2015/06/08/yahoo-shuts-down-pipes. zuletzt abgerufen am 23. August 2017.

[Hamari 14] Juho Hamari, Jonna Koivisto, Harri Sarsa. Does Gamification Work? – A Literature Review of Empirical Studies on Gamification. In Ralph H. Jr. Sprague, Hrsg., Tagungsband: 47th Hawaii International Conference on System Sciences, Seiten 3025–3034. IEEE, jan 2014.

[Henricksen 03] K Henricksen, J Indulska, A Rakotonirainy. Generating Context Management Infrastructure from High-Level Context Models. In M.-S. Chen, P.K. Chrysanthis, M. Sloman, A. Zaslavsky, Hrsg., Tagungsband: Proceedings of the 4th International Conference on Mobile Data Management (MDM) - Industrial Track, 2003.

[Henricksen 06] Karen Henricksen, Ricky Robinson. A survey of middleware for sensor networks: State-of-the-art and future directions. Tagungsband: Proceedings of the International Workshop on Middleware for Sensor Networks, MidSens '06, Seiten 60–65, New York, NY, USA, 2006. ACM.

[Howe 06] Jeff Howe. The rise of crowdsourcing. Wired magazine, 14(6):1–6, 2006.

[Howe 10] Jeff Howe. Crowdsourcing - why the power of the crowd is driving the future of business. http://www.crowdsourcing.com/cs/. zuletzt abgerufen am 23. August 2017.

[IFROSS 17] IFROSS. Institut für Rechtsfragen der Freien und Open Source Software - Lizenz-Center. http://www.ifross.org/lizenz-center. zuletzt abgerufen am 23. August 2017.

[Indulska 03] Jadwiga Indulska, Peter Sutton. Location management in pervasive systems. Tagungsband: Proceedings of the Australasian information security workshop conference on ACSW frontiers, number vi, Seiten 143 – 151, 2003.

[Kaufmann 11] Nicolas Kaufmann, Thimo Schulze, Daniel Veit. More than fun and money. Worker Motivation in Crowdsourcing – A Study on Mechanical Turk". Tagungsband: AMCIS 2011 Proceedings, Band 1 of 5, Seiten 3012–3032. Curran Associates, Inc., 2011.

[Kennedy 63] John F. Kennedy. Remarks Upon Presenting the NASA Distinguished Service Medal to Astronaut L. Gordon Cooper.

[Kiani 11] S.L. Kiani, B. Moltchanov, M. Knappmeyer, N. Baker. Large-scale context-aware system in smart spaces: Issues and challenges. Tagungsband: Internet Communications (BCFIC Riga), 2011 Baltic Congress on Future, Seiten 173–180. IEEE, 2011.

[Kjær 07] Kristian Ellebæk Kjær. A survey of context-aware middleware. In Wilhelm Hasselbring, Hrsg., Tagungsband: SE'07 Proceedings of the 25th conference on IASTED International Multi-Conference: Software Engineering, Seiten 148–155, Anaheim, feb 2007. ACTA Press.

[Klahold 09] André Klahold. Empfehlungssysteme. Vieweg+Teubner, Wiesbaden, 2009.

[Knappmeyer 13] Michael Knappmeyer, Saad Liaquat Kiani, Eike Steffen Reetz, Nigel Baker, Ralf Tonjes. Survey of Context Provisioning Middleware. IEEE Communications Surveys & Tutorials, 15(3):1492–1519, 2013.

[Krumme 17] Jan-Hendrik Krumme. Gabler wirtschaftslexikon, stichwort: Telemedien. http://wirtschaftslexikon.gabler.de/Archiv/17821/telemedien-v12.html. zuletzt abgerufen am 23. August 2017.

[Lackes 17] Richard Lackes, Markus Siepermann. Gabler Wirtschaftslexikon, Stichwort: Benutzer. http://wirtschaftslexikon.gabler.de/Archiv/74931/benutzer-v9.html. zuletzt abgerufen am 23. August 2017.

[Langheinrich 01] Marc Langheinrich. Privacy by design—principles of privacy-aware ubiquitous systems. In Gregory D. Abowd, Barry Brumitt, Steven Shafer, Hrsg., Tagungsband: Ubicomp 2001: Ubiquitous Computing, Seiten 273–291. Springer, Berlin, Heidelberg, 2001.

[Lehsten 08] Philipp Lehsten, Andreas Thiele, René Zilz, Enrico Dressler, Raphael Zender, Ulrike Lucke, Djamshid Tavangarian. Dienste-basierte Kopplung von virtueller und Präsenzlehre. In Silke Seehusen, Ulrike Lucke, Stefan Fischer, Hrsg., Tagungsband: DeLFI, Band 132 of LNI, Seiten 77–88. GI, 2008.

[Lehsten 11a] Philipp Lehsten, Sebastian Bader, Djamshid Tavangarian. CASA - A Context Aware Service Access. 3rd Workshop on Context-Systems Design, Evaluation & Optimisation (CoSDEO 2011), Copenhagen, 2011.

[Lehsten 11b] Philipp Lehsten, Alexander Gladisch, Djamshid Tavangarian. Context-Aware Integration of Smart Environments in Legacy Applications. In David V. Keyson, Mary Lou Maher, Norbert Streitz, Adrian David Cheok, Juan Carlos Augusto, Reiner Wichert, Gwenn Englebienne, Hamid K. Aghajan, Ben J. A. Kröse, Hrsg., Tagungsband: AmI, Band 7040 of Lecture Notes in Computer Science, Seiten 126–135. Springer, 2011.

[Lehsten 13] Philipp Lehsten, Djamshid Tavangarian. CASA - Ein Konzept zur nutzergetriebenen, kontextbasierten Dienstintegration in Lehr- und Lernumgebungen. In Paul Müller, Bernhard Neumair, Helmut Reiser, Gabi Dreo Rodosek, Hrsg., Tagungsband: DFN-Forum Kommunikationstechnologien, Band 217 of LNI, Seiten 1–9. GI, 2013.

[Lehsten 14] Philipp Lehsten, Sebastian Bader, Djamshid Tavangarian. CASA - Context-Aware Service Access. Journal of Integrated Design and Process Science, 18(1):21–38, 2014.

[Manola 04] Frank Manola, Eric Miller. RDF Primer. W3C Recommendation, W3C, Feb. 2004. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

[McGuinness 04] Deborah McGuinness, Frank van Harmelen. OWL web ontology language overview. W3C Recommendation, W3C, Feb. 2004. http://www.w3.org/TR/2004/REC-owl-features-20040210/.

[Melville 02] Prem Melville, Raymond J. Mooney, Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. Tagungsband: in Eighteenth National Conference on Artificial Intelligence, Seiten 187–192, 2002.

[Melzer 07] Ingo Melzer. Service-orientierte Architekturen mit Web Services - Konzepte, Standards, Praxis (2. Aufl.). Spektrum Akademischer Verlag, 2007.

[Negroponte 96] Nicholas Negroponte. Being Digital. Vintage Books, first edit Auflage, 1996.

[OMG 11] Object Management Group OMG. Business Process Model and Notation (BPMN) Version 2.0. Technischer Bericht, jan 2011.

[Open Source Initiative 17] Open Source Initiative. The MIT License. `https://opensource.org/licenses/MIT`. zuletzt abgerufen am 19. November 2017.

[Papazoglou 06] Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, Frank Leymann, Bernd J. Krämer. 05462 service-oriented computing: A research roadmap. In Francisco Cubera, Bernd J. Krämer, Michael P. Papazoglou, Hrsg., Tagungsband: Service Oriented Computing (SOC), number 05462 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2006. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

[Posch 11] Torsten Posch, Klaus Birken, Michael Gerdom. Basiswissen Softwarearchitektur. dpunkt-Verlag, Heidelberg, 3. aktual. Auflage, 2011.

[Qin 08] Zheng Qin, Jian-Kuan Xing, Xiang Zheng. Software Architecture. Advanced Topics in Science and Technology in China. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[Quinn 11] Alexander J Quinn, Benjamin B Bederson. Human computation. Tagungsband: Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11, Seiten 1403–1412, New York, New York, USA, 2011. ACM Press.

[Ridley 03] Marc Ridley. Evolution. Wiley, 3. Auflage, 2003.

[Rosen 04] Lawrence Rosen. Open Source Licensing: Software Freedom and Intellectual Property Law. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004. Online verfügbar unter: https://www.rosenlaw.com/oslbook.htm , zuletzt abgerufen am 19. November 2017.

[Roussaki 06] I. Roussaki, M. Strimpakou, N. Kalatzis, M. Anagnostou, C. Pils. Hybrid context modeling: A location-based scheme using ontologies. Tagungsband: Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'06), Seiten 2–7. IEEE, 2006.

[Salatino 16] Mauricio Salatino, Mariano De Maio and Esteban Aliverti. Mastering JBoss Drools 6. Packt Publishing, 1. Auflage, 2016.

[Schaar 10] Peter Schaar. Privacy by Design. Identity in the Information Society, 3(2):267–274, apr 2010.

[Schilit 94] B Schilit, N Adams, R Want. Context-aware computing applications. In Luis-Felipe Cabrera, Mahadev Satyanarayanan, Hrsg., Tagungsband: Workshop on Mobile Computing Systems and Applications, Band Santa Cruz of Proceedings. Workshop on Mobile Computing Systems and Applications (Cat. No.94TH06734), Seiten 85–90. Columbia Univ, New York, NY, USA, IEEE Comput. Soc. Press, 1994.

[Sjurts 17] Insa Sjurts. Gabler Wirtschaftslexikon, Stichwort: Rundfunkstaatsverträge. http://wirtschaftslexikon.gabler.de/Archiv/569794/rundfunkstaatsvertraege-v2.html. zuletzt abgerufen am 23. August 2017.

[Smith 14] Brett Smith. GNU GPLv3 Eine Kurzanleitung - GNU Projekt - Free Software Foundation. https://www.gnu.org/licenses/quick-guide-gplv3.html. zuletzt abgerufen am 23. August 2017.

[Strang 04] Thomas Strang, Claudia Linnhoff-Popien. A context modeling survey. Tagungsband: First International Workshop on Advanced Context Modelling, Reasoning And Management at UbiComp 2004, Nottingham, England, September 7, 2004, Band Workshop, Seiten 31–41, Nottingham, 2004.

[Surowiecki 05] James Surowiecki. The Wisdom of Crowds. Anchor Books. Anchor, 2005.

[Tan 12] Rong Tan, Junzhong Gu, Zhou Zhong, Peng Chen. SOCOM: Multi-sensor Oriented Context Model Based on Ontologies. Tagungsband: 2012 Eighth International Conference on Intelligent Environments, Seiten 236–242. IEEE, jun 2012.

[Tanenbaum 08] Andrew S. Tanenbaum, Maarten van Steen. Verteilte Systeme : Prinzipien und Paradigmen. Pearson Studium, München, 2. aktual. Auflage, 2008.

[Tavangarian 09a] Djamshid Tavangarian. Pervasive University. it - Information Technology, 51(1):3–5, jan 2009.

[Tavangarian 09b] Djamshid Tavangarian, Ulrike Lucke. Pervasive University - A Technical Perspective Die Pervasive University aus technischer Perspektive. it - Information Technology, 51(1):6–13, jan 2009.

[The Apache Software Foundation 04] The Apache Software Foundation. Apache License, Version 2.0. http://www.apache.org/licenses/LICENSE-2.0. zuletzt abgerufen am 23. August 2017.

[Torvalds 17] Linus Torvalds, Junio C. Hamano, Shawn O. Pearce. Git: Fast version control system. git-scm.com/. zuletzt abgerufen am 23. August 2017.

[Turner 04] David Turner. The LGPL and Java - GNU Project - Free Software Foundation. https://www.gnu.org/licenses/lgpl-java.en.html. zuletzt abgerufen am 23. August 2017.

[von Ahn 08] Luis von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, Manuel Blum. reCAPTCHA: human-based character recognition via Web security measures. Science (New York, N.Y.), 321(5895):1465–8, sep 2008.

[Wang 04] Xiao Hang Wang, Tao Gu, Da Qing Zhang, Hung Keng Pung. Ontology based context modeling and reasoning using OWL. Tagungsband: IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second, Seiten 18–22. IEEE, 2004.

[Weiser 91] Mark Weiser. The computer for the 21st century. Scientific American, 265(3):94–104, 1991.

[Wendt 13] Stefan Wendt, Philipp Lehsten, Djamshid Tavangarian. Gamification as a Service - Die Integration von spieletypischen Elementen in die Pervasive Universität. In Matthias Horbach, Hrsg., Tagungsband: GI-Jahrestagung, Band 220 of LNI, Seiten 346–360. GI, 2013.

[Wittmann 13] Michael Wittmann, Georg Wittmann, Ernst Stahl, Stefan Weinfurter. Digitalisierung der Gesellschaft. ibi research an der Universität Regensburg GmbH, 2013.

[Zender 10] Raphael Zender. Service-basierte Infrastruktur für pervasive Lehr- und Lernarrangements. Dissertation, Universität Rostock, 2010.

Eigene Publikationen und ihr Beitrag

Im Zusammenhang mit dieser Arbeit entstand eine Reihe von Publikationen entstanden. Diese dokumentieren nicht, wie in anderen Forschungsvorhaben, einzelne Aspekte des Gesamtvorhabens, sondern beschreiben die Entwicklung des Vorhabens als solches. Daher werden sie an dieser Stelle separat aufgeführt und ihr Beitrag zu CASA wird genannt.

- Philipp Lehsten, Andreas Thiele, René Zilz, Enrico Dressler, Raphael Zender, Ulrike Lucke, Djamshid Tavangarian. Dienste-basierte Kopplung von virtueller und Präsenzlehre. In Silke Seehusen, Ulrike Lucke, Stefan Fischer, Hrsg., Tagungsband: DeLFI, Band 132 of LNI, Seiten 77–88. GI, 2008

  Dieser frühe Konferenzbeitrag befasst sich mit der Service-orientierten Kopplung zwischen einer Präsenzveranstaltung und einer parallel stattfinden Veranstaltung in einer virtuellen Umgebung. Dabei wurde bereits das Stud.IP verwendet, um die Integration zwischen den Veranstaltungen zu leisten. Der Fokus lag dabei auf dem didaktischen Mehrwert sowie der technischen Kopplung über Technologie- und Systemgrenzen hinweg.

- Philipp Lehsten, Raphael Zender, Ulrike Lucke, Djamshid Tavangarian. A serviceoriented approach towards context-aware mobile Learning Management Systems. Tagungsband: PerCom Workshops, Seiten 268–273. IEEE, 2010

  Dieser Konferenzbeitrag griff die Problematik auf, dass die Lernmanagementsysteme wie Stud.IP nicht aus sich selbst heraus in der Lage sind sich den beschränkten Darstellungsmöglichkeiten auf mobilen Geräten anzupassen. Außerdem sind sie nicht in der Lage, die Kontextinformationen, wie zum Beispiel die Position der Veranstaltung oder des Nutzers, zu verwenden. Daher wurde in diesem Beitrag ein System vorgestellt, das unter Nutzung von Web Services und einem wissensbasierten System die relevanten Informationen aus dem Stud.IP zusammenfasst und über eine Smartphone-Anwendung verfügbar macht. Dieses System kann als Vorstufe zu CASA bezeichnet werden, da es bereits einige der Systembausteine verknüpft. Entgegen dem späteren Konzept wurde hier noch mit einer dedizierten Android-Anwendung gearbeitet, was einerseits eine Beschränkung auf eine Plattform darstellt und andererseits die Installation von Software auf dem mobilen Gerät voraussetzt. Würde man jedoch heute einen CASA-Knoten in die vorgeschlagene Architektur aufnehmen und die App als Anwendung mit CASA-Plugin betrachten, wäre dies mit dem CASA-Konzept vereinbar und eine sinnvolle künftige Erweiterung.

- Philipp Lehsten, Raphael Zender, Ulrike Lucke, Djamshid Tavangarian. Ein Serviceorientierter Ansatz für kontextbewusste Mash-Ups in mobilen Lernmanagementsystemen. In Ulrik Schroeder, Hrsg., Tagungsband: Mensch & Computer Workshopband, Seiten 216–221. Logos Verlag, 2010

  Dieser Konferenzbeitrag entstand kurz nach dem vorigen und erweitert das darin vorgestellte Konzept um die Idee der Mash-Ups. Diese dienen darin der Personalisierung der Lernerfahrung. Dieser Weg wurde jedoch nicht weiter verfolgt, da sich die bereits erwähnten Probleme der Plattformabhängigkeit und Heterogenität zeigten. Jede Weiterentwicklung wäre eine spezielle Entwicklung für dieses spezifische LMS und dessen Inhalt und Struktur gewesen und hätte damit einen unvertretbar großen Aufwand bei Weiterentwicklungen bedeutet. Außerdem wäre die dynamische Komposition von Mash-Ups eine Herausforderung in vielen Bereichen, von der technischen Kopplung bis zur semantischen Beschreibung der Fähigkeiten der Anwendung.

- Philipp Lehsten, Ulrike Lucke, Djamshid Tavangarian. Conceptual Integration of Security Management in Designing Context-aware Environments. In Wolfgang Karl, Dimitrios Soudris, Hrsg., Tagungsband: ARCS Workshops. VDE-Verlag, 2011

  In diesem Konferenzbeitrag wurde ein System vorgestellt, das den Security-Aspekt in das Zusammenspiel zwischen Gerätesteuerung, Lernmanagementsystem und Kontextserver aufnimmt. Dazu wurde im Rahmen einer Bachelorarbeit die bestehende Software des LMS erweitert. Diese Erweiterung sollte es erlauben, raum- und nutzerspezifisch auf die Funktionalitäten der Raumtechnik zuzugreifen und gleichzeitig AAA (Authentication, Authorization und Accounting) vorzuhalten. So sollte der Zugriff und die Konfiguration von neuen Geräten vereinfacht und gleichzeitig der Sicherheitsaspekt von Anfang an berücksichtigt werden. Wie bereits in der letzten Publikation war rückblickend die Entwicklung zu spezifisch für dieses eine Geräte- und Raumensemble und hätte einen großen Aufwand bei der Verallgemeinerung für andere Geräte und Räume erfordert. Mit der heutigen CASA-Entwicklung könnte dieser Aspekt jedoch erneut aufgegriffen werden und im Rahmen von künftigen Erweiterungen kann auch die Regulierung von Gerätezugriffen durch CASA unterstützt werden.

- Philipp Lehsten, Alexander Gladisch, Djamshid Tavangarian. Context-Aware Integration of Smart Environments in Legacy Applications. In David V. Keyson, Mary Lou Maher, Norbert Streitz, Adrian David Cheok, Juan Carlos Augusto, Reiner Wichert, Gwenn Englebienne, Hamid K. Aghajan, Ben J. A. Kröse, Hrsg., Tagungsband: AmI, Band 7040 of Lecture Notes in Computer Science, Seiten 126–135. Springer, 2011

  Dieser Konferenzbeitrag stellt die Dienstintegration in den Mittelpunkt und zeigt, wie sich gerade Altanwendungen durch ein System wie CASA, das hier noch nicht so betitelt wird, erweitern lassen. Es wird in dem Beitrag erneut das Szenario zur Integration von Gerätediensten in eine Lernumgebung gewählt. Die Architektur weist bereits einen Intermediate Layer aus, der in Struktur und Funktion mit dem CASA-Knoten vergleichbar ist. Auch wenn das Szenario geblieben ist, wurde das Konzept erweitert und stellt die Dienstempfehlung und Integration in Altanwendungen in den Fokus. Außerdem wird ein Workflow vorgeschlagen, der eine Definition von Aktionen und Regeln verlangt, um die Integration zu organisieren.

- Philipp Lehsten, Sebastian Bader, Djamshid Tavangarian. CASA - A Context Aware Service Access. 3rd Workshop on Context-Systems Design, Evaluation & Optimisation (CoSDEO 2011), Copenhagen, 2011

  Dieser Poster-Beitrag ist der erste Beitrag, der die zentrale Idee von CASA zeigt. Es wird illustriert, wie sich das Problem der Heterogennität des Dienstzugriffs in ubiquitären Umgebungen durch eine sich an die Situation anpassende Nutzerschnittstelle angehen lässt. Dabei wird bereits die Idee des Crowdsourcing genannt, um Kontextmodelle, Regeln und Aktionen zu definieren.

- Philipp Lehsten, Djamshid Tavangarian. CASA - Ein Konzept zur nutzergetriebenen, kontextbasierten Dienstintegration in Lehr- und Lernumgebungen. In Paul Müller, Bernhard Neumair, Helmut Reiser, Gabi Dreo Rodosek, Hrsg., Tagungsband: DFNForum Kommunikationstechnologien, Band 217 of LNI, Seiten 1–9. GI, 2013

  Dieser Konferenzbeitrag stellt das CASA-Konzept sowie die Umsetzung der Fallstudie für das Stud.IP vor. Dabei wird im Besonderen der Fokus auf die Organisation verschiedener Knoten gelegt, die unterschiedliche Grade an Privatsphäre haben. Der monodirektionale Informationsfluss wird ebenso erläutert wie Kaskadierung der Dienstempfehlung. Dieser Beitrag stellt daher die erste Dokumentation des CASA-Systems mit einer Implementierung dar.

- Stefan Wendt, Philipp Lehsten, Djamshid Tavangarian. Gamification as a Service - Die Integration von spieletypischen Elementen in die Pervasive Universität. In Matthias Horbach, Hrsg., Tagungsband: GI-Jahrestagung, Band 220 of LNI, Seiten 346–360. GI, 2013

  Dieser Konferenzbeitrag behandelt die Erweiterung des CASA-Systems um den Aspekt der Gamification. Die Realisierung entstand im Rahmen einer Bachelorarbeit und zeigt eine Methode, die Neugier der Anwender zu nutzen, um die verschiedenen Systeme einer Hochschule zu erkunden und sie gleichzeitig zur aktiven Nutzung zu motivieren. Das System zeigt, wie sich durch Nutzung einer Service-orientierten Architektur dezentrale Provider von virtuellen Belohnungen verbinden lassen und zudem zeigt es eine Umsetzung mit CASA als einem dieser Provider und dem Stud.IP als graphische Schnittstelle.

- Philipp Lehsten, Sebastian Bader, Djamshid Tavangarian. CASA - Context-Aware Service Access. Journal of Integrated Design and Process Science, 18(1):21–38, 2014

  Dieser Journal-Beitrag fasst die Entwicklung von CASA zusammen und entwickelt auf Basis von Herausforderungen und Beobachtungen aus dem Bereich des Zugriffs auf allgegenwärtige Dienste die Anforderungen, die auch für die Entwicklung von CASA maßgeblich waren. Es werden die Strukturen innerhalb von CASA erläutert und die Konzepte wie Regeln und Aktionen werden formalisiert beschrieben. Abschließend wird neben der Vorstellung der Stud.IP-Fallstudie eine konzeptionelle Übertragung des CASA-Ansatzes auf die ebenfalls im Umfeld von MuSAMA entwickelten Goalaviors vorgestellt. Diese beschreiben, ähnlich wie die Regeln in CASA, eine Menge von Zuständen und Zielen. Mit der Verknüpfung beider Systeme könnte CASA auch in der Lage sein, unscharfe Situationen zu analysieren und auf Basis unvollständiger Informationen zu agieren.

Selbstständigkeitserklärung

Philipp Lehsten
Ausbau 4D
18069 Lambrechtshagen

Hiermit versichere ich, dass ich die von mir vorgelegte Arbeit selbstständig verfasst habe, dass ich die verwendeten Quellen, Internet-Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Rostock, den 16. Februar 2018

_____

Philipp Lehsten

Lebenslauf

Philipp Lehsten

Geboren am	22. Juni 1985
Geburtsort	Güstrow
Staatsbürgerschaft	Deutsch
E-Mail	philipp.lehsten@gmail.com
Fremdsprachen	Englisch fließend in Schrift und Sprache, Grundkenntnisse in Spanisch und Französisch

Anstellungen und Erfahrungen

seit 03/2015	Anstellung als Projektleiter in der Software-Entwicklung bei der ITZ Rostock GmbH
06/2013-03/2015	Anfertigung der Dissertationsschrift und Umsetzung des in der Promotion entwickelten CASA-Konzeptes für das Rechenzentrum der Universität Rostock im Rahmen einer freiberuflichen Entwicklertätigkeit
03/2013-06/2013	Organisation und Leitung eines Symposiums zum Thema „Zu viel des Smarten? - Wie viel Assistenz braucht der Mensch?“ mit mehreren renommierten Referenten und über 50 Teilnehmern
01/2010-04/2013	Promotiosstipendium im Rahmen des DFG-Graduiertenkolleges MuSAMA zum Thema „Intelligente Integration und Dissemination von Diensten in Smart Environments“
06/2009-12/2009	Masterarbeit zum Thema „Kontext- und Service-basierte Erweiterung der Lernplattform Stud.IP für eine mobile Nutzung“
02/2008-12/2009	Studium Informationstechnik/Technische Informatik an der Universität Rostock (Master-Stufe)
10/2008-04/2009	Praktikum bei der IBM Deutschland Research and Development GmbH in Böblingen mit dem Thema „Entwicklung einer offenen, generischen Infrastruktur bzw. Architektur für Online Games“
02/2008	Bachelor of Science, Technische Informatik
10/2004-02/2008	Studium Informationstechnik/Technische Informatik an der Universität Rostock (Bachelor-Stufe)

Rostock, 15. Februar 2018

Thesen

1. Die Heterogenität von Nutzergruppen, Endgeräten und Domänen ist die zentrale Herausforderung, die es bei der Dienstvermittlung und -nutzung in intelligenten Umgebungen zu überwinden gilt.
2. Eine dezentralisierte und vielgestaltige Umgebung wie ein Hochschul-Campus stellt an das Paradigma der pervasiven Umgebungen neue Anforderungen im sozialen, organisatorischen und technischen Bereich, die über die Anforderungen klassischer intelligenter Umgebungen hinausgehen.
3. Die Einbeziehung des Nutzers in die Akquise von neuen, möglicherweise relevanten Diensten und die anschließende Dissemination dieser Dienstkandidaten nach nutzergenerierten Filtern ermöglicht sowohl die Wahrung von Aktualität als auch die stete Weiterentwicklung von Systemen zur Dienstempfehlung.
4. Die kontextsensitive Einbettung von möglichen Dienstkandidaten in Altanwendungen, die den Nutzern bekannt und vertraut sind, ermöglicht eine effiziente, homogene und medienbruchfreie Repräsentation neuer Dienste in Umgebungen mit statischen Softwareumgebungen.
5. Der Schutz von Daten und die Wahrung des Rechts auf informationelle Selbstbestimmung sind notwendige Bestandteile von großflächigen, pervasiven Systemen und müssen bereits zum Designzeitpunkt berücksichtigt und fokussiert werden.
6. Die Aufteilung von Domänen nach ihrem Grad an Privatsphäre (privat/gruppenöffentlich/öffentlich) und ein daran orientierter, monodirektionaler Informationsfluss stärken sowohl den Datenschutz der Nutzer als auch deren Möglichkeiten zur informationellen Selbstbestimmung.
7. Für kontextsensitive Systeme ist eine Peer-to-Peer-ähnliche, knotenbasierte Architektur gut geeignet, da mit Hilfe dieser eine strukturierte Kooperation innerhalb von Domänen ebenso möglich ist wie eine Trennung zwischen verschiedenen Bereichen.
8. Die Erweiterung des Lernmanagementsystems Stud.IP um das CASA-System hat die Umsetzbarkeit und Sinnhaftigkeit des Konzeptes gezeigt.
9. Die Initiierung und Aufrechterhaltung der Nutzer-Motivation zur Beteiligung an nutzergetriebenen Systemen stellt einen kritischen Erfolgsfaktor innerhalb dieser Systeme dar und ist daher in jeder Entwicklungsphase zu berücksichtigen.