Kontextadaptive Anwendungsarchitekturen für das mobile Cloud Computing

Dissertation

zur Erlangung des akademischen Grades
Dr. rer. nat.
an der Fakultät für Mathematik, Informatik und Naturwissenschaften
der Universität Hamburg

eingereicht beim Fach-Promotionsausschuss Informatik von
Gabriel Orsini
aus Hamburg

Hamburg, 6. Dezember 2017

Vorsitzender der Prüfungskommission: Prof. Dr. W. Maleej
Erstgutachter: Prof. Dr. W. Lamersdorf
Zweitgutachter: Prof. Dr. W. Menzel

Tag der Disputation: 5. Dezember 2017

Zusammenfassung

Der stetige technologische Fortschritt im Bereich der Informationstechnologie hat innerhalb der letzten Jahre zu einer fortlaufenden Miniaturisierung der beteiligten Geräte geführt. Diese Miniaturisierung wird als der entscheidende Wegbereiter für die Realisierung von Mark Weisers Vision des *Ubiquitous Computings* gesehen, die die Durchdringung von Alltagsgegenständen mit Informationstechnologie beschreibt.

Die Miniaturisierung hat bereits eine Reihe neuer Kategorien von Geräten wie Smartphones oder Tablet-PCs hervorgebracht, die sich aufgrund ihrer Größe durch eine hohe Mobilität im Zusammenhang mit ihrer Benutzung auszeichnen. Diese mobilen Geräte können allerdings nicht immer den Anforderungen der Nutzer an unterstützte Anwendungen, Speicherkapazität und Akkulaufzeit gerecht werden, da der fortlaufenden Miniaturisierung nur in Teilen eine entsprechende Leistungssteigerung folgt. Zukünftig ist von einem weiteren Anstieg dieser Anforderungen an mobile Geräte auszugehen, die die Kooperation ressourcenbeschränkter mobiler Geräte mit weiteren Ressourcen in ihrer Umgebung erforderlich machen, um die Anforderung der Nutzer zu erfüllen. Diese Kooperation wird auch als mobiles Cloud Computing beschrieben. Ein Beispiel findet sich in der Auslagerung von Berechnungen durch Sprachassistenzsysteme wie Apple Siri oder Microsoft Cortana, die Eingaben ihrer Nutzer zur Verarbeitung in eine entfernte Infrastruktur übertragen.

Um die beschriebene Kooperation zu realisieren, gilt es allerdings zu berücksichtigen, dass sich durch die hohe Mobilität dieser Geräte ihre Umgebung, also ihr Kontext, häufig verändert und eine entsprechende Adaption der Interaktion mit der Infrastruktur erforderlich macht. Dies wiederum führt zu einem wachsenden Bedarf an Softwaresystemen, die in der Lage sind, sich selbstständig an ihren Ausführungskontext anzupassen. Eine kontextadaptive Anwendung kann sich so beispielsweise bei einer niedrigen Übertragungsbandbreite durch eine Reduktion der Qualität an einen veränderten Kontext anpassen und so die Nutzbarkeit der Anwendung sicherstellen. Ebenso kann sie einen niedrigen Ladezustand der Batterie erfassen und entsprechende Teile ihrer Implementierung durch eine energieeffiziente Variante ersetzen, um die verbleibende Nutzungsdauer des mobilen Gerätes im Batteriebetrieb zu maximieren.

Entsprechend liegt der zentrale Forschungsaspekt dieser Dissertation auf der Untersuchung und Erarbeitung von Konzepten für eine kontextadaptive Anwendungsarchitektur, die es Anwendungen auf mobilen Geräten erlaubt, sich mittels verschiedener Adaptionsstrategien anzupassen. Dies ermöglicht es die umgebende Infrastruktur im Rahmen des beschriebenen Kooperationsszenarios der Sprachassistenz zu nutzen und die Adaptionsstrategien in Abhängigkeit vom aktuellen und zukünftigen Kontext eines mobilen Geräts automatisiert und proaktiv anzupassen.

Als erstes Ergebnis dieser Arbeit wird hierzu eine Anwendungsarchitektur für mobiles Cloud Computing vorgestellt, die darauf abzielt, die verteilte Ausführung unterschiedlicher mobiler Anwendungen zu unterstützen. Mithilfe verschiedener Adaptionsstrategien erlaubt es diese Architektur die Funktionalität eines mobilen Gerätes zu erweitern, bestehende Funktionalität zu beschleunigen oder eine insgesamt höhere Verfügbarkeit und Nutzbarkeit mobiler Anwendungen zu ermöglichen. Anschließend wird gezeigt, dass Kontextinformationen dazu verwendet werden können, das mobile Cloud Computing effizienter zu gestalten. Als zweites Ergebnis dieser Arbeit wird entsprechend ein generischer Prozess zur Kontextadaption vorgestellt, welcher die erforderlichen Adaptionsstrategien weitgehend eigenständig lernt und dabei den zukünftigen Kontext eines mobilen Gerätes antizipiert.

Durch diese Beiträge werden Entwickler in die Lage versetzt, mobile Anwendungen zu entwickeln, die verteilt und kontextadaptiv ausgeführt werden. Hierzu müssen sie sich nicht mit den Details der Verteilung und Adaption auseinandersetzen, da diese mobilen Anwendungen selbständig die Veränderungen ihres Nutzungskontextes lernen und antizipieren.

Abstract

The ongoing technological progress in the field of information technology has led to an ongoing miniaturization of the devices involved. This miniaturization is seen as the main enabler for the realization of Mark Weiser's vision of Ubiquitous Computing, which describes the pervasion of everyday objects with information technology.

This miniaturization has already brought about a variety of new devices, such as smartphones or tablets, which, due to their size, are characterized by enabling high mobility in conjunction with their use. These mobile devices, however, may not always meet the requirements of the users with regard to supported applications, storage capacity, and battery life, as the ongoing miniaturization is followed only partially by an opposing increase in performance. In the future, we expect increasing challenges for these mobile devices, which require the cooperation of resource-constrained mobile devices with additional resources in their environment in order to meet the users' requirements. This cooperation is also referred to as Mobile Cloud Computing. An example is the computational offloading in natural language processing by personal assistants such as Apple Siri or Microsoft Cortana, which transfer the input of their users to a remote infrastructure for processing.

In order to achieve the described cooperation, the high mobility of these devices and their frequently changing environment, i.e. their context, has to be taken into account and the interaction with the infrastructure needs to be adapted accordingly. This in turn leads to a growing need for software systems that are capable of adapting themselves independently to their respective execution contexts. A context-adaptive application is able to, for example, adapt to a changed context like a decreasing transmission bandwidth by reducing the quality, thus ensuring the usability of the application. Likewise, it can detect a low charge state of the battery and replace corresponding parts of its implementation with an energy-efficient variant in order to maximize the remaining uptime of the mobile device in battery mode.

Accordingly, the main research focus of this dissertation is on the investigation and elaboration of concepts for a context-adaptive application architecture, which allows applications on mobile devices to adapt themselves by means of different adaptation strategies. For example, by using the surrounding infrastructure to enable the mentioned scenario of distributed natural language processing by adapting the distribution strategy automatically and proactively to the current and future context of a mobile device.

The first result of this work represents an application architecture for mobile cloud computing, which aims to support the distributed execution of a multitude of mobile applications. Supported by different adaptation strategies, this architecture allows to extend the functionality of a mobile device, to accelerate existing functionality or to achieve a higher availability and usability of mobile applications. Subsequently, it is shown that context information can be used to increase the efficiency of mobile cloud computing. Hence, the second result of this thesis is a generic context adaptation process, that learns the necessary adaptation strategies to a large extent independently while anticipating the future context of a mobile device.

The developed architecture enables developers to build mobile applications that are distributed and context-adaptive without having to deal with the details of the distribution and adaptation as these mobile applications independently learn and anticipate changes in their individual usage context.

Danksagung

In meiner Tätigkeit als Doktorand und wissenschaftlicher Mitarbeiter haben mich eine Reihe von Menschen im Entstehungsprozess dieser Arbeit begleitet. Jeder von ihnen hat auf seine Weise zum Gelingen beigetragen. Allen voran gilt mein großer Dank Herrn Lamersdorf. Er gab mir stets alle Freiheiten in Bezug auf Forschung und Lehre, hat mich von Anfang an in diesem Vorhaben unterstützt, war immer zur Stelle und wurde seiner Rolle als Doktorvater mehr als gerecht. Ebenso möchte ich Herrn Menzel für seine Unterstützung und die Bereitschaft zur Übernahme des Zweitgutachtens danken.

Mein Dank gilt ebenso meinen Kollegen bei VSIS, die dieses Vorhaben begleitet haben. Hier gilt mein besonderer Dank Dirk, der mir stets ein Vorbild war. Seine Unterstützung beim Umschiffen der Untiefen des akademischen Handelns, aber auch die vielen Ermutigungen haben ihren ganz eigenen Teil zum Gelingen dieser Arbeit beigetragen. Alex, Kai und Lars danke ich für die sehr gute technische Unterstützung und die vielen wichtigen Impulse. Ebenso danke ich Christopher, Fabian, Julian, Kristof und Wolf für die gute Zusammenarbeit, die vielen neuen Ideen und vor allem für die Ablenkungen im genau richtigen Moment. Gleichzeitig danke ich Anne und Volker für die Organisation der vielen bürokratischen Stolperfallen, die sich an der Universität Hamburg üblicherweise ergeben. Ebenso danke ich den zahlreichen Studierenden, deren Arbeiten ich begleiten durfte, insbesondere gilt mein Dank hier Fabian Besner, Anne-Victoria Meyer und Simone Stella.

Im Laufe der Jahre gab es natürlich auch außerhalb der Universität eine Reihe wichtiger Menschen, die mich inspiriert, unterstützt und mir beigestanden haben. Hier gilt mein Dank Hanna, die mich als Vorbild zu diesem Experiment inspiriert hat. Gleichzeitig danke ich meinen Freunden für ihr offenes Ohr, ihre guten Ratschläge und ständigen Ermutigungen, auf dem richtigen Weg zu sein.

Natürlich gilt mein Dank auch meiner Familie. Allen voran meinen Eltern, die mich sowohl zu diesem Vorhaben motiviert als auch im Laufe der Zeit unterstützt haben. In gleichem Maße gilt mein Dank natürlich auch meiner Schwester, die mir stets einen neuen Blickwinkel auf viele Dinge im Leben eröffnet hat. Ebenso gilt mein Dank dem Teil meiner Familie, der mich aus der Ferne und bei vielen Besuchen ebenso begleitet, immer weiter ermutigt und unterstützt hat.

Hamburg im Juni 2017

Gabriel Orsini

Inhaltsverzeichnis

1. Einleitung

> *„Ubiquitous computing has as its goal the enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user.“ [Wei93]*

Mark Weisers Sichtweise des Computers im 21. Jahrhundert [Wei91], die den Begriff des *Ubiquitous Computings* prägte und durch das einleitende Zitat von ihm dargestellt wird, beschreibt die Transformation der heutzutage sichtbaren Computer – bis hin zur Unsichtbarkeit, die durch die Durchdringung von Alltagsgegenständen mit Informationstechnologie entsteht. Diese Vision basiert auf dem Stand der Technik der frühen 1990er-Jahre. Viele der damaligen technischen Restriktionen existieren heute nicht mehr oder nur noch in abgeschwächter Form. Entsprechend findet sich Weisers Vorstellung dieser intelligenten Gegenstände heute unter anderem in mobilen Geräten wie Smartphones, die uns die Realisierung des unsichtbaren Computers bereits ein großes Stück näher gebracht haben.

Die hohe Relevanz dieser Entwicklung zeigt sich bei der Betrachtung aktueller Trends und Entwicklungen im Bereich des Mobile Computings. So findet die Gestaltung und Ausrichtung von Inhalten und Produkten immer öfter unter dem Leitbild „Mobile First“ statt [Mic14] – ein Ansatz, bei dem sich die Darstellung von Inhalten primär an den Eigenschaften und Restriktionen mobiler Endgeräte orientiert.

Mobile Geräte haben sich für eine Vielzahl von Menschen zu selbstverständlichen Alltagsgeständen entwickelt, die weit mehr Aufgaben übernehmen, als ihre Nutzer an Termine und Telefonnummern zu erinnern. Stattdessen unterstützen sie uns intelligent bei einer Vielzahl unserer täglichen Aufgaben oder erledigen diese selbstständig für uns, beispielsweise die Routenfindung oder die Suche nach Informationen zur Wetterlage oder Verkehrssituation. Belege für die Relevanz dieser Geräte in unserem Alltag finden sich damit vor allem im Verbreitungsgrad und der täglichen Nutzungsdauer dieser Geräte. So besitzen beispielsweise 80 Prozent aller Deutschen ein Smartphone, knapp jeder Fünfte nutzt es dabei bereits über zwei Stunden pro Tag [For16]. Weitere Belege für die Relevanz dieser Entwicklung finden sich im weltweiten Absatz von Smartphones – im Jahr 2015 wurden 1,4 Milliarden dieser Geräte verkauft [IDC16a]; ihre Verbreitung ist damit so weit vorangeschritten, dass gut jeder vierte Mensch ein solches Gerät besitzt [Sta16].

Mit Blick auf aktuelle Prognosen verschiedener Marktbeobachter ist davon auszugehen, dass sich diese Trends weiter verstärken. So wird aktuell geschätzt, dass im Jahr 2020 die Anzahl verkaufter *Wearables* weltweit bei 82,5 Millionen Stück liegen wird [IDC16b]. Insgesamt wird jedoch angenommen, dass durch weitere neue Kategorien von Geräten die Anzahl der mit dem Inter-

net verbundenen Geräte noch weitaus stärker steigt. Die Anzahl der am sogenannten *Internet of Things* teilnehmenden intelligenten Gegenstände wird bis in das Jahr 2020 auf 20,8 Milliarden geschätzt [Gar15]. Die Potenziale dieser Entwicklungen sind vielfältig, neue Nutzungsszenarien entstehen täglich und können heute nur in Ansätzen abgeschätzt werden. So erlauben mobile Geräte auch gänzliche neue Anwendungen, die mithilfe der in ihnen verbauten Sensoren sowohl ihren Nutzer, ihre Umgebung als auch ihren Nutzungskontext wahrnehmen können. Eine Eigenschaft, die im Zusammenhang mit der Mobilität dieser Geräte immer öfter als ein entscheidendes Qualitätskriterium für die Nutzbarkeit dieser Anwendungen angesehen wird [Gei08].

Ein Großteil dieser Geräte arbeitet wegen ihrer begrenzten Rechen- und Speicherressourcen und ihrer begrenzten Energievorräte jedoch nicht autonom, sondern kommuniziert drahtlos mit anderen Geräten – diese Kooperation und Interaktion ist mitverantwortlich für ein bereits im Jahr 2016 monatlich drahtlos übertragenes Datenvolumen von weltweit 3,7 Exabyte, eine Zunahme von 74 Prozent zum Vorjahr [Cis16].

Welche Herausforderungen diese Entwicklungen mit sich bringen, wie sich Anwendungen für mobile Geräte so entwickeln lassen, dass sie zur Erweiterung ihrer beschränkten Ressourcen mit umliegenden Geräten im Rahmen eines mobilen Cloud Computings kooperieren können und wie diese Interaktion sinnvoll in Abhängigkeit vom Kontext dieser Geräte adaptiert werden kann, soll in der vorliegenden Dissertation untersucht werden.

1.1. Grenzen der Mobilität

Die eingangs erwähnten Trends wurden durch die beschriebene Weiterentwicklung mobiler Geräte begünstigt. So stieg beispielsweise die Rechenleistung eines typischen Smartphones wie dem iPhone um den Faktor zwei mit jeder Generation [Tom16], die Kapazität des Akkus jedoch lediglich um zehn Prozent [ZN10]. Entsprechend wird davon ausgegangen, dass mobile Geräte aufgrund der zunehmenden Miniaturisierung auch zukünftig einer Ressourcenbeschränkung unterliegen werden und ihre Leistungsfähigkeit im Vergleich zu stationären Geräten eingeschränkt sein wird [ASA+14].

Dies umfasst vor allem eine begrenzte Rechen- und Speicherkapazität, eine vom Kontext des Gerätes abhängige wechselnde Konnektivität und schwankende Bandbreite zu anderen Geräten sowie den stark begrenzten Energievorrat mobiler Geräte [LSJ+13]. Insbesondere die letzte Einschränkung wird von Nutzern dieser Geräte jedoch als entscheidend eingestuft. So nennt eine Studie der IDC Research Inc. aus dem Jahr 2014 die Akkulaufzeit über alle relevanten Mobilplattformen von den Nutzern als mit Abstand wichtigstes Kriterium für den Kauf eines Smartphones [Jer14].

Was hingegen die Nutzung mobiler Geräte angeht, besteht der Anspruch der Nutzer darin, die Dienste, die sie vorher stationär genutzt haben, auch mobil zu nutzen. Dies schließt ressourcenintensive Anwendungen wie Bild- und Videobearbeitung, aber auch Spiele ein und zeigt sich im Vorhandensein ent-

sprechender Anwendungen in den App-Stores [BRS06]. Prominente Beispiele sind die Video- und Bildbearbeitung, Gesichtserkennung [Ope16], Cloud Gaming [Nvi15] und Sprachassistenten für mobile Geräte [App13]. Aufgrund der zuvor beschriebenen existierenden Einschränkungen mobiler Geräte setzen einige mobile Anwendungen wie Sprachassistenten und Cloud Gaming bereits auf eine Kooperation mit anderen leistungsfähigeren Geräten, um die jeweiligen Beschränkungen mobiler Geräte zu umgehen. Es ist anzunehmen, dass dieser Kooperationsbedarf wegen der weiteren Miniaturisierung und weiteren Vernetzung zukünftiger intelligenter Gegenstände weiter zunehmen wird.

Um den genannten Einschränkungen mobiler Geräte zu begegnen und diese Kooperation sinnvoll umzusetzen, bieten sich die seit einigen Jahren verfügbaren Standards für Mobilkommunikation (IEEE 802.11ac [IEE13], LTE Advanced [3GP13]) an. Diese Standards ermöglichen die spontane Kooperation mobiler Geräte, einerseits mit leistungsstärkeren Geräte in der näheren Umgebung, sogenannten *Cloudlets* [SBCD09], oder mit zentralisierten Cloud-Diensten [AFG+10]. Letztere bieten durch das permanente Vorhalten von Rechen- und Speicherleistung sowie die Bereitstellung und Abrechnung im Minutentakt nahezu unbegrenzte Ressourcen auf Abruf an.

Diese Kooperation zwischen mobilen Geräten und die Integration von Cloud-Diensten in das Mobile Computing wird auch als *mobiles Cloud Computing* bezeichnet. Als wesentliche Herausforderungen des mobilen Cloud Computings gelten, neben weiteren, vor allem die hohe Heterogenität der beteiligten Geräte, die sinnvolle Aufteilung von Funktionalität im Rahmen der Kooperation zwischen mobilen und stationären Geräten [AGS+15] [LAS+15] und die hohe Dynamik, der mobile Geräte im Hinblick auf ihren Nutzungskontext unterliegen und die einen konstanten Anpassungsbedarf erfordern [FLR13]. Abbildung 1.1 zeigt diese Kooperation und veranschaulicht insbesondere den Anpassungsbedarf, der aus dem wechselnden Kontext mobiler Geräte entsteht. So kann ein Anwender Cloud-Dienste nutzen, um die Funktionalität seines mobilen Geräts zu erweitern, beispielsweise um Dokumente aus einer zentralen Quelle abzurufen und sie nach der Bearbeitung dort zu aktualisieren. Der hierbei wechselnde Kontext, insbesondere die wechselnde Konnektivität, stellt eine von vielen Herausforderungen dar, die im folgenden Abschnitt konkretisiert werden, um auf die zentrale Forschungsfrage dieser Arbeit hinzuarbeiten.

1.2. Problemstellung

Wie zuvor beschrieben sind mobile Geräte in den letzten Jahren deutlich leistungsfähiger geworden und sammeln über ihre eingebauten Sensoren zusätzlich Kontextinformationen. Weiterhin bringt die Verfügbarkeit von Cloud-Diensten für die breite Öffentlichkeit neue Anwendungsmöglichkeiten in Bezug auf die Verarbeitung und Speicherung großer Datenmengen mit mobilen Geräten. Ebenso ist auch die Anbindung der mobilen Geräte an die Infrastruktur durch Mobilfunkstandards wie 802.11ac und LTE Advanced deutlich beschleunigt worden. Die wesentlichen Restriktionen mobiler Geräte bestehen

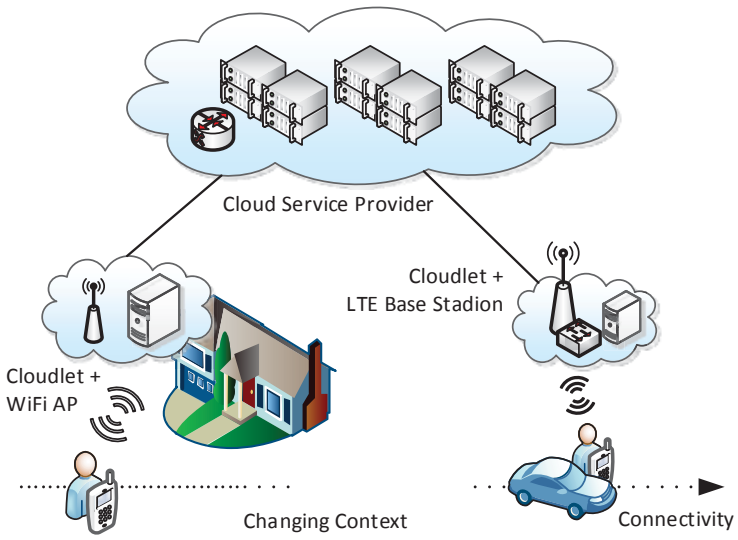

Abbildung 1.1.: Exemplarische Darstellung des mobilen Cloud Computings

jedoch weiterhin in einer beschränkten Rechen- und Speicherkapazität, einer ständig wechselnden Verbindungsqualität und einer begrenzten Akkuleistung [FLR13, Sat17b]. Der als vielversprechend angesehene Lösungsansatz des mobilen Cloud Computings versucht diesen Problemen durch die zuvor beschriebene Kooperation mobiler Geräte mit weiteren verfügbaren Ressourcen zu begegnen.

Einen höheren Funktionsumfang oder eine verbesserte Leistung mobiler Anwendungen durch Kooperation mit weiteren Ressourcen zu erreichen, ist jedoch nicht trivial. Existierende Ansätze aus dem Bereich der (stationären) verteilten Systeme können oft nicht direkt auf das Mobile Computing und damit auf mobiles Cloud Computing übertragen werden, da mobile Geräte den zuvor erwähnten Restriktionen unterliegen. Der Versuch, den Defiziten mobiler Geräte zu begegnen, schafft damit zusätzliche Herausforderungen, welche neben weiteren in der Erkennung nutzbarer Ressourcen innerhalb eines laufend wechselnden Kontextes, der Entscheidung über die verteilte Ausführung einer mobilen Anwendung und der Aufrechterhaltung der Informationssicherheit liegen. Insbesondere der oft wechselnde Nutzungskontext erschwert die Anwendung bestehender Konzepte für verteilte Systeme, bei deren Entwicklung diese Restriktionen keine Berücksichtigung finden mussten.

Aus den gezeigten Herausforderungen der wachsenden Anforderungen der Nutzer, der begrenzten Ressourcen mobiler Geräte, der wechselnden Konnektivität dieser Geräte und der hohen Dynamik, mit der sich der Nutzungskontext dieser Geräte ändert, ergibt sich die Problemstellung, die in der vorliegenden Arbeit untersucht werden soll.

Diese umfasst zuerst die aus den Herausforderungen abgeleiteten Bedarfe zur verteilten Ausführung mobiler Anwendungen, um die begrenzten Ressourcen mobiler Geräte durch die Nutzung ihrer Infrastruktur zu erweitern. Ebenso umfasst sie die für solch eine verteilte Ausführung erforderliche Ad-

aptionsfähigkeit einer entsprechenden *Anwendungsarchitektur*. Diese Anwendungsarchitektur hat die Aufgabe die verteilte Ausführung nicht nur effizient, sondern im Hinblick auf die wechselnde Konnektivität mobiler Geräte spontan möglich zu machen. Hierzu sollte sie einen solchen Adaptionsbedarf erkennen und umsetzen, um sich damit *kontextadaptiv* zu verhalten.

Die nähere Untersuchung dieser Problemstellung stützt sich auf die folgenden zwei Annahmen und soll in den darauffolgenden Forschungsfragen weiter konkretisiert werden:

- A1: Die Anforderungen der Nutzer steigen, und zukünftig werden vermehrt mobile Anwendungen entwickelt, die von einer kooperativen Ausführung durch mobiles Cloud Computing profitieren.
- A2: Die Berücksichtigung des Kontextes mobiler Geräte wirkt sich positiv auf die Effizienz der Kooperation im mobilen Cloud Computing aus.

Haupt-Forschungsfrage: Wie kann die verteilte Ausführung mobiler Anwendungen im mobilen Cloud Computing ermöglicht und unter Berücksichtigung des Nutzungskontexts effizient gestaltet werden?

Diese Haupt-Forschungsfrage lässt sich in zwei Forschungsfragen und ihre jeweiligen Teil-Forschungsfragen aufteilen, die in der vorliegenden Arbeit sukzessive beantwortet werden sollen:

Forschungsfrage 1: Welche Anwendungsarchitektur ist für eine verteilte Ausführung mobiler Anwendungen im mobilen Cloud Computing geeignet und anzupassen, sodass sie in der Lage ist, den Nutzungskontext zu berücksichtigen?

Um diese Frage zu beantworten, müssen die folgenden Teil-Forschungsfragen beantwortet werden:

- Welche Anwendungsdomänen können von einer solchen Architektur profitieren?
- Welche Herausforderungen bestehen durch die verteilte Ausführung mobiler Anwendungen unter Verwendung des mobilen Cloud Computings und können von einer angemessenen Lösung erfüllt werden?
- Welche Anforderungen sollte eine Anwendungsarchitektur für mobiles Cloud Computing abdecken, um die bestehenden Herausforderungen angemessen zu adressieren?
- Welche Konzepte für die verteilte Ausführung von Anwendungen existieren bereits, und wie können diese um Aspekte der Berücksichtigung des Kontexts für das mobile Cloud Computing erweitert und angepasst werden?
- Wie kann eine angemessene Unterstützung der Entwickler bei der Nutzung einer solchen Anwendungsarchitektur umgesetzt werden?

Forschungsfrage 2: Wie können Kontextinformationen verwendet werden, um die verteilte Ausführung mobiler Anwendungen im mobilen Cloud Computing positiv zu beeinflussen?

Um diese Frage zu beantworten, müssen die folgenden Teil-Forschungsfragen beantwortet werden:

- In welcher Form kann die Berücksichtigung des Kontextes die Benutzbarkeit mobiler Anwendungen positiv beeinflussen und die Nutzung der zu entwickelnden Anwendungsarchitektur optimieren?
- Wie kann ein zukünftiger Kontext antizipiert werden und hierbei mit wechselnder Qualität der Kontextdaten umgegangen werden?
- Wie kann das Expertenwissen der Entwickler angemessen in Adaptionsprozesse und die Verarbeitung der Kontextinformationen integriert werden?
- Wie kann die zusätzliche Komplexität, die durch die Berücksichtigung der Kontextinformationen entsteht, vor den Entwicklern solcher Anwendungen verborgen werden?

Der zentrale Forschungsaspekt der Arbeit liegt entsprechend auf der gemeinsamen Betrachtung zweier miteinander verbundener Problemstellungen: der effizienten verteilten Ausführung mobiler Anwendungen durch mobiles Cloud Computing und der Berücksichtigung des Nutzungskontextes dieser Anwendungen sowie der beteiligten Geräte, um durch diese beiden Aspekte sowohl die Anzahl der Szenarien, in denen solch eine Kooperation sinnvoll möglich ist, als auch die Effizienz dieser Kooperation zu erhöhen.

1.3. Zielsetzung

Die Verfügbarkeit von Cloud-Diensten und ausreichend leistungsfähigen mobilen Geräten schafft erstmals die Möglichkeit, die Potenziale des mobilen Cloud Computings für die breite Masse zu öffnen. Jedoch hat die Auswertung der Literatur gezeigt, dass eine Architektur, die die nötigen Funktionalitäten für die angestrebte kontextadaptive verteilte Ausführung mobiler Anwendungen bietet, bisher nur in Ansätzen existiert und die heute vorhandenen Potenziale damit noch ungenutzt bleiben. Etablierte Verfahren zur verteilten Ausführung können damit nur bedingt genutzt werden, da sie auf die hohe Dynamik hinsichtlich des wechselnden Kontextes mobiler Umgebungen nicht ausgelegt sind [KCK11, FLR13, Sat17a].

Ausgehend von der beschriebenen Problemstellung soll in dieser Arbeit ein Konzept für eine kontextadaptive Anwendungsarchitektur entwickelt werden. Dieses ermöglicht es mobilen Anwendungen, verteilt und kontextadaptiv ausgeführt zu werden, um die in den beiden vorherigen Abschnitten genannten Beschränkungen zu umgehen und damit der Anforderung der Nutzer, eine

möglichst große Anzahl von Anwendungstypen zu unterstützen, gerecht zu werden. Hierzu sollen die folgenden konkreten Forschungsziele adressiert werden.

Das erste Forschungsziel beschreibt eine Anwendungsarchitektur für mobiles Cloud Computing, die die Heterogenität und die begrenzten Ressourcen mobiler Geräte sowie den schnell wechselnden Kontext dieser Geräte berücksichtigt und damit die Umsetzung von Adaptionsbedarfen zur Laufzeit einer mobilen Anwendung ermöglicht. In diesem Zusammenhang soll die zu entwickelnde Anwendungsarchitektur zusätzlich die Entwickler angemessen bei der Entwicklung entsprechender mobiler Anwendungen unterstützen. Dieses Forschungsziel zielt damit auf die erste Forschungsfrage ab.

Das zweite Forschungsziel umfasst den Entwurf eines generischen Prozesses zur Kontextadaption, der es mobilen Geräten ermöglicht, sowohl auf ihren aktuellen als auch ihren zukünftigen Nutzungskontext zu schließen und hieraus Entscheidungen hinsichtlich eines aktuellen oder zukünftigen Adaptionsbedarf zu treffen. Der Dienst soll hierzu eine Prognose für zukünftige Kontextattribute beinhalten, auch wenn hierfür nur wenige Informationen, unterschiedliche Kontextdaten oder eine wechselnde Datenqualität zugrunde liegen. Dieser Adaptionsdienst soll sich dabei in die Anwendungsarchitektur aus dem ersten Forschungsziel integrieren und dem Entwickler eine angemessene Unterstützung im Hinblick auf die Nutzung der Informationen zur Kontextadaption bieten. Dieses Forschungsziel zielt entsprechend auf die zweite Forschungsfrage ab.

Ergänzend soll eine Systemunterstützung entworfen werden, die die beiden Forschungsziele miteinander kombiniert und Entwickler in die Lage versetzt, kontextadaptive mobile Anwendungen zu entwickeln, die sich im mobilen Cloud Computing effizient nutzen lassen. Das Endergebnis stellt ein Konzept dar, dessen Umsetzbarkeit durch eine prototypische Implementierung gezeigt wird und das im Rahmen einer Simulation anhand verschiedener Beispielanwendungen evaluiert wird. Entsprechend zielt das Konzept auf eine Unterstützung des mobilen Cloud Computings für möglichst viele, unterschiedliche Kategorien mobiler Anwendungen ab. Diese Unterstützung soll auch dann ermöglicht werden, wenn häufige und komplexe Adaptionsbedarfe erkannt und bewältigt werden müssen. Abbildung 1.2 zeigt entsprechend im rechts dargestellten Anwendungsfall das von dieser Arbeit adressierte Szenario: Unterschiedliche mobile Anwendungen sollen die sie umgebende Infrastruktur für ein mobiles Cloud Computing nutzen können, auch wenn ein schnell wechselnder Kontext eine häufige und komplexe Adaption an den selbigen erfordert.

Das Ziel der vorliegenden Dissertation ist es damit, Konzepte bereitzustellen, die die zusätzliche Komplexität des mobilen Cloud Computings beherrschbar machen und eine angemessene Unterstützung der Entwickler erlauben, um die verteilte und kontextadaptive Ausführung von Anwendungen zwischen mobilen und stationären Geräten zu ermöglichen. Hierdurch soll die Nutzbarkeit mobiler Geräte durch eine neue Klasse verteilt ausführbarer Anwendungen verbessert werden, die den Nutzungskontext in ihr Verhalten einbeziehen.

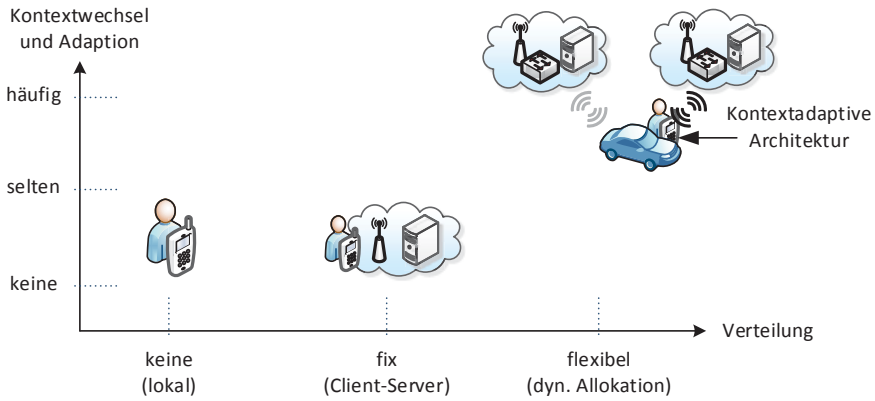

Abbildung 1.2.: Unterstützte Kooperationsformen

1.4. Ergebnisse und Beiträge

Wesentlicher Beitrag der vorliegenden Dissertation ist es, den Herausforderungen im Bereich der Entwicklung von Anwendungen im mobilen Cloud Computing zu begegnen, indem ein tieferes Verständnis der existierenden Problemstellungen erarbeitet wird und angemessene Lösungsansätze bereitgestellt werden.

Die erzielten Ergebnisse sollen Entwickler mobiler Anwendungen unterstützen und in die Lage versetzen, Anwendungen zu entwickeln, die verteilt ausgeführt werden und die Veränderungen ihres Nutzungskontextes adaptieren. Dies soll in einer Form ermöglicht werden, bei der die Entwickler mobiler Anwendungen sich nicht mit den Details der Verteilung und Adaption auseinandersetzen müssen. Im Laufe der konzeptionellen Untersuchungen und prototypischen Implementierungen sind im Rahmen dieser Arbeit unterschiedliche Beiträge zur Wissenschaft entstanden, von denen die relevantesten an dieser Stelle hervorgehoben werden sollen:

Als erstes Ergebnis wurden Anwendungsdomänen ausgewählt, die sich für mobiles Cloud Computing eignen, um im Anschluss die bei der Entwicklung und Nutzung auftretenden Problemstellungen zu identifizieren. Diese Problemstellungen wurden im Rahmen eines Anforderungskataloges an eine entsprechende Architektur für mobiles Cloud Computing festgehalten, wobei die generellen Probleme in diesem Zusammenhang aufgezeigt und entsprechende Lösungsalternativen evaluiert wurden [OBL17a].

Basierend auf diesem Anforderungskatalog wurde eine repräsentative Anzahl existierender Ansätze aus verschiedenen Forschungsrichtungen im Umfeld des mobilen Cloud Computings untersucht, um deren Eignung im Hinblick auf die Anwendungsdomänen zu bewerten [OBL17a]. Ebenso wurde eine Basisarchitektur für mobiles Cloud Computing konzipiert, die die kontextadaptive und verteilte Ausführung einer mobilen Anwendung unterstützt [OBL15a]. In diesem Zusammenhang wurde zusätzlich eine Design Guideline für Entwickler bereitgestellt [OBL15c].

Weiterhin wurde eine Anwendungsarchitektur auf Basis des Referenzdesigns für die in [OBL15c] entwickelte Systemunterstützung konzipiert, die eine verteilte und kontextadaptive Ausführung mobiler Anwendungen ermöglicht und dem Entwickler Konzepte bereitstellt, um typische Probleme des mobilen Cloud Computings zu lösen. Dabei wurden für die ermittelten Teilprobleme bestehende Lösungsmöglichkeiten bewertet sowie neue Lösungen für offene Problemstellungen entwickelt [OBL15a, OBL16a]. Hierzu passend wurde ein generischer Prozess zur Kontextadaption entwickelt, der es der konzipierten Systemunterstützung ermöglicht, den zukünftigen Kontext eines mobilen Geräts vorherzusagen und hieraus Adaptionsstrategien zu entwickeln, um das Verhalten einer mobilen Anwendung proaktiv anzupassen [OBL16b, OBL17b].

Diese konzeptionellen Ergebnisse wurden im Rahmen einer prototypischen Implementierung und mithilfe einer Simulation zur Abbildung realistischer Nutzungsmuster und Kontextzustände mobiler Geräte evaluiert [OBL16a, OBL16b]. In diesem Zusammenhang wurde die *CloudAware-Middleware* als Systemunterstützung für mobiles Cloud Computing vorgestellt.

In der vorliegenden Arbeit wurden hierzu bestehende Ergebnisse aus den Forschungsgebieten der verteilten Ausführung von Anwendungen (*Distributed Computing*) und der Kontextsensitivität (*Context Awareness*/*Context-Aware Computing*) zusammengeführt. Hierbei kann festgehalten werden, dass es gelungen ist, die angestrebte Erweiterung bestehender Konzepte für die verteilte Ausführung im mobilen Cloud Computing nutzbar zu machen und den Nutzungskontext der Geräte explizit in die für eine effiziente Umsetzung benötigen Adaptionsprozesse zu integrieren.

1.5. Gang der Untersuchung

In der vorliegenden Arbeit werden zwei miteinander verbundene Themenstellungen im Anwendungsgebiet des mobilen Cloud Computings untersucht. So spielen Architekturen für verteilte Systeme als auch Kontextdatenverarbeitung zwei gleichberechtigte Rollen, allerdings wird trotz eines einheitlichen Vorgehensmodells in den jeweiligen Themenstellungen methodisch unterschiedlich vorgegangen:

Im Bereich der Architekturen für verteilte Systeme berührt diese Arbeit die Teilgebiete der verteilten Dienstausführung, der Middleware und der Prozessmigration. Da in den genannten Teilgebieten zahlreiche Arbeiten zu ähnlichen Problemstellungen vorliegen, wird primär auf eine strukturierte Literaturrecherche zurückgegriffen. Anschließend wird basieren auf bestehenden Ansätzen zunächst eine Evaluation der technischen Restriktionen einerseits und der Anforderungen der Nutzer andererseits durchgeführt, um die praktische Umsetzbarkeit und den Nutzen des in dieser Arbeit entstehenden Konzepts sicherzustellen. Bestehende Arbeiten zur verteilten Dienstausführung werden dabei im Hinblick auf ihre Übertragbarkeit auf das mobile Cloud Computing hin untersucht und bewertet, um im Anschluss ein Konzept für eine Anwendungsarchitektur und Systemunterstützung bereitzustellen.

Im Bereich der Verarbeitung von Kontextdaten wird hingegen ein explorativer und datenanalytischer Ansatz gewählt. Im Gegensatz zur verteilten Dienstausführung existieren in der Literatur nur wenige Konzepte, die solch einen möglichst generischen Prozess zur Kontextadaption angemessen abdecken. Existierende verwandte Ansätze sind zudem nur bedingt übertragbar, da die Natur der zugrunde liegenden Daten oft variiert. Entsprechend wird mithilfe üblicher Prozesse für die Mustererkennung in Datenbanken zunächst eine ausreichend große Basis von Kontextdaten untersucht und im Anschluss ein geeigneter Prozess für die Kontextadaption entworfen, der in der Lage ist, Muster aus den vorhandenen Daten zu extrahieren und hierauf basierend Prognosen eines zukünftigen Kontexts durchzuführen. Ausgehend hiervon berührt diese Arbeit zusätzlich die Forschungsgebiete des Context-Aware Computings [DA99] und des Data Minings.

Für beide Themenstellungen finden, wie in Abbildung 1.3 veranschaulicht wird, hierzu aus methodischer Sicht zunächst deskriptive Verfahren, im Rahmen der Problemdefinition, Anwendung. Diese stellen den ersten Schritt des angewendeten Vorgehensmodells dar. Die Ergebnisse dienen dem darauffolgenden erklärenden Schritt einer Anforderungsanalyse als Grundlage. Auf dieser Basis erfolgt anschließend der konstruktive Teil des Vorgehensmodells: die konzeptionelle Erarbeitung eines Lösungsansatzes. An diesen schließt sich der experimentelle Teil des Vorgehensmodells, in Form der Umsetzung der konzeptionellen Beiträge durch eine entsprechend prototypische Implementierung der entwickelten Architektur, an. Die im Rahmen der Evaluation, dem empirischen Teil dieses Prozesses, gewonnenen Erkenntnisse dienen letztlich der Bewertung der erarbeiteten Lösungsansätze. Praktisch werden die einzelnen Schritte dabei in einem agilen Vorgehensmodell mehrfach durchlaufen und die Ergebnisse aus der Entwicklung verschiedener prototypischer Implementierungen genutzt, um daraus in einer nachfolgenden Iteration Erkenntnisse für die Weiterentwicklung der entwickelten Konzepte abzuleiten.

Der gewählte Ansatz erlaubt es damit, die Erkenntnisse und Beobachtungen vorhergehender Iterationen zu berücksichtigen und ein ganzheitliches Problemverständnis für die verteilte und kontextadaptive Ausführung mobiler Anwendungen im mobilen Cloud Computing zu schaffen. Gleichzeitig zielt er darauf ab die praktische Nutzbarkeit der entwickelten Konzepte und Architekturen zu belegen, um deren Anwendbarkeit im praktischen Einsatz sicherzustellen.

1.6. Aufbau der Arbeit

Die vorliegende Arbeit behandelt die Entwicklung einer kontextadaptiven Anwendungsarchitektur für das mobile Cloud Computing. Wie im vorhergehenden Abschnitt gezeigt, behandelt die Arbeit hierzu zwei wesentliche Teilprobleme: die Entwicklung von Adaptionsstrategien, die den aktuellen und zukünftigen Kontext eines mobilen Gerätes berücksichtigen, und die Entwicklung einer

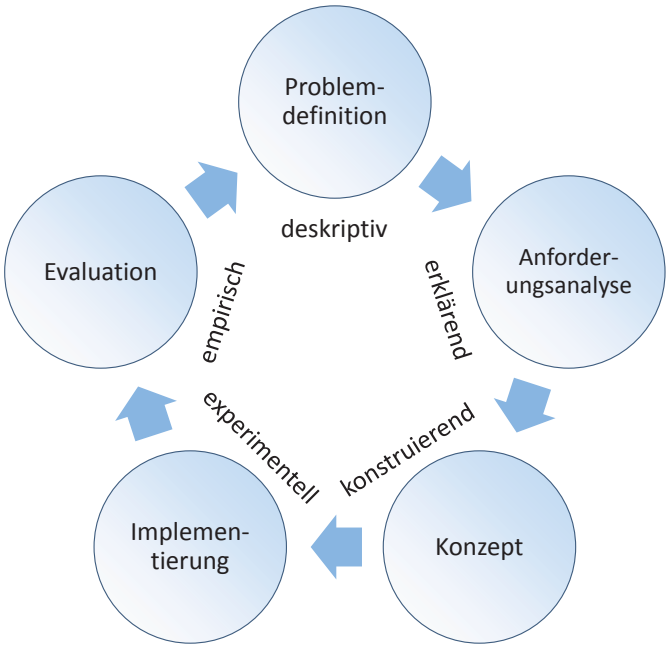

Abbildung 1.3.: Angewendetes Vorgehensmodell

entsprechenden Anwendungsarchitektur, die es ermöglicht, die kontextadaptiven Adaptionsstrategien umzusetzen.

Entsprechend führt das folgende Kapitel 2 zunächst in die Grundlagen des mobilen Cloud Computings ein. Hierfür werden relevante Grundlagen klassischer verteilter Systeme vorgestellt. Im Anschluss wird ein Überblick über die relevanten Grundlagen des Mobile Computings gegeben. Hierfür werden nach einer Vorstellung verschiedener Mobilitätsformen und mobiler Geräte die Einschränkungen dieser gegenüber stationären Geräten auch im Abgleich mit klassischen verteilten Systemen herausgestellt. Anschließend wird der Informationsaustausch, die Interaktion und die Kooperation mit anderen mobilen und stationären Geräten erläutert. Hierfür werden verschiedene Nutzungskonzepte und Sichtweisen anhand verwandter Forschungsgebiete diskutiert. Entsprechend werden die Grundlagen des mobilen Cloud Computings und verwandter Konzepte vorgestellt und abschließend anhand verschiedene Anwendungsfälle weiter veranschaulicht.

Kapitel 3 beleuchtet im Anschluss das Themengebiet der Kontextadaption. Hierzu werden zunächst die unterschiedlichen Definitionen des Kontextbegriffs diskutiert. Anhand von Beispielen werden Kontextdaten klassifiziert und ihre verschiedenen Repräsentationsformen erläutert. Anschließend wird der Prozess der Kontextverarbeitung erläutert. Daraufhin werden geeignete Verfahren ausgewählt und näher betrachtet, die eine generische Verarbeitung von Kontextdaten im Hinblick auf zukünftige Kontextzustände berücksichtigen und weitergehende Auswertungskonzepte bis hin zur Prognose mithilfe von Verfahren aus dem Bereich des maschinellen Lernens erlauben. Abschlie-

ßend werden Adaptionsverfahren vorgestellt, die die Berücksichtigung aktueller und zukünftiger Kontextzustände ermöglichen.

Kapitel 4 diskutiert die verschiedenen Anwendungsbeispiele, die als Referenz für die praktische Umsetzbarkeit des in dieser Arbeit vorgestellten Konzepts herangezogen werden sollen. Hierauf basierend erfolgt eine Anforderungsanalyse, die die allgemeinen Kriterien für Softwarequalität mit den speziellen Anforderungen einer Anwendungsarchitektur für mobiles Cloud Computing kombiniert. Im Anschluss werden existierende Lösungsansätze klassifiziert und eine Auswahl von diesen gegen den so entstandenen Kriterienkatalog evaluiert, wobei eine weitere Klassifizierung dieser Lösungen erfolgt.

Basierend auf den in Kapitel 4 ermittelten Anforderungen beginnt in Kapitel 5 zunächst die Entwicklung einer Basisarchitektur für mobiles Cloud Computing. Anschließend erfolgt die Konzeption einer konkreten kontextadaptiven Anwendungsarchitektur und einer entsprechenden Systemunterstützung. Hierfür werden zunächst Adaptionsbedarfe von Anwendungen im mobilen Cloud Computing erläutert und im Anschluss die bei der Konzeption einer solchen Architektur auftretenden Teilprobleme mit existierenden Lösungsansätzen kontrastiert. Im Anschluss werden eine kontextadaptive Anwendungsarchitektur und eine zugehörige Systemunterstützung vorgestellt, um darauf aufbauend den für die Umsetzbarkeit des Konzepts nötigen Kontextadaptionsmechanismus zu entwickeln.

In Kapitel 6 erfolgt zunächst eine prototypische Implementierung der entwickelten Konzepte. Hierzu werden die für die Systemunterstützung notwendigen Komponenten realisiert und das Konzept des entwickelten Kontextadaptionsmechanismus auf seine Anwendbarkeit hin optimiert. Dieser Entwurf wird anschließend in Kapitel 7 im Rahmen einer qualitativen Evaluation gegen den Kriterienkatalog und im Rahmen einer quantitativen Evaluation im Rahmen zweier Evaluationsprojekte bewertet. Kapitel 8 fasst die Beiträge und Ergebnisse dieser Arbeit zusammen, beleuchtet diese kritisch und gibt einen Ausblick auf zukünftige Entwicklungen. Hierzu wird diskutiert, ob es mithilfe der entwickelten Konzepte gelungen ist, die Zielsetzung der Kooperation ressourcenbeschränkter mobiler Geräte mit ihrer Infrastruktur erfolgreich zu realisieren.

2. Grundlagen des mobilen Cloud Computings

Wie in der Einführung beschrieben, ist die Mobilität in das Zentrum unseres täglichen Denkens und Handelns gerückt. Dieses Kapitel führt entsprechend in die Grundlagen des mobilen Cloud Computings ein. Hierfür werden relevante Grundlagen klassischer verteilter Systeme eingeführt. Zusätzlich wird eine Kombination dieser Konzepte erläutert, die auch als Cloud Computing bekannt ist. Im Anschluss wird eine Einführung in die relevanten Grundlagen des Mobile Computings gegeben; hierfür werden nach einem Überblick über verschiedene Mobilitätsformen und mobile Geräte Einschränkungen dieser mobilen Geräte gegenüber stationären Geräten herausgestellt. Anschließend wird der Informationsaustausch, die Interaktion und die Kooperation mit anderen mobilen und stationären Geräten erläutert. Hierfür werden die Grundlagen des mobilen Cloud Computings, beginnend mit einer Diskussion existierender Definitionen und Begriffserklärungen anhand verschiedener Anwendungsbereiche, näher erläutert. Abschließend werden verwandte Kooperationskonzepte wie das Edge Computing und Fog Computing, vorgestellt.

2.1. Cloud Computing und verteilte Systeme

Der folgende Abschnitt hat zum Ziel, zunächst das Verständnis für den Begriff des Cloud Computings zu präzisieren. Im Anschluss sollen hierauf aufbauende und für diese Arbeit relevante Merkmale des Cloud Computings näher vorgestellt werden.

2.1.1. Definition und Bezug zu verteilten Systemen

> *„Cloud computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services. The services themselves have long been referred to as Software as a Service (SaaS).“ [AFG+10]*

Diese Definition des Begriffs Cloud Computing sowie die damit verbundenen Eigenschaften sind weder in der Wissenschaft noch in der Praxis eindeutig [VRMCL08]. Die initiale, oft zitierte Definition von Armbrust et al. beschreibt Cloud Computing als zwei verschiedene Dinge: einerseits als die Bereitstellung von (Anwendungs-)Diensten über das Internet; andererseits als Hardware-Infrastruktur und Management-Dienst, die diese Anwendungsdienste bereitstellen. Diese Definition berücksichtigt allerdings nur unzureichend, was üblicherweise unter diesem Begriff an Konzepten und Technologien aufgefasst wird. Eine neuere Definition von Buyya et al., die diese berücksichtigt

und die stellvertretend für den Begriff des Cloud Computings in dieser Arbeit herangezogen wird, lautet wie folgt:

> *„Cloud computing is a technological advancement that focuses on the way we design computing systems, develop applications, and leverage existing services for building software. It is based on the concept of dynamic provisioning, which is applied not only to services but also to compute capability, storage, networking, and information technology (IT) infrastructure in general. Resources are made available through the Internet and offered on a pay-per-use basis from cloud computing vendors.“ [BVS13]*

Aufbauend auf dieser Definition sollen in diesem Abschnitt nun die übrigen für diese Arbeit relevanten Aspekte des Cloud Computings beschrieben werden. Betrachtet man diese Definition, stellt sie zunächst die Anwendungsbereiche von Cloud Computing heraus. Diesem Bezug folgend ist Cloud Computing als Entwicklungsparadigma für Systeme sowie Anwendungen zu verstehen und beschreibt die Art, wie bestehende Dienste integriert und verfügbar gemacht werden. Versucht man Cloud Computing nun anhand dieser Definition in bestehende Forschungsgebiete der Informatik einzuordnen, zeigt eine allgemein anerkannte Definition eine hohe Übereinstimmung auf:

> *„A distributed system is a collection of independent computers that appears to its users as a single coherent system.“ [Tv08]*

Dieser Definition nach existieren zwei wichtige Kriterien, die ein verteiltes System ausmachen. Die erste bezieht sich auf die Hardware und besagt, dass es aus mehreren unabhängigen und autonomen Einheiten besteht. Die zweite bezieht sich auf die Software und betrifft die Eigenschaft eines verteilten Systems, nach außen hin und dem Nutzer gegenüber als ein einziges System zu erscheinen. Es liegt damit nahe, Cloud Computing als einen Anwendungsbereich verteilter Systeme aufzufassen (vergleiche [FZRL08, CDK12, BVS13]).

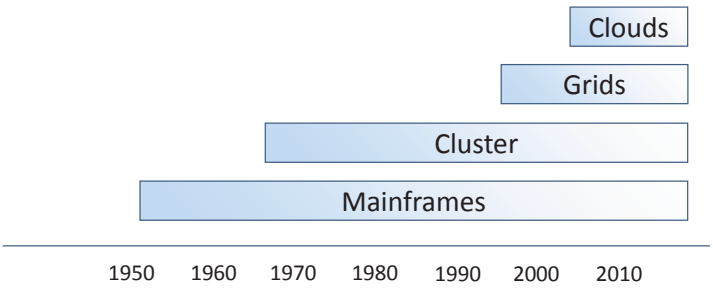

Abbildung 2.1.: Entwicklung des Cloud Computings, nach (BVS13)

Obwohl der Begriff des Cloud Computings in den vergangenen zehn Jahren eine hohe Popularität erreicht hat, liegen seine Ursprünge weiter zurück und sind in der Entwicklung verteilter Systeme zu finden, wie Abbildung 2.1 verdeutlicht. Erste verteilte Systeme wie Cluster, Rechnerverbünde vernetzter

Computer, die als günstige Alternative zu Mainframes entwickelt wurden, erforderten die Entwicklung entsprechender Systemsoftware, die für die Steuerung eines solchen verteilten Systems verantwortlich waren und diese Systeme nach außen hin als ein System erschienen ließen. Das als Weiterentwicklung verstandene *Grid Computing* beschreibt damit den nächsten Schritt, den Zusammenschluss einzelner und heterogener Cluster über das inzwischen ausreichend performante Internet hin zu weltweit abrufbaren Rechen- und Speicherressourcen *[BVS13]*. Als Gründe für diesen Zusammenschluss werden einerseits neue Problemstellungen aufgeführt, die sich nicht mit einem einzelnen Cluster bearbeiten ließen, anderseits wird oft geringe Auslastung einzelner Cluster genannt. Dieser Aspekt, der gleichzeitig in der Definition von Buyya et al. in *[BVS13]* als zweiter wichtiger Aspekt des Cloud Computings genannt wird, soll durch das in Abbildung 2.2 gezeigte Beispiel veranschaulicht werden.

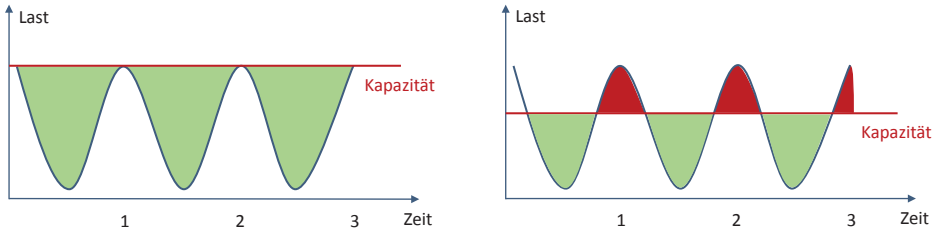

Abbildung 2.2.: Auslastungsmuster eines Rechenzentrums nach (AFG+10)

Das Abdecken sämtlicher Spitzenlasten eines Rechenzentrums (Variante A) führt zu einem Vorhalten von Ressourcen, beispielsweise Rechenleistung, die sowohl aus ökologischer als auch aus ökonomischer Sicht heraus nicht sinnvoll erscheint. Entscheidet man hingegen unter kostenoptimalen Kriterien, bewusst weniger Ressourcen vorzuhalten, führt dies im gezeigten Beispiel zu regelmäßigen Überlastsituationen (Variante B). Der Versandhändler Amazon sah sich diesem Problem ebenfalls ausgesetzt und entschied sich, stets ausreichend Kapazität vorzuhalten, diese aber, wenn sie nicht benötigt wurde, Dritten als Produkt in Form der Amazon Web Services [Ama16] zur Verfügung zu stellen. Dieses Konzept der dynamischen Provisionierung wird in der Definition von Buyya et al. als Pay-per-Use und als dritter wesentlicher Aspekt des Cloud Computings beschrieben und abweichend auch mit „pay as you go“ bezeichnet. Cloud Computing wird damit als Weiterentwicklung des Cluster, vor allem aber des Grid Computings, verstanden, bei dem insbesondere der Aspekt der dynamischen Provisionierung weiterentwickelt und fokussiert wurde, um eine effiziente und kostengünstige Bereitstellung von Ressourcen zu ermöglichen [BVS13].

Diesen Ursprüngen folgend lassen sich viele der Eigenschaften verteilter Systeme direkt oder zumindest indirekt auch auf Cloud-Computing-Systeme übertragen. Dies sind insbesondere ihre Offenheit, Verteilungstransparenz, Skalierbarkeit und Verfügbarkeit sowie ihre Partitionstoleranz, die sich folglich ebenfalls in Cloud-Computing-Systemen finden. Im Unterschied zum Grid

Computing oder klassischen verteilten Systemen bieten sie ihre Dienste üblicherweise einer großen Anzahl von Nutzern an. Der Fokus dieser Arbeit liegt dabei auf der Bereitstellung von Cloud-Ressourcen an mobile Nutzer, weswegen im Folgenden verschiedene Ebenen der Bereitstellung sowie Nutzungsmodelle für die verschiedenen Ressourcentypen vorgestellt werden.

2.1.2. Abstraktionsniveau und Nutzungsmodelle

Im Zusammenhang mit der Verbreitung des Cloud Computings haben sich verschiedene Nutzungsmodelle etabliert, die sich auf das Abstraktionsniveau der bereitgestellten Ressourcen beziehen. Nach einer Definition des NIST aus dem Jahr 2011 [MG10] lassen sich diese zunächst in drei Gruppen einteilen:

Infrastruktur als Dienst (Infrastructure as a Service – IaaS) Im Abstraktionsniveau des IaaS werden Basisressourcen wie Rechen-, Speicher- oder Netzwerkressourcen bereitgestellt, die dem Nutzer normalerweise virtualisiert zugänglich gemacht werden und ihm ermöglichen, beliebige Software auszuführen. Die Buchung und Steuerung der Ressourcen erfolgt dabei üblicherweise mithilfe webbasierter Management-Schnittstellen [BVS13]. Ein Beispiel sind die zuvor genannten Amazon Web Services [Ama16].

Plattform als Dienst: (Platform as a Service – PaaS) Im Abstraktionsniveau des PaaS nutzt ein Kunde eine vorhandene Ausführungsumgebung, um eigenen oder bereits existierenden Programmcode, der mit der Plattform des Cloud-Dienstleisters kompatibel ist, auszuführen. Die Steuerung und Konfiguration erfolgt dabei auf der Ebene der veröffentlichten Anwendung oder der Ausführungsumgebung selber. Auf mögliche darunterliegende IaaS-Dienste besteht allerdings kein Zugriff. Ein Beispiel für dieses Modell ist die Google AppEngine [Goo16], die es erlaubt, automatisch skalierende Webanwendungen zu entwickeln, die erst beim Zugriff kostenpflichtig abgerechnet werden.

Software als Dienst: (Software as a Service - SaaS) Das nächsthöhere Abstraktionsniveau ist die Bereitstellung von Software, die als direkt durch den Endanwender[1] nutzbar charakterisiert ist. Die Konfiguration erstreckt sich ebenfalls nur auf diese Ebene. Auch in diesem Modell besteht üblicherweise kein Zugriff auf gegebenenfalls darunterliegende PaaS- oder IaaS-Schichten. Ein Beispiel ist Microsofts webbasiertes Office-Paket [Mic16a].

Anzumerken ist weiterhin, dass für die Bereitstellung eines SaaS-Dienstes verschiedene PaaS und IaaS als Basis dienen können, dies jedoch keine Voraussetzung darstellt. Abbildung 2.3 zeigt diese verschiedenen Ebenen.

[1] Die Begriffe Endanwender und Nutzer werden synonym verwendet. Der Begriff Anwender bezeichnet hingegen den Software-Entwickler

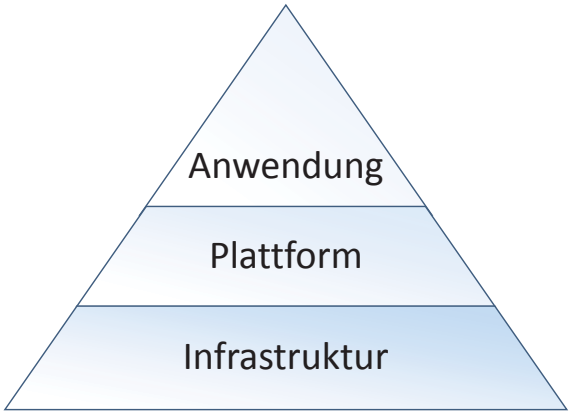

Abbildung 2.3.: Nutzungsmodelle des Cloud Computings

Ein zweites relevantes Differenzierungsmerkmal neben dem Abstraktionsniveau betrifft den Zugriff auf Cloud-Dienste. Laut NIST können dabei drei verschiedene Modelle unterschieden werden, die nach [BVS13] wie folgt definiert sind:

Öffentliche Cloud (Public Cloud) Ressourcen sind generell frei zugänglich, was der Fall bei den zuvor genannten Beispielen der Amazon Web Services oder Microsofts webbasiertem Office-Paket ist.

Private Cloud (Private Cloud) Sie stellen organisationsinterne Clouds dar und stehen üblicherweise nur einem eingeschränkten Nutzerkreis zur Verfügung. Vorteile sind in diesem Fall die Skalierbarkeit und die bessere Ressourcenauslastung für organisationsinterne Dienste.

Hybride Clouds (Hybrid Clouds) Sie bezeichnen die Mischform der zuvor genannten Modelle, bei denen bestehende kritische und schützenswerte Workflows unternehmensintern und übrige Workflows in öffentlichen Clouds ausgeführt werden.

Die seit wenigen Jahren verfügbaren Cloud-Dienste haben damit erstmalig die ortsunabhängige Nutzung von Rechen- und Speicherdiensten zu Preisen ermöglicht, die zuvor nicht realisierbar waren. Inwieweit mobile Geräte von diesen Diensten profitieren können, wird anschließend an die Einführung in das Mobile Computing beschrieben.

2.2. Mobile Computing

Begünstigt durch verschiedene Entwicklungen in der Informations- und Kommunikationstechnologie ist die Mobilität in das Zentrum unseres täglichen Denkens und Handelns gerückt. Die ortsunabhängige Erreichbarkeit und die Möglichkeit zum ständigen Informationsaustausch zwischen Nutzern mobiler Geräte, aber auch den Geräten untereinander ist alltäglich geworden. Bereits 25 Jahre zuvor wurden diese Entwicklungen von Mark Weiser, dem späteren Chief Technologist des Xerox PARC, vorhergesagt:

> *„Ubiquitous computing has as its goal the enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user." [Wei93]*

Dieser eher nutzer- und nutzungsorientierte Ansatz geht davon aus, dass Computer zukünftig nicht mehr sichtbar, sondern unsichtbar in unseren Alltag integriert sind. Diese Entwicklung beschreibt Mark Weiser in drei Phasen, von denen die erste die Zeit der Mainframes darstellt, also große, teure und gemeinsam genutzte Geräte. Die zweite Phase beschreibt den klassischen Personal Computer, der vornehmlich von einer Person exklusiv genutzt wird. Die dritte Phase stellt ein weltweit verbundenes System intelligenter Gegenstände dar, bei denen die eigentlichen Computer für ihre verschiedenen Nutzer unsichtbar werden.

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

Seiner Vorstellung nach wird der Mensch zukünftig von einer Vielzahl intelligenter Gegenstände umgeben, die miteinander (drahtlos) verbunden sind und uns den ständigen Zugriff auf Dienste und Informationen ermöglichen. Er beschreibt dabei das Verschwinden von Technik in der heutigen Form hin zu einer Durchdringung alltäglicher Gegenstände mit Informationstechnologie, die für ihre Nutzer nicht explizit als solche wahrgenommen werden. Laut Mark Weiser soll dieser Übergang zwischen 2005 und 2020 stattfinden.

Dieses Szenario ist von einer fortschreitenden Miniaturisierung von Computern, aber auch von verschiedensten Trends der Gebiete Mobilkommunikation, den Entwicklungen kontextadaptiver Systeme und insbesondere auch einem zukünftig veränderten Nutzungsverhalten abhängig. Der folgende Abschnitt führt in die mit dem Mobile Computing verbundenen Grundlagen und Forschungsgebiete einführen, die für diese Arbeit relevant sind.

2.2.1. Einführung und Begriffsklärung

Das Forschungsgebiet des *Mobile Computings* beschäftigt sich nach [Rot02] sowohl mit Fragen der Kommunikation von mobilen Benutzern (Mobilkommunikation) als auch mit mobilen Endgeräten und den zugehörigen Anwendungen.

Es stellt somit einen unscharfen Oberbegriff für die Datenverarbeitung auf einem tragbaren Computer dar. International sind verschiedene weitere Begriffe gebräuchlich, die im Folgenden zunächst vorgestellt und im Anschluss diskutiert werden sollen.

Pervasive Computing Neben der stark auf die Nutzung ausgerichteten Vision des Ubiquitous Computings ist die verwandte Definition des *Pervasive Computings* zukunftsnäher und stärker technisch geprägt. So geht es hier primär um die allgegenwärtige Informationsverarbeitung, wie sie in [HMNS03] beschrieben wird:

> *„Everywhere at anytime - This common slogan expresses in a nutshell the goal of Pervasive or Ubiquitous Computing. Both terms describe the visible and mobile front-end for the next generation of integrated IT applications. Pervasive Computing includes flexible and mobile devices like personal digital assistants, mobile phones, pagers, hand-held organizers and home entertainment systems, which will access or provide a rich diversity of applications.“ [HMNS03]*

Pervasive Computing wird, nach einer aktuelleren Definition, auch als Oberbegriff für Technologien und Konzepte von Anwendungssystemen verstanden, die mobile und heterogene Endgeräte nutzen, die untereinander vernetzt sind und die das Internet als Kommunikations- und Dienstplattform nutzen [Fuc09].

Nomadic Computing Im Vergleich zum Mobile Computing wird im *Nomadic Computing* ein noch weitaus stärkerer Fokus auf die Mobilität des Nutzers gelegt. Nach [Fuc09] wird hier die Entwicklung von Systemen und Infrastrukturen fokussiert, die die Nutzer in und während ihrer Mobilität unterstützen. Nach [Rot02] liegt der Fokus hier auf der Mobilität des Nutzers, wohingegen im Mobile Computing auch Probleme behandelt werden, die sich am Zielort einer Reise ergeben. Nach [Fuc09] gilt es für eine erfolgreiche Umsetzung eines Nomadic Computings vor allem drei Dinge zu vereinen: Erstens die Mobilität der Teilnehmer, wofür dem System bekannt sein muss, wo sich seine einzelnen Nutzer befinden. Zweitens muss es in der Lage sein, die einzelnen Nutzer zu unterscheiden und zu identifizieren. Drittens muss das System die jeweiligen Bedürfnisse der einzelnen Nutzer kennen und wissen, wie sie befriedigt werden können. Zusammenfassend wird eine erfolgreiche Umsetzung wie folgt definiert:

> *„Das Angebot an Diensten, die Qualität der Dienste und die Informationsinhalte, die einem Nutzer angeboten werden, sind abhängig vom Ort, an dem sich der Nutzer aufhält, und vom Kontext, in dem die Anwendung stattfindet, und müssen daran angepasst werden.“ [Rot02]*

Anzumerken ist, dass hier erstmals Bezug auf den Ausführungskontext genommen und die Relevanz der Anpassung herausgestellt wird, ein Aspekt auf den im Kapitel 3 noch explizit eingegangen wird.

Wearable Computing Ein weiter Begriff im Umfeld des Mobile Computings ist das sogenannte *Wearable Computing*. Nach [Fuc09] werden unter diesem Schlagwort alle Aktivitäten zusammengefasst, die sich mit dem Tragen von Computern beschäftigen. Wesentliche Merkmale dieser Geräte sind laut [Fuc09] beschrieben als:

- Sie sind portabel, während sie betrieben werden. Das heißt, sie sind explizit dafür entwickelt, nicht nur portabel zu sein, sondern während des Tragens auch benutzt zu werden.
- Sie sollen ohne den Gebrauch der Hände bedient werden, um ihren Nutzer nicht zu behindern.
- Sie sind in der Lage, den Anwender auf sich aufmerksam zu machen, zum Beispiel bei einem Mobiltelefon durch einen Klingelton oder durch ein haptisches Feedback wie eine Vibration.
- Sie sind ständig eingeschaltet, da sie als Kleidungsstück oder Accessoire zu einem ständigen Begleiter geworden sind.
- Sie können ihre Umgebung wahrnehmen, um ihren Nutzer zu unterstützen, damit diese Geräte zu intelligenten Gegenständen im Sinne des Ubiquitous Computings werden.

Das Wearable Computing kann damit als eine spezielle Form des Pervasive Computings verstanden werden, die sich mit der Durchdringung der nächsten Umgebung des Nutzers beschäftigt [Fuc09]. Abbildung 2.4 zeigt eine exemplarische Auswahl von Wearables.

Abbildung 2.4.: Exemplarische Darstellung von Wearables

Augmented Reality Dienen diese Geräte wie die Datenbrille Google Glass zusätzlich der Anreicherung der physischen Realität mit zusätzlichen Informationen, so sind dies Beispiele für *Augmented Reality*, ein weiteres Teilgebiet des Mobile Computings. Dieser Begriff beschreibt nach [Rot02] die Anreicherung der physischen Welt um zusätzliche Informationen. Ein Beispiel für solch eine Anwendung ist in Abbildung 2.5 gezeigt, wo ein mobiles Gerät dazu genutzt wird, ein Möbelstück in das von der Kamera aufgenommene Bild des Raumes zu projizieren.

Abbildung 2.5.: Augmented Reality-Anwendung (Wik16)

Die initiale Definition des Mobile Computings von Jörg Roth bezieht sich lediglich auf Fragestellungen der Kommunikation, wohingegen sich in [Fuc09] eine weitere Definition findet. Diese versucht, die genannten Forschungsgebiete und Trends in einer gemeinsamen Definition zu erfassen, die in dieser Arbeit als stellvertretend für den Begriff des Mobile Computings herangezogen wird:

> *„Das Ziel des Mobile Computing ist es, den Benutzer und dessen Anwendungen mit effektiven rechnerunterstützten Konzepten, Verfahren und Lösungen zu versorgen, die es ihm ermöglichen, in einem heterogenen Umfeld mit stets unsicherer Verbindungslage (private) Daten und Informationen zu lesen und zu bearbeiten, und dies unabhängig von Ort und Zeit.“ [Fuc09]*

In der vorliegenden Arbeit liegt zudem der Schwerpunkt nur indirekt auf der Hardware mobiler Geräte und den von ihnen verwendeten Standards zur Mobilkommunikation. Der Fokus liegt dagegen auf ihren (mobilen) Anwendungen, die kontextbasiert mit ihrer Infrastruktur interagieren. Um diese Interaktion erfolgreich zu realisieren, ist die Berücksichtigung unterschiedlicher Eigenschaften und Restriktionen mobiler Geräte erforderlich, welche in den nun folgenden Abschnitten vorgestellt werden sollen.

2.2.2. Klassifikation von Mobilitätsformen

Der Begriff der Mobilität im Kontext des Mobile Computings ist mehrdeutig, entsprechend soll diese Mobilität im Weiteren sukzessive dargestellt werden. So wird einerseits in der Definition von Roth in [Rot02] der Bezug zur Mobilkommunikation als Kommunikation von mobilen Nutzern hergestellt. Hiervon abzugrenzen ist der Begriff der drahtlosen Kommunikation (*wireless communication*), die die Art der Anbindung mobiler Geräte an die übrige Infrastruktur beschreibt. Entsprechend ergeben sich die in Tabelle 2.1 gezeigten Kombinationen von Mobilitätsaspekten. Die Mobilität im Mobile Computing wird dabei überwiegend, und auch in dieser Arbeit, im Sinne der Kombination aus mobiler und drahtloser Kommunikation verstanden.

	Nichtmobile Kommunikation	Mobile Kommunikation
Drahtgebunde Kommunikation	Workstation in einer Büroumgebung	Notebook im Hotelzimmer, angebunden über Modem
Drahtlose Kommunikation	Workstation im drahtlosen lokalen Netz	Notebook über Mobiltelefon drahtlos angebunden

Tabelle 2.1.: Mobile und drahtlose Kommunikation nach (Rot02)

Diese Art der Mobilität kann weiter unterteilt werden, nach [Pan04] wird hier weiter zwischen physischer und logischer Mobilität unterschieden. Physische Mobilität beschreibt hierzu die Bewegung physischer Gegenstände in einem physischen Raum; logische Mobilität hingegen beschreibt die Bewegung logischer Einheiten, beispielsweise einer mobilen Anwendung, innerhalb eines logischen Raums wie den verschiedenen Computern eines Netzwerks.

Die physische Mobilität kann dabei weiter in drei Kategorien eingeteilt werden: Endgerätemobilität (Terminal Mobility), Benutzermobilität (Personal Mobility) und Dienstmobilität (Service Mobility), die in [Pan04] wie folgt beschrieben werden:

Endgerätemobilität Die Endgerätemobilität bezeichnet die physische Mobilität von mobilen Geräten. Sie ist gegeben, wenn mobile Geräte unabhängig von ihrem Aufenthaltsort vernetzt und damit nutzbar bleiben. Sie erfordert im Regelfall das Vorhandensein drahtloser Netzwerkschnittstellen im mobilen Gerät. Ein Beispiel für Endgerätemobilität sind Mobiltelefone, die normalerweise ständig im Mobilfunknetz eingebucht sind und es ihrem Nutzer erlauben, unterwegs Mobilkommunikation zu nutzen.

Benutzermobilität Im Falle der Benutzermobilität rückt der Fokus auf die Mobilität des Nutzers selbst. Sie ist gegeben, wenn ein Nutzer ortsabhängig verschiedene Endgeräte nutzen kann, um auf Dienste zuzugreifen. Diese Art der Mobilität erfordert üblicherweise eine Authentifizierung gegenüber dem Dienst oder Netzwerk. Ein Beispiel ist der Zugriff auf das Mo-

bilfunknetz über verschiedene Mobiltelefone und die Authentifizierung mithilfe der persönlichen SIM-Karte.

Dienstmobilität Diese Art der Mobilität fokussiert die Mobilität des Dienstes selber. Hier kann ein Benutzer ortsunabhängig auf ein und denselben Dienst zugreifen. Ein Beispiel hierfür ist ein E-Mail-Client der auf verschiedenen, auch mobilen Endgeräten zur Verfügung steht.

Ebenso kann die logische Mobilität weitergehend unterschieden werden. In [FHLM01] und [SS02] wird diese in Sitzungsmobilität (Session Mobility), Netzwerkmobilität (Network Mobility) und Komponentenmobilität (Software Component Mobility) differenziert und wie folgt beschrieben:

Sitzungsmobilität Sie bezeichnet den unterbrechungsfreien Zugriff auf eine aktive Sitzung des Nutzers von verschiedenen Geräten. Sitzungen können hierbei pausiert und wieder aufgenommen werden, um die (physische) Mobilität ihres Nutzers zu unterstützen. Teilweise ist hierbei eine Anpassung der in der Sitzung genutzten Dienste an die speziellen Eigenschaften der jeweiligen Endgeräte erforderlich [FHLM01]. Ein Beispiel für Sitzungsmobilität ist die Nutzung von cloud-basierten Diensten zur Bearbeitung von Dokumenten im Sinne eines „Software as a Service“.

Netzwerkmobilität Als Netzwerkmobilität wird die Fähigkeit eines drahtlosen Netzwerkes beschrieben, sich selbst physikalisch zu bewegen, teilweise gänzlich ohne eine ortsgebundene Infrastruktur zur Mobilkommunikation [FHLM01]. Diese als *(mobiles) Ad-hoc-Netzwerk* beschriebene Topologie wird im Laufe dieser Arbeit noch näher erläutert. Beispiele hierfür finden sich vor allem in der direkten Kommunikation zwischen mobilen Geräten. Wenn diese Geräte beispielsweise in Fahrzeuge integriert sind, bilden sie sogenannte *Vehicular-ad-hoc-Netzwerke* (VANET), um Informationen, beispielsweise zu Gefahrensituationen wie Aquaplaning, an entgegenkommende Fahrzeuge weitergeben können.

Komponentenmobilität Sie beschreibt die Mobilität einer Softwarekomponente, die von einem Computer zu einem anderen verschoben werden kann. Nach [FHLM01] kann dies beispielsweise ein Datenbankmanagementsystem inklusive der Daten umfassen. Angelehnt an diese Definition wird in [FPV98] weiter zwischen einer starken und einer schwachen Mobilität von Programmcode unterschieden. Die Definition der Komponentenmobilität in [FHLM01] entspricht dabei der schwachen Mobilität, bei der lediglich Programmcode und Daten der Anwendung verschoben werden. Als Erweiterung dieser Definition wird bei der starken Mobilität zusätzlich der Ausführungszustand (execution state) zwischen zwei Systemen migriert. Üblicherweise geschieht dies mithilfe eines serialisierten (Speicher-)Abbilds des jeweiligen Betriebssystemprozesses [KCRG15, LCTB+06].

In der vorliegenden Arbeit liegt der Fokus in Bezug auf die physische Mobilität im Wesentlichen bei der Endgerätemobilität, bezieht jedoch Aspekte der Dienstmobilität mit ein. Im Hinblick auf die logische Mobilität fokussiert diese Arbeit den Aspekt der Komponentenmobilität und berücksichtigt zusätzlich Aspekte der Sitzungs- und Netzwerkmobilität.

2.2.3. Kategorien mobiler Geräte

Wie im vorherigen Abschnitt gezeigt wurde, unterscheiden sich mobile Geräte gegenüber stationären Geräten insbesondere durch ihre Mobilität. Diese unterschiedlichen Mobilitätsanforderungen haben verschiedene Klassen mobiler Geräte hervorgebracht, die im Folgenden vorgestellt werden sollen. Im Anschluss werden die speziellen Eigenschaften und Restriktionen im Abgleich mit stationären Systemen diskutiert.

Mobile Geräte werden hierzu primär anhand ihrer Leistungsfähigkeit unterschieden. In [Fuc09] und [Rot02] werden diese, jeweils leicht abweichend und für die folgende Aufstellung zusammengefasst, in die Kategorien der Laptops und Notebooks, der Handhelds, der Wearables und in die Kategorie der eingebetteten Systeme aufgeteilt, die im Folgenden näher beschrieben werden:

Laptops und Notebooks Mobile Geräte dieser Kategorie ähneln in Bezug auf Leistungsfähigkeit, Ressourcen und Eigenschaften stark stationären Computern. Abgesehen von der mit einem Gewicht von ein bis drei Kilogramm einhergehenden Portabilität unterscheiden sie sich im Wesentlichen durch einen begrenzten Energievorrat von stationären Geräten. Sie zeichnen sich stationären Geräten gegenüber durch zusätzliche drahtlose Kommunikationsschnittstellen aus. Auf dieser Kategorie von Geräten werden im Allgemeinen dieselben Betriebssysteme und Anwendungen eingesetzt wie auf stationären Geräten.

Handhelds (Tablets und Smartphones) Weitaus kleiner als Laptops und Smartphones sind sogenannte Handhelds. Die zu Beginn der 1990er-Jahre ursprünglich als kompakter und tragbarer Computer für die Kalender-, Adress- und Aufgabenverwaltung konzipierten Geräte wurden bereits vor einigen Jahren nahezu vollständig durch Smartphones verdrängt. Smartphones haben sich dabei für eine Vielzahl von Menschen zu selbstverständlichen Alltagsgeständen entwickelt, die weit mehr Aufgaben übernehmen, als an Termine und Telefonnummern zu erinnern. Stattdessen unterstützen sie uns bei einer Vielzahl unserer täglichen Aufgaben (unter anderem Kommunikation) oder übernehmen diese gänzlich für ihre Nutzer wie bei der Routenfindung oder der Suche nach Informationen.

Diese Kategorien von Geräten bieten oft spezielle, auf die Mobilitätsbedürfnisse zugeschnittene, Eingabemöglichkeiten wie Trackpoints, berührungssensitive Flächen (Touchpad) oder werden im Fall von Tablets mit einem speziellen Stift gesteuert [Rot02]. Tablets werden dabei in [Rot02]

als Notebooks ohne oder mit einer wegklappbaren Tastatur definiert, bei denen Texteingaben häufig über eine Schrifterkennung vorgenommen werden.

Smartphones sind in [Rot02] noch als Handhelds definiert, die man in einer Hand halten kann. In der aktuelleren Definition aus [Fuc09] werden sie als Kombination zwischen Organizer und Mobiltelefon beschrieben. In dieser Arbeit werden Smartphones hingegen als verkleinerte Variante von Tablets und mit diesen zusammen betrachtet, da sich beide Gerätetypen im Hinblick auf ihre Leistungsfähigkeit und Funktionen stark ähneln.

Wearables Werden mobile Geräte direkt am Körper getragen, wie Smartwatches, Datenbrillen oder Google Glass, die der Gerätekategorie peripherer Head-Mounted-Displays zugeordnet werden [MHAU15], oder sind diese in die Kleidung eines Nutzers integriert, so ist die Rede von sogenannten Wearables [Fuc09]. Gegenüber Handhelds ist diese Gerätekategorie zusätzlich in ihren Ressourcen und damit in ihrer Leistungsfähigkeit und ihren Funktionen beschränkt.

Eingebettete Systeme Obwohl sie in [Fuc09] im Zusammenhang mit ihrer Leistungsfähigkeit zusammen mit leistungsbeschränkten Sensorknoten genannt werden, werden eingebettete Systeme in [BC12] als Geräte durchaus verschiedener Leistungsklassen beschrieben. In [Rot02] findet sich keine Entsprechung zu diesen Geräten.

> *„Embedded systems are computers with constraints. [...] In particular, embedded computer systems have distinctive constraints with respect to intended applications and form factors, power, system resources and features, and assumptions about user behavior. [...] Unlike general-purpose machines, embedded systems are typically designed for one target application, or class of target applications.“ [BC12]*

Dieser Definition folgend sind diese Geräte auf einen bestimmten Zweck hin entwickelt, was sich oft in einer spezifischen Hardware und Software dieser Geräte zeigt, die oft auf genau einen Zweck hin optimiert ist. Sie sind daher meist nicht in der Lage, Standardsoftware auszuführen.

Im Rahmen der vorliegenden Arbeit wird insbesondere die Kategorie der Handhelds betrachtet, weswegen diese im folgenden Abschnitt im Abgleich mit stationären Geräten in Bezug auf ihre Eigenschaften und Funktionalitäten näher betrachtet werden sollen.

2.2.4. Eigenschaften und Einschränkungen mobiler Systeme

Dieser Definition von Handhelds als mobilen Geräte folgend sollen die Unterschiede zwischen typischen mobilen und stationären Geräten aus dem Jahr 2013 aufgezeigt werden. Anhand der Auflistung werden in Tabelle 2.2 dazu der

aktuelle Stand und die Entwicklung mobiler Geräte sowie ihre spezifischen Eigenschaften und Restriktionen aufgezeigt. Obwohl die verfügbaren Ressourcen bei der Nutzung von Cloud-Diensten prinzipiell deutlich höher liegen, wurde, um eine möglichst hohe Vergleichbarkeit für die Ausführung typischer Anwendungen (für mobile Geräte) sicherzustellen, hier exemplarisch eine einzelne Server-Instanz ausgewählt.

	Jahr	Modell	CPU	Speicher	RAM	Display	Datenüber-tragung	Max. Übertra-gungsrate	Akku
mobil	2014	Samsung Galaxy S5	Krait 400 (2,5 GHz, 4 Kerne)	32 GB	2 GB	5" 1920 x 1080	LTE	150 Mbit	2600 mAh
stationär	2014	Dell Precision T7600 Workstation	Intel® Xeon® E5-2630 Prozessor (Six Core, 2,3 GHz)	1 TB	16 GB	24" 1920 x 1080	Ethernet	1 Gbit/s	-
Cloud	2014	Amazon EC2 cr1.8xlarge	2 x Intel Xeon E5-2670, (Eight-Core)	unlimitiert	244 GB	-	Ethernet	10 Gbit/s	-

Tabelle 2.2.: Ausgewählte mobile und stationäre Geräte (Ama13, Wik13, Del16)

Rechenleistung und Speicherkapazität In Bezug auf die Rechenleistung liegen die Geräteklassen noch deutlich auseinander, es gilt im Hinblick auf die abrufbare Leistung mobiler Geräte dabei zusätzlich, deren begrenzten Energievorrat zu berücksichtigen, der die tatsächlich zur Verfügung stehende Rechenleistung weitaus stärker limitiert. Wie noch gezeigt wird, kann dies einen der wesentlichen Gründe für die verteilte Ausführung mobiler Anwendungen im Rahmen des mobilen Cloud Computings darstellen.

Der Arbeitsspeicher mobiler Geräte umfasst aktuell durchschnittlich zwei Gigabyte. Hiervon wird jedoch ein nicht unwesentlicher Teil durch das mobile Betriebssystem belegt, was den für mobile Anwendungen zur Verfügung stehenden Arbeitsspeicher weiter beschränkt. Der zur Verfügung stehende nichtflüchtige Speicher mobiler Geräte umfasst hingegen bis zu 32 Gigabyte, hier ist allerdings zu berücksichtigen, dass dieses Merkmal hauptsächlich im Sinne einer vertikalen Preis- und Produktdifferenzierung Verwendung findet und nur bedingt die technischen Möglichkeiten widerspiegelt. Im Vergleich hierzu finden sich bei den stationären Geräten weitaus höhere Speicherkapazitäten, welche zusätzlich erweiterbar sind und damit nicht die Obergrenze darstellen.

Akkuleistung Die Energiedichte der in mobilen Geräten verwendeten Batterien steigt jährlich um fünf bis zehn Prozent [ZN10, Rob13]. Zusammen mit einer gesteigerten Effizienz der jeweils verwendeten übrigen Komponenten geht jedoch nicht die zu erwartende Verbesserung der Laufzeit mobiler Geräte im Batteriebetrieb einher. Ein wesentlicher Grund hierfür ist der in gleichem Maße zunehmende Funktionsumfang mobiler Ge-

räte, beispielsweise in Form größerer Bildschirme oder zusätzlicher Anwendungen [FK13]. Abhängig von ihrer Nutzung liegt die Laufzeit mobiler Geräte im Batteriebetrieb damit zwischen einigen Stunden bis hin zu ungefähr einer Woche. Ein möglichst sparsamer Umgang mit dieser Ressource ist damit entscheidend für die Verfügbarkeit und Nutzbarkeit dieser mobilen Geräte. In diesem Zusammenhang ist der Energievorrat stationärer Geräte als unbegrenzt anzusehen.

Anzeige & Benutzungsschnittstelle Obwohl die Auflösung der mobilen Geräte in bestimmten Fällen bereits den stationären Geräten entspricht, bestehen aufgrund der geringeren Bildschirmdiagonale Einschränkungen in den Anzeige- und Bedienmöglichkeiten. Zu erwähnen ist, dass die Gerätekategorie der Tablets mit ihren Bildschirmdiagonalen bis zu zehn Zoll hier andere Maßstäbe setzt. In Bezug auf die Darstellung aufwendiger zu berechnender Bildschirminhalte stehen mobile Geräten durch die begrenzte Rechenleistung in ihrer Leistung hinter derjenigen der stationären Geräte. In vielen Fällen greifen die mobilen Geräte zur Darstellung dieser Inhalte auf die Unterstützung spezieller Grafikprozessoren, analog der Architektur stationärer Geräte, zurück [BC12].

Sensoren Heutige mobile Geräte, insbesondere Smartphones, besitzen im Vergleich zu stationären Geräten eine Vielzahl von Sensoren, die den Nutzungskontext der Geräte und ihrer Nutzer erkennen. Die so erhobenen Daten werden genutzt, um die Qualität und Benutzbarkeit mobiler Anwendungen zu verbessern oder neue Klassen von Anwendungen bereitzustellen. Ein typisches mobiles Gerät besitzt unter anderem die folgenden physischen Sensoren: eine hochauflösende Kamera, einen Beschleunigungsmesser, ein Barometer, ein Gyroskop, ein Hygrometer, einen Kompass, einen GPS-Empfänger, einen Annäherungssensor sowie einen Helligkeitssensor und ein Thermometer. Mit Bezug auf das mobile Cloud Computing können diese Sensoren genutzt werden, um den aktuellen Nutzungskontext des mobilen Gerätes zu berücksichtigen und gegebenenfalls Adaptionsprozesse auszulösen.

Betriebssysteme und Ausführungsumgebung Aktuelle mobile Geräte der Kategorien Tablet und Smartphone verwenden üblicherweise Prozessoren mit einer x86- oder einer ARM-Mikroarchitektur, um dem Nutzer ein modernes Multitasking-Betriebssystem bereitzustellen [BC12]. Es existieren aktuell zwei verschiedene Mobilplattformen mit relevantem Marktanteil: iOS der Firma Apple mit einem Anteil von 12,9 Prozent der in Europa verkauften Geräte im zweiten Quartal 2016 und die von der Open Handset Alliance entwickelte Mobilplattform Android mit einem entsprechenden Marktanteil von 86,2 Prozent [Gar16].

Hinsichtlich der Erweiterbarkeit kann hier nur die Mobilplattform Android als offene Plattform bezeichnet werden. So erlaubt die iOS-Plattform keine Modifikationen des Betriebssystems und beschränkt die

Ausführung von eigenen mobilen Anwendungen auf solche, die zuvor durch den Hersteller der Mobilplattform freigegeben und digital signiert wurden. Diese Restriktionen ergeben sich in weiten Teilen aus den für mobile Geräte nötigen erhöhten Sicherheitsanforderungen zum Schutz der Daten und der Privatsphäre des Nutzers. Im Gegensatz zur breiten Verfügbarkeit und flexiblen Einsetzbarkeit verschiedener Betriebssysteme für stationäre Geräte beschränkt sich die Auswahl bei mobilen Geräten meist auf das vom Hersteller des Gerätes angebotenen Betriebssystem. Prinzipiell ist auch der Einsatz alternativer offener Betriebssysteme wie Linux auf einigen mobilen Geräten möglich, die praktische Nutzbarkeit dieser Lösung auf einer breiten Anzahl mobiler Geräte ist jedoch aktuell nicht sichergestellt. Selbst größere Projekte wie Ubuntu 4 Android [Can16b, Can16a] unterstützen nur eine relativ geringe Anzahl der am Markt verfügbaren Geräte.

Sicherheit Die hohe Mobilität und geringe Größe mobiler Geräte bringen zunächst das Risiko des Verlusts mit sich. Damit besteht die Möglichkeit, dass unberechtigte Dritte Zugriff auf schützenswerte Informationen erhalten [AGRS05].

Konnektivität Gegenüber stationären Geräten, die üblicherweise eine stabile Anbindung zu den übrigen Computern eines Netzwerks besitzen, sind mobile Geräte oft auf drahtlose Kommunikation angewiesen, um mit anderen Geräten zu kommunizieren und interagieren. Diese Verbindung leidet üblicherweise unter einer stark und schnell wechselnden Verbindungsqualität und spontanen Verbindungsabbrüchen, was es erforderlich macht, die Entwicklung mobiler Anwendungen auf diese speziellen Restriktionen hin auszurichten.

Zusammenfassend kann festgehalten werden, dass mobile Geräte stationären Geräten gegenüber insbesondere über eine begrenzte Rechen- und Speicherkapazität, aber vor allem über einen begrenzten Energievorrat verfügen. Zusätzlich ist ihre Konnektivität den häufigen Änderungen ihres Nutzungskontexts unterworfen; entsprechend sollen im folgenden Abschnitt zunächst die für die Mobilkommunikation genutzten Architekturen und im Anschluss die konkreten Kommunikationsstandards näher vorgestellt werden.

2.2.5. Architekturen für Mobilkommunikation

Nach [Fuc09] lassen sich im Bereich der Mobilkommunikation grundlegend zwei verschiedene Arten der Vernetzung unterscheiden: Infrastruktur- und Ad-hoc-Netze. Der wohl größte Unterschied dieser beiden Ansätze betrifft die Verbindung der einzelnen Teilnehmer untereinander und damit die Topologie des Netzwerks. Nach [Fuc09] sind diese zwei grundlegend verschiedenen Architekturen wie folgt charakterisiert.

Infrastruktur-Vernetzung Diese Art der Vernetzung ist dadurch charakterisiert, dass ein kabelgebundenes und zuverlässiges Infrastruktur-

Netzwerk (Core-Network) vorhanden ist, das für das Routing von Datenpaketen verantwortlich ist. An dieses wiederum ist ein Zugangsnetzwerk (Access-Network) angebunden, welches für die Anbindung mobiler Geräte verantwortlich ist, wie in Abbildung 2.6 veranschaulicht wird.

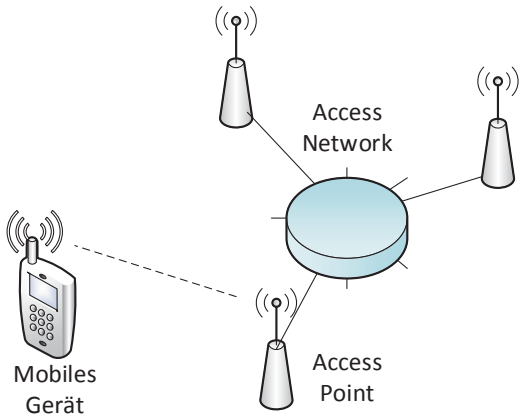

Abbildung 2.6.: Topologie einer Infrastruktur-Vernetzung

Ad-hoc-Vernetzung Ein Ad-hoc-Netzwerk zeichnet sich durch das Fehlen des zuvor beschriebenen Infrastruktur-Netzwerks aus. Das Fehlen dieser zuverlässigen Infrastruktur erfordert es, dass die beteiligten Geräte direkt miteinander kommunizieren und untereinander die Weiterleitung von Datenpaketen koordinieren, wie in Abbildung 2.7 veranschaulicht ist.

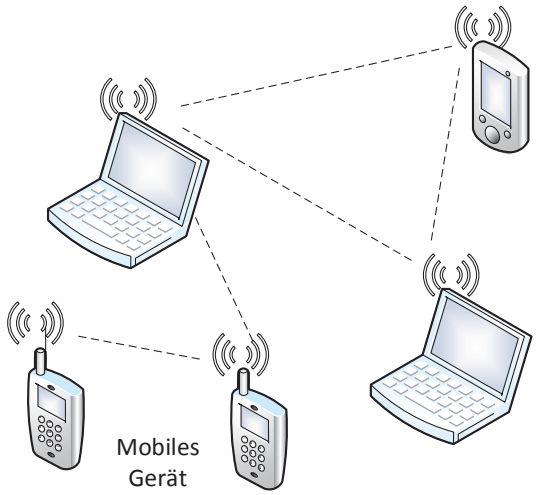

Abbildung 2.7.: Topologie einer Ad-hoc-Vernetzung

Hierdurch weisen die Netzknoten eine weitaus höhere Autonomie auf, was eine eigenständige Koordination erforderlich macht. Üblicherweise handelt es sich bei den an einer Ad-hoc-Vernetzung beteiligten Geräten um mobile Geräte. Der häufig wechselnde Nutzungskontext dieser mobilen Geräte führt jedoch dazu, dass die Geräte während ihrer Nutzung ständig bestimmte Netzwerke verlassen, während sie neue betre-

ten. Dies wiederum erklärt die ständig wechselnde Topologie, der Ad-hoc-Netzwerke unterliegen. In Ad-hoc-Netzwerken übernehmen ressourcenstärkere mobile Geräte häufig die Verwaltung des Netzwerks. Ebenso werden leistungsschwächere Teilnehmer des Netzwerks von Aufgaben wie dem Weiterleiten oder Zwischenspeichern von Nachrichten entlastet.

Eine hierzu abweichende Einteilung von [Ham05] beschreibt in diesem Zusammenhang die Kommunikationsarchitekturmodelle Client-Server und Peer-to-Peer. Erstere teilt die Teilnehmer eines verteilten Systems in Dienstkonsumenten (Clients) und Dienstproduzenten (Server) ein. Diese Rollen können dabei prinzipiell auch zwischen beteiligten Geräten wechseln – je nachdem, ob der Server selber als Konsument für andere Dienstanbieter auftritt, oder wenn beide Geräte Dienste für das jeweils andere Gerät bereitstellen. Demgegenüber beschreibt das Peer-to-Peer-Modell alle Teilnehmer als gleichberechtigte Kommunikationspartner, die ohne zentrale Kontrolle agieren. Im Allgemeinen konsumieren und produzieren sie Dienste gleichberechtigt [BCGS04].

Der Vorteil der Peer-to-Peer-Kommunikationsarchitektur ist ihr dezentraler Charakter. Dieser bringt prinzipiell eine höhere Ausfallsicherheit mit sich, da keine Koordinationsinstanz vorhanden ist, die als verfügbar und erreichbar vorausgesetzt werden muss. Dieser Vorteil führt allerdings oft zu höheren Verwaltungssaufwänden. Weiterhin erfordert die Ad-hoc-Vernetzung ein Vertrauen vieler Teilnehmer untereinander, wohingegen im Client-Server-Modell nur einem zentralen Dienstleister vertraut werden muss [Ham05].

2.2.6. Informationsaustausch mit mobilen Geräten

Eine der wichtigsten Eigenschaften mobiler Geräte ist die Fähigkeit zum Informationsaustausch, sowohl untereinander als auch mit der Infrastruktur in Form stationärer Zugangspunkte und stationärer Geräte. Die Kommunikation erfolgt dabei drahtlos oder unter Zuhilfenahme stationärer Infrastrukturkomponenten[2] [Rot02]. Die für diese Kommunikation üblicherweise verwendeten Kommunikationsstandards werden im Folgenden jeweils kurz erläutert und in die entsprechenden Architekturen der Mobilkommunikation eingeordnet.

Wireless Personal Area Networks Wireless Personal Area Networks (WPAN) stellen eine Unterkategorie lokaler Netzwerke dar, die der kostengünstigen und stromsparenden Vernetzung der verschiedenen mobilen Geräte eines Nutzers dienen [CDK12]. Im Bereich der Wireless Personal Area Networks hat sich der Bluetooth-Protokollstack (IEEE 802.15.1) [Blu16] zu einem weitverbreiteten Standard entwickelt und wird daher an dieser Stelle stellvertretend für WPANs, basierend auf [CDK12], vorgestellt.

Bluetooth ermöglicht den Aufbau von Punkt-zu-Punkt-Verbindungen oder Ad-hoc-Piconetzen. Einzelne Geräte stellen im letzteren Fall die

[2] In der vorliegenden Arbeit soll stellvertretend der Begriff der Netzwerk-Infrastrukturkomponenten verwendet werden.

Knoten eines solchen Netzes dar. In einem Piconetz existiert ein Master, der die Kommunikation mit weiteren Slaves initiiert. Ein *Piconet* kann dabei bis zu sieben aktive Knoten und insgesamt bis zu 255 passive (parked), auf eine Kommunikation wartende Knoten enthalten. Ein Knoten, der in mehr als einem Piconet aktiv ist, kann zusätzlich als Brücke zwischen Piconetzen dienen und so die Entstehung sogenannter *Scatternets* ermöglichen, wie in Abbildung 2.8 veranschaulicht wird. Die aktuelle Version 4.2 des Bluetooth-Protokolls ist auf einen niedrigeren Energieverbrauch hin ausgerichtet und erlaubt so Datenübertragungsraten von bis zu einem Mbit/s, kann jedoch unter Zuhilfenahme eines IEEE 802.11-basierten Datenkanals bis auf 24 Mbit/s angehoben werden. Die Reichweite beträgt dabei je nach Geräteklasse von unter zehn bis zu 100 Metern [Fuc09].

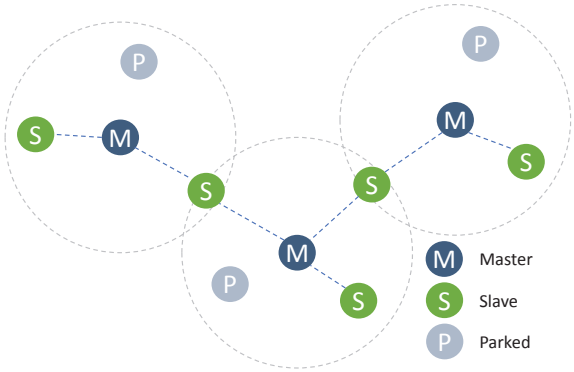

Abbildung 2.8.: Bluetooth Piconetze und Scatternets

Der Bluetooth-Protokollstack enthält eine Reihe anwendungsspezifischer Protokolle, die sogenannten Bluetooth-Profile. Einige hiervon decken unter anderem die Dienstsuche und den Netzwerkzugriff ab. Mithilfe anderer kann der für die Kommunikation in verteilten Systemen etablierte TCP/IP-Protokollstapel abgebildet werden [Min16, NSI+15]. Bluetooth bietet sich damit im Kontext dieser Arbeit an, um die energiesparende Kommunikation und Kooperation mobiler Geräte in der direkten Umgebung zu realisieren und insbesondere diese zu initiieren, da der Bluetooth-Protokollstack entsprechende Mechanismen für die Suche nach Geräten und den von ihnen angebotenen Diensten bereitstellt.

Wireless Local Area Networks Wireless Personal Area Networks (WLAN) bieten eine im Gegensatz zu WPANs deutlich höhere Abdeckung im Bereich von ungefähr 150 Metern [CDK12]. Diese ist jedoch abhängig von dazwischenliegenden Objekten, die diesen Abdeckungsbereich deutlich reduzieren können. Der Quasi-Standard für diese Art von Netzwerken ist der IEEE-Standard 802.11. Ziel dieses Standards war es, eine kostengünstige drahtlose Anbindung mobiler Geräte zu ermöglichen [CDK12]. Die WLAN-Schnittstellen mobiler Geräte können hierzu entweder im Ad-hoc- oder im Infrastrukturmodus betrieben werden [BCGS04].

Der Ad-hoc-Modus, der die zuvor beschriebene Peer-to-Peer-Kommunikation realisiert, erlaubt allerdings keine Weiterleitung von Paketen, sodass sich dieser Modus nur für eine geringe Anzahl von Clients eignet, die sich physisch nahe beieinander befinden. Weitaus üblicher ist der Infrastrukturmodus, in welchem ein Zugangsnetz in Form sogenannter Zugangspunkte (access points) realisiert ist [CDK12]. Dieser Modus ermöglicht es, dass kabellos verbundene mobile Geräte miteinander über ein Infrastrukturnetzwerk kommunizieren – unabhängig davon, ob ein direkter Funkkontakt zwischen diesen Geräten vorhanden ist. Als Infrastrukturnetzwerk ist in diesem Fall oft eine bestehende Infrastruktur für die drahtgebundene Kommunikation wie Ethernet (IEEE 802.3) direkt nutzbar. Der Standard IEEE 802.11 ermöglicht es dadurch, Zugangsnetzwerke für die drahtlose Kommunikation mobiler Geräte kostengünstig in bestehende Netzwerkinfrastrukturen einzubinden. Hierdurch können Anwendungen auf mobilen Geräten mit anderen mobilen und stationären Geräten gleichermaßen kommunizieren.

Der Standard IEEE 802.11 unterstützt hierzu in der aktuellen Norm 802.11ac [IEE13] je nach Konfiguration eine theoretische Übertragungsrate von 1300 Mbit/s, allerdings nur im 5-Ghz-Signalband, welches nur innerhalb einer geringen Distanz zur Basisstation gut nutzbar ist. Eine weite Verbreitung hat aktuell noch die vorhergehende Norm 802.11n [IEE10], die entsprechend für die Messungen und weiteren Untersuchungen in dieser Arbeit als Grundlage herangezogen wird. Mit dieser Norm sind theoretisch Bandbreiten von 600 Mbit/s erreichbar; häufig anzutreffen sind jedoch Konfigurationen, die die theoretische Datenübertragungsrate auf 450 Mbit/s beschränken. Praktisch kann diese Datenübertragungsrate jedoch kaum erreicht werden, was im Wesentlichen durch die Entfernung zum Zugangspunkt und die gemeinsame Nutzung des Übertragungsmediums durch mehrere Nutzer begründet ist.

Wireless Wide Area Networks Neben der Kommunikation in WLANs findet die drahtlose Integration mobiler Geräte auch über Wireless Metropolitan Area Networks (WMAN) oder Wireless Wide Area Networks (WWAN) statt. Ein wesentlicher Unterschied gegenüber WLANs besteht darin, dass die Realisierung dieser Infrastrukturnetze oft eine Leistung großer Netzbetreiber für Mobilkommunikation ist, die für die Nutzung der Infrastruktur und deren Dienste Gebühren erheben [Fuc09]. Hierzu authentifiziert sich ein mobiler Nutzer gegenüber dem Netzwerk und dessen Betreiber üblicherweise mithilfe eines Subscriber Identity Modules in Form einer sogenannten SIM-Karte. Im Gegensatz zu den zuvor vorgestellten Netzwerktypen, die genehmigungsfrei betrieben werden können, nutzen WMANs und WWANs lizenzpflichtige Frequenzbänder. Im Bereich der WMANs galt der IEEE-Standard 802.16 (WiMAX) über Jahre als erfolgreicher Kandidat sowohl für die ortsgebundene als auch für die mobile

drahtlose Kommunikation mit dem Internet, wurde jedoch durch Weiterentwicklungen der GSM-Standards letztlich obsolet.

Im Bereich der WWANs hat der *GSM*-Standard zum ersten Mal einer großen Anzahl von Nutzern den Zugang zur Mobilkommunikation eröffnet und die zuvor existierenden heterogenen und teilweise inkompatiblen Systeme abgelöst [Fuc09]. Der auf Textnachrichten und Telefongesprächen ausgerichtete GSM-Standard erlaubte in der ersten Generation jedoch nur Bandbreiten von 9,6 Kbit/s [DPS13]. Höhere Datenraten konnten erst mit der Erweiterung *General Packet Radio Service* (GPRS) realisiert werden. GPRS ist ein paketvermittelter, rein datenorientierter Dienst für den Informationsaustausch. Abhängig von der Signalstärke und Netzauslastung erlaubt dieser einen Datenaustausch mit Geschwindigkeiten bis zu 56 Kbit/s und ermöglicht mobilen Geräten hierdurch eine IP-basierte Konnektivität zum Internet [Fuc09].

Eine wesentliche Weiterentwicklung gegenüber GPRS stellt das *Universal Mobile Telecommunications System* (UMTS) dar. Dieser Mobilfunkstandard der dritten Generation wurde etabliert, um eine sanfte Migration von GSM hin zu modernen Codierungs- und Zugriffsverfahren zu ermöglichen, die ein vielfaches der GSM-Datenraten erlauben. Hierzu wurden Handover-Mechanismen zwischen GSM und UMTS vorgesehen, die es ermöglichen, Gespräche zwischen beiden Infrastrukturen zu übergeben [Fuc09]. UMTS erlaubt Datenraten von bis zu 384 Kbit/s im Downstream. Erweiterungen wie der *High Speed Packet Access* (HSPA) erlauben Datenraten, je nach Endgerätekategorie und Netzbetreiber, die sich bis zu 42,2 Mbit/s in Empfangsrichtung bewegen [Tel16], obwohl durch den Standard noch höhere Übertragungsraten vorgesehen sind [DPS13].

Die Nachfolgetechnologie *Long Term Evolution* (LTE) stellt die aktuelle Generation von Mobilfunkstandards dar. Ein wesentlicher Unterschied zu vorherigen Standards, bei denen die Übertragung von Daten eine Erweiterung zu einem primär für die Vermittlung von Telefongesprächen entwickelten Standard ist, besteht bei LTE darin, dass ein paketvermitteltes IP-basiertes Netz die Basis bildet. Daten und Sprachsignale werden hier paketvermittelt übertragen [DPS13]. Mit der aktuellen Protokollerweiterung *LTE Advanced*, die auch als vierte Generation der Mobilfunknetzwerke (4G) bezeichnet wird, sind theoretische Übertragungsraten, abhängig vom Endgerät und Netzbetreiber, von bis zu 300 Mbit im Downstream realisierbar [Tel16], wobei der Standard auch hier deutlich höhere Übertragungsraten bis in den Bereich von über einem GBit/s vorsieht [DPS13].

Bis zum Start der Mobilfunknetze der dritten Generation (wie UMTS) waren die Möglichkeiten für die Übertragung größerer Inhalte (wie Webseiten) stark begrenzt, lediglich die Übertragung speziell angepasster Inhalte über Protokolle wie WAP war sinnvoll möglich. Mit dem Einzug dieser (UMTS: 2004) und der folgenden Generation (LTE: 2012) von Mobilfunkstandards wurde in

Deutschland die ortsunabhängige Nutzung des Internets, zumindest in den Ballungsräumen, sinnvoll möglich. Im Bereich der WWANs ist damit eine in den vergangenen Jahren deutliche Steigerung der Übertragungsraten erzielt worden. Betrachtet man jedoch die in Abbildung 2.9 gezeigte exemplarische Abdeckung des LTE-Mobilfunkstandards durch einen Netzbetreiber, so zeigt sich, dass hohe Bandbreiten nur lokal begrenzt für mobile Nutzer verfügbar sind.

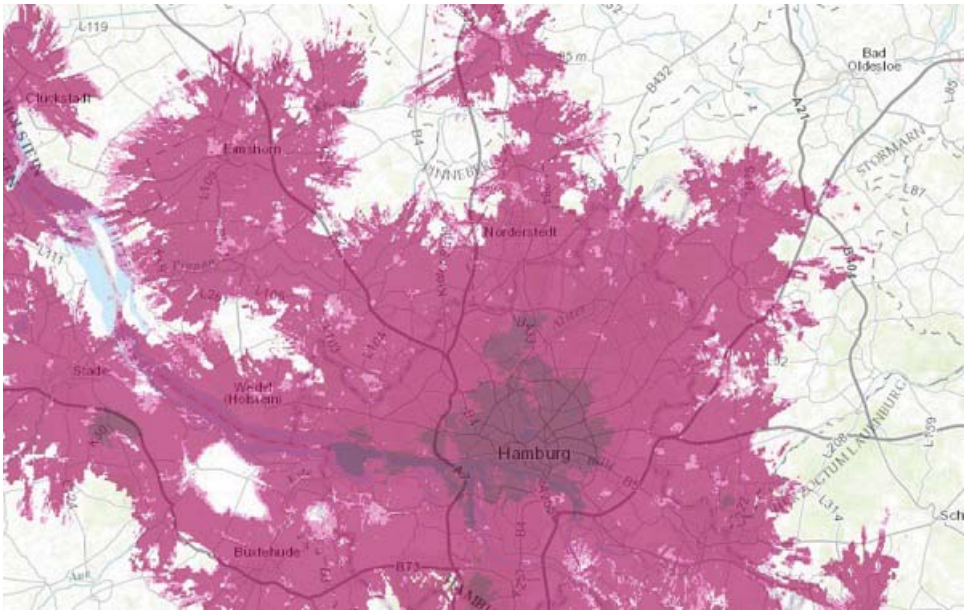

Abbildung 2.9.: Netzabdeckung des LTE Mobilfunkstandards (LTE 150) (Tel16)

Zusätzlich sind alle aus den vorhergehenden drei Abschnitten erwähnten Standards für die Mobilkommunikation anfällig für Störungen, was in Kombination mit dem schnell wechselnden Nutzungskontext mobiler Geräte zu einer häufig wechselnden Verbindungsqualität führt. Damit soll festgehalten werden, dass die im Zusammenhang dieser Arbeit betrachteten mobilen Geräte, die sich mit ihrem Nutzer bewegen, einem schnell wechselnden Kontext verschiedener Standards für die Mobilkommunikation ausgesetzt sind, was es zu berücksichtigen gilt, wenn diese Geräte mit anderen Geräten oder der Infrastruktur interagieren. Inwieweit sich klassische und mobile verteilte Systeme durch die aufgezeigten Beschränkungen unterscheiden, soll im nun folgenden Abschnitt beschrieben werden.

2.2.7. Abgrenzung klassischer und mobiler verteilter Systeme

Verteilte Systeme verfolgen eine Reihe von Zielen und weisen hiermit einhergehende charakteristische Eigenschaften auf, die im Folgenden vorgestellt werden sollen, um diese im Anschluss mit den Eigenschaften verteilter Systeme im Mobile Computing (in dieser Arbeit auch als mobile verteilte Systeme bezeichnet) zu kontrastieren. Hierfür soll eine etablierte Definition eines verteilten Systems herangezogen werden:

> *„A distributed system is a collection of independent computers that appears to its users as a single coherent system. “ [Tv08]*

Ein wesentliches Ziel verteilter Systeme ist es, den Zugang zu entfernten Ressourcen zu ermöglichen und diese dabei effizient zwischen verschiedenen Nutzern und Anwendern aufzuteilen [Tv08]. Da die beteiligten Systeme oft eine hohe Heterogenität zueinander aufweisen, ist es notwendig, dass verteilte Systeme ihre bereitgestellten Dienste und Ressourcen über standardisierte Schnittstellen und Protokolle miteinander austauschen, was zu einer Offenheit verteilter Systeme führt. Nur auf dieser Basis können zentrale Eigenschaften wie die gemeinsame Nutzung von Ressourcen, eine erhöhte Verfügbarkeit, Parallelverarbeitung und damit eine Skalierbarkeit des Gesamtsystems durch Hinzufügen weiterer Ressourcen sinnvoll realisiert werden [CDK12].

Ein grundlegendes Konzept zur Umsetzung dieser Eigenschaften ist es, die Kommunikation zwischen den beteiligten Systemen ausschließlich über den Austausch von Nachrichten zu realisieren. Der Grund hierfür liegt darin, dass es die Entwicklung eines verteilten Systems erfordert, zu berücksichtigen, dass die einzelnen Komponenten einer verteilten Anwendung oft auf die verschiedenen beteiligten Instanzen eines verteilten Systems verteilt sind. In diesem Zusammenhang wurden von Peter Deutsch Kriterien im Sinne falscher Annahmen definiert, die es bei der Entwicklung einer verteilten Anwendung zu berücksichtigen gilt [RGO06]:

- Das Netzwerk ist ausfallsicher.
- Das Netzwerk ist sicher.
- Das Netzwerk ist homogen.
- Die Topologie (des Netzwerks) ändert sich nicht.
- Die Latenz ist null.
- Die Bandbreite ist unendlich.
- Die Transportkosten (im Netzwerk) sind null.
- Es existiert nur ein Administrator.

Wie auch in [Tv08] gezeigt stehen diese Aussagen und Eigenschaften im direkten Zusammenhang mit bereits aufgeführten spezifischen Eigenschaften verteilter Systeme in Bezug auf Zuverlässigkeit, Sicherheit und Heterogenität.

Vorwegzunehmen ist in diesem Zusammenhang mit Blick auf die Eigenschaften mobiler Geräte und ihre Konnektivität, dass diese Restriktionen im Bereich mobiler verteilter Systeme noch deutlich ausgeprägter anzutreffen sind. Beispielsweise ändert sich die Topologie in Ad-hoc-Netzwerken ständig, und Bandbreite und Latenz mobiler Geräte sind einer weitaus stärkeren Dynamik unterworfen als stationäre, nicht drahtlos angebundene Geräte.

2.2.8. Transparenzeigenschaften verteilter Systeme

Ein weiteres wesentliches Ziel aus der im vorhergehenden Abschnitt gezeigten Definition eines verteilten Systems nimmt Bezug darauf, wie ein Nutzer oder eine Anwendung eines verteilten Systems mit diesem interagiert. Verbirgt ein verteiltes System seine interne Struktur und damit die einzelnen Aspekte der Verteilung vor dem Nutzer oder der Anwendung, so verhält sich das System diesen gegenüber transparent, indem es sich nach außen hin als ein Gesamtsystem darstellt [Tv08]. Transparenz meint in diesem Zusammenhang die Unsichtbarkeit oder Durchsichtigkeit eines Systems gegenüber seinen Nutzern, von denen es nicht explizit wahrgenommen wird. Das Gegenteil von Transparenz in Bezug auf Middleware ist die Awareness [Tv08].

Transparenz wird in klassischen verteilten Systemen oft angestrebt und ist auch wünschenswert, da für die Nutzer des Systems die Details der Verteilung oft nicht relevant und gegebenenfalls sogar unerwünscht sind und sie verborgen werden sollten. Diese Eigenschaft wird als Verteilungstransparenz bezeichnet und gliedert sich nach [Tv08] und [CDK12] in die folgenden Arten:

- Zugriffstransparenz: Sie beschreibt den gleichartigen Zugriff auf Ressourcen, unabhängig von Unterschieden in der Datendarstellung oder dem tatsächlichen Ressourcenzugriff. Ein Beispiel ist der Zugriff über UNC-Pfade, unabhängig vom tatsächlichen Speicherort.

- Ortstransparenz: Sie bezieht sich auf den gleichartigen Zugriff auf Ressourcen innerhalb eines verteilten Systems, unabhängig von deren physischer Position, die durch eine logische Namenszuweisung der Ressourcen realisiert wird. Ein Beispiel hierfür ist das Domain Name System.

- Migrationstransparenz: Sie verbirgt, dass ein Dienst an eine andere physische Position verschoben werden kann. Ein Beispiel ist die Verschiebung eines Namensdienstes innerhalb eines Netzwerkes.

- Relokationstransparenz: Sie erlaubt Migrationstransparenz auch während des Zugriffs auf eine Ressource durch eine Anwendung. Ein Beispiel ist die Migration eines laufenden Betriebssystemprozesses auf einen anderen physischen Host.

- Replikationstransparenz: Sie verbirgt, dass eine Ressource repliziert und damit mehrfach vorhanden ist.

- Nebenläufigkeitstransparenz: Sie verbirgt den gleichzeitigen konkurrierenden Ressourcenzugriff und stellt eine Konsistenz des Zugriffs sicher.

- Fehlertransparenz: Sie verbirgt den Ausfall und die Wiederherstellung einer Ressource.

- Skalierungstransparenz: Sie verbirgt, dass ein System durch das Hinzufügen von Ressourcen seine internen Strukturen anpasst.

Als die zwei wichtigsten Transparenzeigenschaften im Zusammenhang mit verteilten Systemen werden in [CDK12] die Zugriffs- und die Ortstransparenz genannt. Als Beispiel für die Relevanz eines orts- sowie zugriffstransparenten Dienstes wird dort die E-Mail aufgeführt, da es zum Senden einer E-Mail weder der Kenntnis des physischen noch des logischen Ortes des Empfängers im Netzwerk bedarf. Die Erfüllung der verschiedenen weiteren Transparenzeigenschaften bestimmt dabei den Grad der Transparenz eines verteilten Systems [CDK12].

Eine konkrete Technologie, die sich Transparenzeigenschaften zunutze macht, um heterogene Ressourcen nach außen hin als für den Nutzer gleichartig anzubieten, ist die Virtualisierung [CDK12]. Ziel der Virtualisierung ist es, eine Abstraktionsschicht zwischen einer Anwendung und einer Ressource zu etablieren, um beispielsweise heterogene Ressourcen wie spezielle Netzwerkadapter nach außen hin als übliche Netzwerkschnittstellen erscheinen zu lassen. Virtualisierung wird als eine der zentralen Technologien für Cloud Computing betrachtet [BVS13]. Im Cloud Computing findet sich oft die (Betriebs-) Systemvirtualisierung (system virtualization), die es zum Ziel hat, gleichartige virtualisierte Betriebssystem-Instanzen auf heterogener Hardware bereitzustellen, um die Skalierbarkeit zu erhöhen und die Austauschbarkeit der zugrunde liegenden Ressourcen zu ermöglichen.

Hohe Transparenzgrade von stationären verteilten Systemen können jedoch auch nachteilig sein, wenn es beispielsweise um die Maskierung von Systemausfällen geht, die unter Umständen besser dem Nutzer explizit mitgeteilt werden und ihm die Möglichkeit zum Abbruch bieten [CDK12]. Weitere Beispiele betreffen unter anderem die Konsistenzbedingungen. Im Zusammenhang mit Replikations- und Nebenläufigkeitstransparenz können diese zu längeren Synchronisierungsphasen führen und sich negativ auf die Performance des Gesamtsystems auswirken. Zusammen mit der noch höheren Heterogenität der Geräte in verteilten mobilen Systemen kann es sinnvoll sein, gewisse Transparenzeigenschaften aufzugeben und eine konkrete Awareness bestimmter Verteilungsaspekte den Anwendungen gegenüber explizit zu machen [Sat01]. Die Details hinsichtlich eines sinnvollen Transparenzgrades werden im Laufe der Konzeption einer adaptiven Anwendungsarchitektur weiter detailliert.

2.2.9. Mobiles Dilemma

Nachdem in den vorhergehenden Abschnitten spezielle Eigenschaften mobiler Geräte herausgestellt wurden, sollen diese in einer praxisnahen Problemstellung abschließenden verdeutlicht werden, die in [Fuc09] als das mobile Dilemma beschrieben wird.

Werden die gezeigten Eigenschaften mobiler Geräte zusammen mit den Verbindungseigenschaften betrachtet, denen diese mobilen Geräte unterworfen sind, ergeben sich gegensätzliche Anforderungen an die Architektur einer mo-

bilen Anwendung, die durch die folgenden zwei Annahmen und entsprechenden Schlussfolgerungen dargestellt werden:

- Annahme 1: Betrachtet man die gezeigten Verbindungseigenschaften mobiler Geräte, die oft einer ständig wechselnden Verbindungsqualität und Verbindungsabbrüchen unterworfen sind, und nimmt man hinzu, dass Mobilkommunikation, insbesondere in Weitverkehrsnetzen und im Ausland, vergleichsweise kostenintensiv ist, so kann laut [Fuc09] die folgende Aussage eine logische Schlussfolgerung darstellen:

- Schlussfolgerung 1: Ein Großteil der Funktionalität einer mobilen Anwendung sollte sich auf dem mobilen Gerät selber befinden, um die Kommunikationsbedarfe mit der Infrastruktur zu minimieren und damit Abhängigkeiten – möglicherweise aufgrund von Verbindungsabbrüchen nicht erreichbarer, stationärer Geräte – zu minimieren.

- Annahme 2: Betrachtet man hingegen die vorgestellten Eigenschaften mobiler Geräte, so fallen insbesondere die beschränkten Rechen- und Speicherressourcen, aber auch der endliche Energievorrat auf. Darüber hinaus bestehen meist nur beschränkte Ein- und Ausgabemöglichkeiten, zusätzlich sind diese Geräte und die auf ihnen enthaltenen Informationen einem höheren Sicherheitsrisiko, zum Beispiel durch Verlust oder Diebstahl, ausgesetzt. Dies wiederum führt zur folgenden Schlussfolgerung.

- Schlussfolgerung 2: Aufgrund der genannten Einschränkungen sollen mobile Geräte als reine Anzeige- und Eingabegeräte fungieren, also als sogenannte *Thin-Clients*. Zentrale, sicherheitskritische und rechenintensive Aufgaben sollten hingegen in die Infrastruktur verlagert werden.

Das mobile Dilemma beschreibt damit das Problem der Allokation von Funktionalitäten innerhalb eines mobilen verteilten Systems, die in einem offensichtlichen Widerspruch zueinander stehen. Laut [Fuc09] ist hier anwendungsspezifisch über eine individuelle Allokation zu entscheiden. Diese Problemstellung ist, wie einleitend dargestellt, ebenfalls eine der zentralen Problemstellungen innerhalb des mobilen Cloud Computings und wird entsprechend im Laufe der Arbeit bei der Entwicklung einer kontextadaptiven Anwendungsarchitektur für mobiles Cloud Computing aufgegriffen und weiter detailliert.

2.3. Mobile Clouds

Bevor das mobile Cloud Computing beschrieben wird, soll zunächst das übergreifende Konzept mobiler Clouds beschrieben werden. Sie sind allgemein als der Zusammenschluss mobiler Geräte zu verstehen.

2.3.1. Einführung und Begriffsklärung

Eine weitergehende Definition mobiler Clouds, die sich nicht nur auf technische Aspekte beschränkt, ist die folgende von Fitzek et al. [FK13]:

> *„[...] a mobile cloud is a cooperative arrangement of dynamically connected nodes sharing opportunistically resources. Both mobile and wireless network technologies are opportunistically combined to achieve a number of possible goals. Mobile clouds can be considered as an evolutive step towards bringing cloud–based services closer to the user themselves. “*

Im Zusammenhang mit dem Konzept der mobilen Clouds sind mobile Geräte in einem weiteren Kontext zu betrachten als nur als mobile Terminals einer zentralen Infrastruktur. Als Beispiel ist hier exemplarisch die immer größer werdende Zahl verschiedener Sensoren zu nennen. In diesem Zusammenhang werden mobile Clouds als Abstraktion für ein System, bestehend aus verteilten und miteinander verbundenen Ressourcen, bezeichnet [FK13]. Mobile Clouds sind in diesem Zusammenhang von [FK13] wie folgt charakterisiert:

- Durch die Ambition, eine Kooperation zwischen Geräten und ihren Nutzern herzustellen.
- Durch die Dynamik mobiler Datenverbindungen, die einer ständig wechselnden Verbindungsqualität unterliegen und die dazu führt, dass eine hohe Fluktuation der beteiligten Geräte in mobilen Clouds herrscht.
- Durch die Art der Verbindung der Geräte untereinander, die üblicherweise direkt miteinander verbunden sind.
- Durch die opportunistische Kooperation der verbundenen Geräte untereinander.

Ein wichtiger Aspekt im Zusammenhang dieser Charakterisierung ist es, die Ressourcen auf ein definiertes Ziel hin zu teilen. Dieses Ziel kann für ein einzelnes Gerät, mehrere Geräte oder für die gesamte mobile Cloud gelten. Dieser Aspekt ist entscheidend, wenn es um die Ausdehnung einer mobilen Cloud geht, wobei sich Mobile Clouds üblicherweise durch die von ihnen verwendeten Standards für Mobilkommunikation in ihrer Ausdehnung auf WPANs (Bluetooth) und WLANs (802.11) beschränken [FK13]. Entsprechend lassen sich nach [FK13] die folgenden drei Ebenen mobiler Clouds unterscheiden, die aufeinander aufbauen:

Kooperierende Clouds (cooperative Clouds) Eine mobile Cloud ist eine kooperative Beziehung von räumlich konzentrierten mobilen Geräten, bei denen jedes zusätzlich durch Zugangspunkte oder Basisstationen zu anderen Netzwerken verbunden sein kann. Die Ziele dieser Art der Kooperation können beispielsweise eine gesteigerte Dienstverfügbarkeit, eine

insgesamt höhere Leistung oder eine verbesserte Konnektivität im Sinne eines höheren Durchsatzes sein [FK13]. Diese Art der mobilen Clouds erlaubt beispielsweise die flexible und effiziente Nutzung üblicherweise beschränkter Ressourcen wie drahtlose Kommunikationsverbindungen, wie Abbildung 2.10 zeigt.

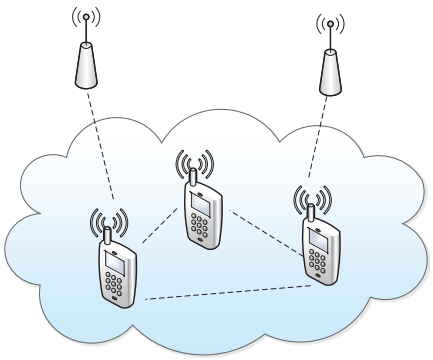

Abbildung 2.10.: Kooperierende mobile Clouds, nach (FK13)

Ressourcen-Clouds (resource clouds) Die Interaktion zwischen beteiligten Ressourcen muss sich dabei nicht auf die zuvor genannten Kombination von Kommunikationsverbindungen beschränken. Eine mobile Cloud kann hierbei als Pool für eine Vielzahl von Ressourcen dienen.

Soziale Clouds (social clouds) Als dritte Ebene oberhalb der Ressourcenbereitstellung wird die Interaktion zwischen verschiedenen mobilen Clouds betrachtet. Diese ermöglicht die Interaktion zwischen Nutzern, die sich nicht in direkter Nähe zueinander befinden, sondern in unterschiedlichen mobilen Clouds. Diese Form der Interaktion erfordert entsprechend eine Einbindung der Infrastruktur, wie in Abbildung 2.11 gezeigt wird.

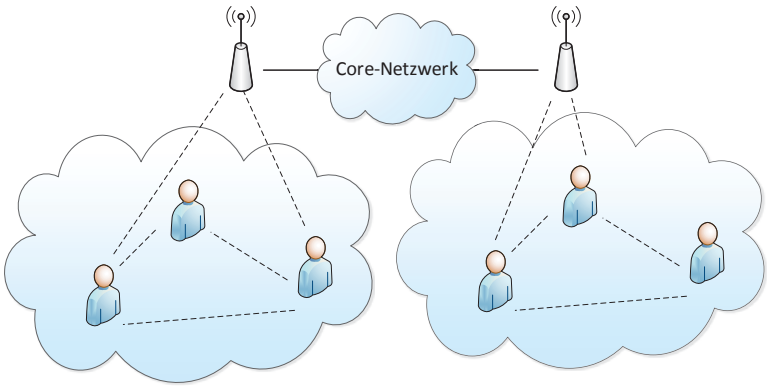

Abbildung 2.11.: Soziale mobile Clouds, nach (FK13)

Nachdem die Charakteristika, die Ziele und die verschiedenen Arten mobiler Clouds vorgestellt wurden, sollen im folgenden Abschnitt die Anwendungsbereiche der verschiedenen mobilen Clouds näher vorgestellt werden.

2.3.2. Anwendungsbereiche mobiler Clouds

Mobile Clouds bieten eine breite Anzahl von Anwendungsmöglichkeiten, die im Folgenden kurz umrissen werden sollen, um sie im Laufe dieser Arbeit konkreten Klassen mobiler Anwendungen zuordnen zu können.

Der initialen Definition einer mobilen Cloud folgend, in der Ressourcen zwischen mobilen Geräten geteilt werden, bietet es sich an, die einzelnen Anwendungsmöglichkeiten anhand der zu teilenden oder teilbaren Ressourcen dieser Geräte zu beschreiben. Nach [FK13] lassen sich die zu teilenden Ressourcen zunächst grob anhand der folgenden Kategorien definieren: Sensoren, Aktoren, Kommunikationsschnittstellen, Anwendungen, Rechen- und Speicherressourcen und Energie. Stellvertretend für diese Kategorien sollen nun im Folgenden Anwendungsmöglichkeiten für die kooperierende Nutzung dieser jeweiligen Ressourcen, angelehnt an [FK13], exemplarisch aufgezeigt werden:

Sensoren Die Mikrofone verschiedener Smartphones können bei einer Konzertaufzeichnung dazu genutzt werden, unerwünschte Nebengeräusche zu ermitteln und den Rauschabstand zu reduzieren, um hierdurch die Qualität der Aufnahme zu erhöhen. Dieses Beispiel ist ebenso auf das sogenannte *Cocktailparty-Problem* übertragbar, bei dem sich ein Zuhörer versucht, auf eine einzelne Unterhaltung in der Gruppe zu konzentrieren, während sich die anderen Teilnehmer der Runde parallel unterhalten [GDB+13]. Die kooperative Nutzung von Sensoren wie Mikrofonen bietet damit die Möglichkeit, die Fokussierung innerhalb sozialer Interaktionen zu erhöhen und kann damit der Art der Kooperationsstrategie nach auch als *Ressourcenverschmelzung* (resource amalgamation) beschrieben werden.

Ebenso erreichen die in vielen Smartphones verbauten Kameras zwar eine für ihre Größe vergleichsweise hohe Qualität in Bezug auf die Abbildungsleistung, aber auch in diesem Fall kann die kooperative Nutzung dieser Sensoren neue Anwendungsbereiche ermöglichen. Beispiele hierfür sind der in Abbildung 2.12 gezeigte Abgleich identischer Bildausschnitte zur Erhöhung des Dynamikumfangs der Bilder (High Dynamic Range) oder die Kombination mehrerer Bildausschnitte zu einem hochauflösenden Panorama.

Auch Sensoren wie GPS-Empfänger können innerhalb mobiler Clouds geteilt werden. Beispielsweise verfügt nicht jedes Gerät über diesen Sensor, oder es befindet sich nicht in einer Position mit offener Sicht zum Himmel. In diesem Fall kann diese Information von einem Gerät in der direkten Umgebung bezogen werden. Weitere Sensoren, die von dieser Art der Kooperation profitieren können, sind der Temperatur- oder der Luftdrucksensor, welche im Rahmen einer Erfassung von Umweltdaten im Sinne eines *Crowdsensings* ausgewertet werden können.

Speicher Eine andere Kategorie von teilbaren Ressourcen stellt der Speicher mobiler Geräte dar. Wie in [SKK+90] gezeigt wird, lassen sich die oft

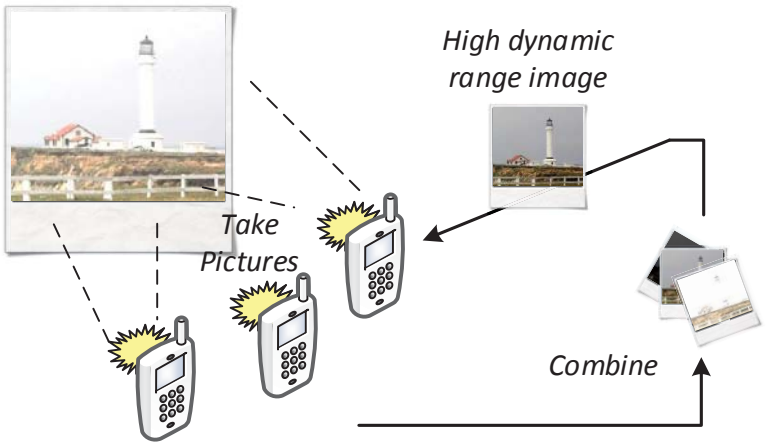

Abbildung 2.12.: Erzeugung von Bildern mit hohem Dynamikumfang

begrenzten Speicherressourcen mobiler Geräte nutzen, um einen großen Teil der wahrscheinlich zu einem bestimmten Zeitpunkt benötigten Daten auf oder in der Nähe eines mobilen Gerätes vorzuhalten. Ein weiterer Aspekt dieser Verteilung betrifft die Ausfallsicherheit. Wird berücksichtigt, dass mobile Geräte verloren oder beschädigt werden können, wie im Zusammenhang mit dem mobilen Dilemma aufgezeigt wurde, kann eine Replikation im Sinne einer gewünschten Redundanz die Verfügbarkeit dieser Daten erhöhen.

Prozessor Ressourcenintensive mobile Anwendungen belasten oft primär den Prozessor eines mobilen Geräts. Eine Auslagerung dieser Berechnungen auf andere Geräte einer mobilen Cloud kann es ermöglichen, mit diesen rechenintensiven Teilen der *Geschäftslogik* (business logic) einer mobilen Anwendung nicht nur ein einzelnes mobiles Gerät zu belasten, sondern diese Belastung zu verteilen. Diese Kooperationsform wird oft auch als mobiles Cloud Computing beschrieben.

Energie Obwohl keine der existierenden drei Arten mobiler Clouds direkte Übertragung von Energie ermöglicht, besteht insbesondere durch die Kooperationsformen der aufgezeigten verteilten Sensornutzung, der verteilten Speicherung von Daten und der Auslagerung von Berechnungen die Möglichkeit, potenziell energieintensive Aufgaben auf andere Teilnehmer einer mobilen Cloud zu verlagern. Hierdurch kann indirekt der begrenzte Energievorrat eines bestimmten mobilen Geräts geschont werden.

Die aufgezeigten Beispiele zeigen die vielfältigen Anwendungsmöglichkeiten mobiler Clouds. Der für diese Arbeit relevante Teilbereich des mobilen Cloud Computings soll im folgenden Abschnitt detailliert werden.

2.4. Mobiles Cloud Computing

Bei Betrachtung der bisher in diesem Kapitel vorgestellten Eigenschaften mobiler Geräte und insbesondere der eingeschränkten und wechselnden Konnektivität dieser Geräte stellt sich die Frage nach einer angemessenen Aufteilung der Funktionalität einer mobilen Anwendung zwischen mobilen Geräten und der sie umgebenden Infrastruktur. Diese Problemstellung wurde in Abschnitt 2.2.9 als das mobile Dilemma veranschaulicht. Als *Infrastruktur* wird in Bezug auf das mobile Cloud Computing dabei nicht nur die Infrastruktur im Sinne eines Infrastrukturnetzwerks (vergleiche Abschnitt 2.2.5) und seiner Komponenten verstanden, sondern ebenso die an diese Infrastruktur angebundenen übrigen stationären Geräte, einschließlich der umliegenden Geräten und bis hin zu den über das Internet angebundenen Cloud-Rechenzentren.

Wird das Problem der Aufteilung der Funktionalität aus dem mobilen Dilemma jedoch als Chance dafür begriffen, flexibel und kontextabhängig zu entscheiden, welcher Teil der Geschäftslogik einer mobilen Anwendung auf einem mobilen Gerät und welcher Teil sich in der Infrastruktur befindet, so entstehen eine Reihe neuer Anwendungsmöglichkeiten für mobile Geräte. Das Forschungsgebiet, welches sich mit den Fragestellungen beschäftigt, die die Verbindung zwischen Mobile Computing und Cloud Computing herstellen, wird auch als mobiles Cloud Computing bezeichnet [De16].

Im folgenden Abschnitt erfolgt eine Einführung in das mobile Cloud Computing, und es werden die für diese Arbeit relevanten Grundlagen erläutert, bevor in den folgenden Abschnitten weitere für diese Arbeit relevante Teilaspekte detailliert werden.

2.4.1. Einführung

Die Ressourcenteilung, die Verteilung und die Mobilität von Anwendungen sind keine erst mit dem mobilen Cloud Computing entstandenen Konzepte. So wurde die *Ressourcenteilung*, zum Beispiel das *Timesharing*, innerhalb von Großrechnern (Mainframes) bereits in den frühen 1960er-Jahren realisiert. Die Möglichkeit, auf diese Großrechner mithilfe von Terminals zuzugreifen, ermöglichte gleichzeitig die Benutzermobilität (vergleiche Abschnitt 2.2.2). Ergänzt um die in den 1970er-Jahren zusammen mit dem *Distributed Computing* entwickelten Verteilungsstrategien und der Virtualisierung von Ressourcen wurde zusätzlich die Komponentenmobilität ermöglicht und damit der Weg für das bedarfsbezogene Nutzungsmodell des Cloud Computings geschaffen. In Teilen ähnelt die als mobile Cloud Computing beschriebene Architektur, dem Prinzip der Terminals und des Großrechners der 1960er- und 1970er-Jahre.

Mit Einführung des *Personal Computers* wurde diese Architektur zu einer stärker dezentralisierten Systemarchitektur weiterentwickelt, da Personal Computer durch ihre schnellere Reaktionszeit bei der Anzeige und Verarbeitung grafisch aufwendigeren Inhalten Vorteile boten, die mithilfe von Terminals über die damals vorhandenen Übertragungskanäle nicht hätten realisiert

werden können. Die in Abschnitt 2.2.6 aufgezeigten Weiterentwicklungen im Bereich der Mobilfunkstandards haben diese Dezentralisierung in den vergangenen Jahren jedoch entbehrlich gemacht, und es zeichnet sich erneut eine stärkere Zentralisierung der Ressourcen innerhalb von Cloud-Rechenzentren ab, die mit leichtgewichtigen mobilen Geräten wie Smartphones interagieren.

Obwohl durch die aufgezeigten Entwicklungen die Forschungsgebiete der verteilten Prozessausführung, der Prozessmigration und der Virtualisierung bereits seit über vierzig Jahren existieren und vielfältig untersucht sind, können diese etablierten Verfahren zur Mobilität von Anwendungen zwischen mobilen Geräten und Cloud-Ressourcen nur bedingt genutzt werden, da sie auf die hohe Dynamik mobiler Umgebungen und die Heterogenität der mobilen Geräte nicht ausgelegt sind [De16].

2.4.2. Historie des Begriffs

Mobiles Cloud Computing versucht die aufgezeigten Leistungsbeschränkung mobiler Geräte durch eine Interaktion und Kooperation im Sinne mobiler Clouds zu reduzieren. Erste Ansätze einer solchen Integration mobiler Geräte und ihrer Infrastruktur gehen auf das Emerald-System [JLHB88] aus dem Jahre 1988 zurück. Dieses stellt eine objektorientierte Sprache zur Verfügung, die die Entwicklung verteilter Anwendungen ermöglicht und bei der die einzelnen Objekte sowie deren Größe von einem einzelnen Datum bis hin zu einem gesamten Betriebssystemprozess innerhalb eines Computernetzwerkes zur Laufzeit verschoben werden können. Diese Lösung wurde für lokale Netzwerke entwickelt, deren Struktur als stabil angesehen wird, sodass Referenzen zwischen Objekten auf unterschiedlichen Systemen ebenfalls als stabil angesehen werden können.

Weitere Forschungsarbeiten haben sich bis zum Jahr 2000 primär mit der Etablierung von Infrastrukturen für die Auslagerung von Berechnungen beschäftigt [KLLB13]. Ein wichtiger Teilbereich hiervon, der bis heute aktiv untersucht wird, betrifft die sogenannte Auslagerungsentscheidung; dies beinhaltet, zu entscheiden, ob ein bestimmter Berechnungsschritt von einem mobilen Gerät in die Infrastruktur verlagert werden soll.

In den folgenden Jahren wurden eine Reihe weiterer Lösungsansätze entwickelt, die sich mit der Problemstellung beschäftigen, mobile Geräte in bestehende verteilte Systemarchitekturen zu integrieren. Das Ziel der Integration war dabei sehr oft die Auslagerung von Berechnungen. Diesem prominenten und intuitiven Anwendungsfall geschuldet finden sich in der Literatur viele Belege, die das mobile Cloud Computing mit der Auslagerung von Berechnungen (computation offloading) gleichsetzen; diese Beobachtung ist jedoch nur bedingt richtig, wie auch von Fitzek und Katz in [FK13] festgestellt wird. Entsprechend soll festgehalten werden, dass mobiles Cloud Computing in der vorliegenden Arbeit ebenfalls nicht mit der Auslagerung von Berechnungen gleichzusetzen ist und es sich hierbei lediglich um einen weitverbreiteten Anwendungsfall handelt.

2.4.3. Aktuelle Definition

Im Hinblick auf die zuvor beschriebenen Ziele des mobilen Cloud Computings soll die weitverbreitete Definition von Dinh et al. für die Definition des Begriffs des mobilen Cloud Computings in dieser Arbeit herangezogen werden:

> *„MCC integrates the cloud computing into the mobile environment and overcomes obstacles related to the performance (e.g., battery life, storage, and bandwidth), environment (e.g., heterogeneity, scalability, and availability), and security (e.g., reliability and privacy) discussed in mobile computing". [DLNW13]*

In Bezug auf die in Abschnitt 2.2.1 vorgestellten Forschungszweige des Mobile Computings ist mobiles Cloud Computing damit nah mit der Idee des per Pervasive Computing verbunden [Sat01]. Dies wird mit der eher technischen Ausprägung des Begriffs begründet, die von M. Satyanarayanan in [Sat01] als die Verbindung der Forschungsgebiete der verteilten Systeme und des Mobile Computings aufgefasst wird. Somit entsprechen eine Reihe der Herausforderungen und Problemstellungen im Bereich des Pervasive Computings auch den Problemstellungen im Bereich des Cloud Computings und der verteilten Systeme, wobei die Übertragbarkeit der dort vorhandenen Lösungsansätze nicht immer gewährleistet ist [Sat01]. Mobiles Cloud Computing bringt aufgrund der hohen Dynamik und Heterogenität mobiler Clouds jedoch eine Reihe von weiteren Anforderungen und Restriktionen mit sich. In diesem Zusammenhang können mobiles Cloud Computing und Mobile Clouds einem inhaltlich weitestgehend deckungsgleichen, jedoch zeitlich früher definierten Term zugeordnet werden, dem des *Cyber Foraging*, ein Begriff der von M. Satyanarayanan im Jahre 2001 wie folgt definiert wurde:

> *„Cyber foraging, construed as "living off the land," may be an effective way to deal with this problem. The idea is to dynamically augment the computing resources of a wireless mobile computer by exploiting wired hardware infrastructure. " [Sat01]*

Einer aktuelleren Definition folgend zielt Cyber Foraging darauf ab, mobile Anwendungen zu entwickeln, die sich im Spannungsfeld zwischen den Forschungsgebieten der verteilten Systeme und des Mobile Computings bewegen und in denen die Aufteilung und Zuordnung von Funktionalitäten einer mobilen Anwendung dazu führt, dass diese dynamisch zwischen den mobilen Geräten und der Infrastruktur verschoben werden können [Fli12]. Die durch Cyber Foraging entstehende Augmentation mobiler Geräte, im Sinne einer Anreicherung ihrer Eigenschaften und Funktionen mithilfe der sie umgebenden Infrastruktur, erweitert die verteilte Ausführung zwischen mobilem Gerät und Cloud auf weitere Geräteklassen. Sind diese Geräte speziell für diesen Zweck vorgesehen und befinden sie sich in der direkten Umgebung des Nutzers, so werden diese auch als *Cloudlets* [Sat01] bezeichnet, was ihre Verwandtschaft mit ihren zentralisierten Pendants, den Cloud-Rechenzentren, verdeutlichen soll.

Der Definition folgend, dass im mobilen Cloud Computing sämtliche stationären Geräte und die Komponenten eines Infrastrukturnetzwerks als Infrastruktur bezeichnet werden, ergeben sich zwei verschiedene Formen mobiler Clouds im mobilen Cloud Computing:

- Das von M. Satyanarayanan in [Sat01] beschriebene klassische Konzept, in welchem die Infrastruktur aus stationären Geräten wie Cloudlets und physisch weiter entfernten Geräten wie Cloud-Servern besteht.
- Das Konzept in [Sat01] erweiternd, bei dem umliegende mobile Geräte zusätzlich der Infrastruktur hinzugerechnet werden. In diesem Fall treten Geräte oft wechselweise sowohl als Konsument als auch als Anbieter von Ressourcen auf [VSDTD12b].

Auf Grundlage dieser verschiedenen Architekturen lassen sich den verschiedenen Geräten innerhalb einer mobilen Cloud Rollen zuordnen, die sich darauf beziehen, ob Geräte entweder Ressourcen konsumieren oder Ressourcen anbieten. In diesem Zusammenhang werden alle Anbieter von Ressourcen innerhalb des mobilen Cloud Computings, die ihre Funktionalität anderen Geräten bereitstellen und somit stellvertretend für sie agieren, entsprechend auch als Stellvertreter oder *Surrogates* bezeichnet und werden von Kemppainen et al. definiert als:

> *„A surrogate is a computing entity that augments the mobile device over a network connection takes over the the computational work of a functionality of a mobile application.“ [Kem11]*

In diesem Zusammenhang existieren weitere zum mobilen Cloud Computing verwandte Konzepte, die sich allerdings weiter als die zuvor genannten Ausprägungen voneinander differenzieren. Entsprechend existieren für diese Konzepte eigene Begriffe, die im Folgenden jeweils kurz umrissen werden, um sie anschließend in Bezug auf die wesentlichen Unterschiede voneinander abzugrenzen.

2.4.4. Mobiles Edge- und Fog Computing

Neben dem mobilen Cloud Computing existiert das etwas jüngere Konzept des *mobilen Edge Computings*, es ist nach Ahmed et al. definiert als:

> *„Mobile Edge Computing is a model for enabling business oriented, cloud computing platform within the radio access network at the close proximity of mobile subscribers to serve delay sensitive, context-aware applications. “ [AA16]*

Im Gegensatz zum mobilen Cloud Computing liegt der Fokus des mobilen Edge Computings auf der Kooperation zwischen ressourcenbeschränkten Geräten, bei der sich die für die Auslagerung genutzten ressourcenstärkeren Geräte oft in der direkten Nähe des ressourcenbeschränkten mobilen Gerätes befinden. Nachdem mobiles Cloud Computing die Grundlage für die Kooperation

zwischen mobilen Geräten und zentralisierten Ressourcen geschaffen hat, erweitert mobiles Edge Computing dieses Konzept zu einem Ansatz, in welchem versucht wird, einen Großteil der auszulagernden Operationen in die direkt umgebende Infrastruktur zu verlagern. Mobiles Edge Computing ähnelt damit der im vorhergehenden Abschnitt vorgestellten Definition des mobilen Cloud Computings, bezieht jedoch die mobilen Geräte selbst in die Kategorie der Ressourcenanbieter mit ein. Die so entstehenden Netzwerke werden in Anlehnung an mobile Clouds auch als *Edge Clouds* bezeichnet [RSW+16, CWSR14], entsprechend werden die Surrogates in diesem Zusammenhang auch als *Edge Nodes* [SVHVV15] oder in dieser Arbeit auch als *Edge-Cloud-Infrastruktur* bezeichnet.

Diese weitergehende Differenzierung in einem eigenen Begriff wird auch mit der Weiterentwicklung der Netzwerk-Infrastrukturkomponenten begründet. Diese bieten neben der Weiterleitung von Datenpaketen zusätzliche Aufgaben wie die Bereitstellung von Ausführungsumgebungen für virtuelle Maschinen an [Cis15] und ermöglichen es hierdurch, Rechen- und Speicherressourcen an den logischen Kanten (edges) eines Netzwerks bereitzustellen. Ein weiteres Beispiel sind LTE-Basisstationen, die die Ausführung von durch Drittanbieter bereitgestellten Softwarekomponenten ermöglichen [Nok15]. Diese Konzentration auf die Ressourcenbereitstellung an den logischen Kanten eines Netzwerks führt nach [OBL15b] und [Sat17b] zu den folgenden Vorteilen gegenüber einer mobilen Cloud-Computing-Architektur, die mithilfe einer zentralisierten Ressourcenbereitstellung umgesetzt wird:

- Geringere Latenz: Die Datenverarbeitung erfolgt in der physischen Nähe der Nutzer, wodurch sich die Zeit für die Datenübertagung reduziert und es mobilen Anwendungen ermöglicht, auch zeitkritische Funktionalität in die Infrastruktur zu verlagern.
- Entlastung des Infrastrukturnetzwerks: Die Datenverarbeitung an den logischen Kanten eines Netzwerks entlastet das Infrastrukturnetzwerk und reduziert die Anforderungen an immer höhere Übertragungskapazitäten desselbigen.
- Erhöhte Zuverlässigkeit: Die Verteilung von Funktionalitäten einer mobilen Anwendung in Form von Diensten, die zuvor zentral in Cloud-Rechenzentren bereitgestellt wurden, in eine Vielzahl von Geräten innerhalb eines Netzwerks erhöht die Verfügbarkeit dieser Dienste.
- Kontextbewusstsein: Der Zustand einer Edge Cloud kann sich in Bezug auf Ressourcenallokation und notwendige Funktionalität direkt an seine Nutzer und deren Anforderungen anpassen.
- Erhöhte Sicherheit: Die Verarbeitung von schützenswerten Informationen in derselben geografischen Region, in der sich auch der Nutzer eines mobilen Gerätes befindet, sorgt zum einen dafür, dass dieselben datenschutzrechtlichen Richtlinien Anwendung finden, und zum anderen, dass diese Informationen nicht über das Internet zu übermitteln sind.

Zusammengefasst erlaubt es mobiles Edge Computing damit, die Funktionalität, die im mobilen Cloud Computing in eine weitgehend zentralisierte Infrastruktur verlagert wird, in die direkte Umgebung eines mobilen Gerätes zu verlagern, um so die aufgezeigten Vorteile, insbesondere der geringeren Latenz und erhöhten Zuverlässigkeit, zu realisieren. Ein weiteres Konzept, welches dem des mobilen Edge Computings ähnelt, ist das *Fog Computing*, von Bonomi et al. definiert als:

> *„.. a highly virtualized platform that provides compute, storage, and networking services between end devices and traditional cloud computing data centers, typically, but not exclusively located at the edge of network. “ [BMZA12]*

In diesem Zusammenhang folgen mobile Anwendungen, die eine Architektur im Rahmen des mobilen Edge Computings nutzen, dem Ziel, lokale Probleme auch lokal zu lösen [NKI+15]. Diese Nähe bringt in vielen Fällen auch eine deutlich verringerte Latenz mit sich, was insbesondere in der Kombination mit klassischem mobilem Cloud Computing eine insgesamt größere Anzahl von Anwendungen die Möglichkeit eröffnet, an den Potenzialen der Kooperation mobiler Geräte mit ihrer Infrastruktur zu partizipieren, die sich wie in [OBL15b] beschrieben wie folgt charakterisieren lassen:

- Die Bereitstellung von Inhalten in der direkten Umgebung eines Nutzers.
- Die Verarbeitung großer Datenmengen, ohne die Erfordernis diese Daten zunächst in eine zentralisierte Infrastruktur zu übertragen.
- Geografisch verteilte Anwendungen, zum Beispiel Sensornetzwerke zu beschleunigen.
- Die verteilte Erfassung und Analyse in Echtzeit.
- Latenzsensitive Anwendungen wie das Cloud Gaming oder die Analyse von Videodaten in Echtzeit zu ermöglichen.

Den gezeigten Ansätzen ist damit gleich, dass sie versuchen, die Ressourcenbeschränkung mobiler Geräte durch eine Kooperation mit deren Infrastruktur zu umgehen. Ein Grund für die überlappende Definition dieser Begriffe kann auch darin gesehen werden, dass die Kanten (edges) eines Netzwerks nicht klar definiert sind. So kann ein mobiles Gerät selber noch zur Kante eines Netzwerks gezählt werden (edge device) oder lediglich der letzte kabelgebundene Zugangspunkt zum Netzwerk als Kante desselbigen bezeichnet werden.

Abbildung 2.13 verdeutlicht abschließend die in diesem Abschnitt eingeführten Begriffe und aufgezeigten Kooperationsformen mobiler Geräte untereinander und mit ihrer Infrastruktur und setzt die vorgestellten Begriffe und Konzepte zueinander in Beziehung.

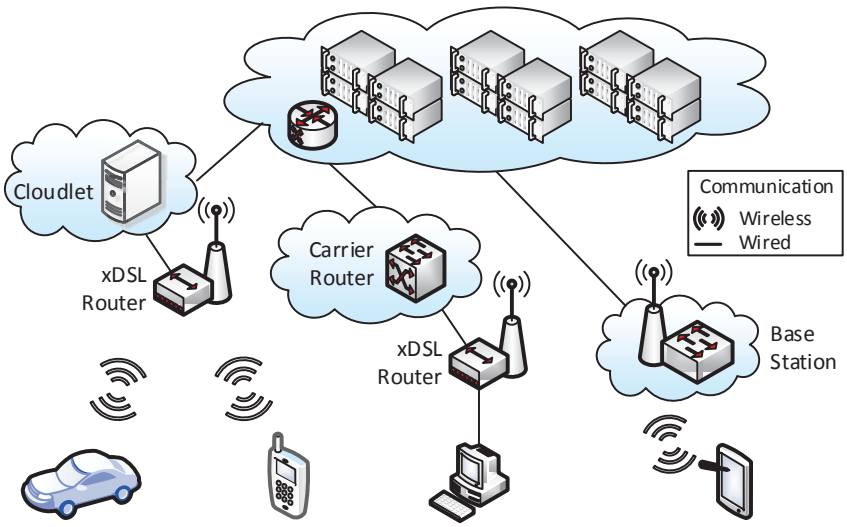

Abbildung 2.13.: Mobiles Cloud Computing

2.5. Zusammenfassung

Mobile Computing beschreibt sowohl die Nutzung drahtloser Kommunikationsschnittstellen als auch die ortsveränderliche Nutzung von Computern. In diesem Zusammenhang können die sogenannten Endgerätemobilität, die Benutzermobilität und die Dienstmobilität als eine wesentliche Klassifikation von Mobilitätsformen unterschieden werden.

Die mit der Mobilität dieser Geräte einhergehende Miniaturisierung führt allerdings zu einer Reihe von Einschränkungen gegenüber stationären Geräten, die sich insbesondere in einer begrenzten Rechen- und Speicherkapazität, einem endlichen Energievorrat und einer häufig wechselnden Konnektivität dieser mobilen Geräte zu der von ihnen genutzten Infrastruktur zeigen. Das „Mobile Dilemma“ veranschaulicht die sich aus diesen Eigenschaften ergebende Problematik der sinnvollen Allokation von Anwendungen und Daten zwischen dem mobilen Gerät und der genutzten Infrastruktur. Mobile Clouds versuchen diese Ressourcenbeschränkungen zu umgehen, indem sie eine Kooperation mobiler Geräte untereinander und mit der von ihnen genutzten Infrastruktur forcieren.

In diesem Zusammenhang versucht das mobile Cloud Computing die Ressourcenbeschränkung mobiler Geräte dadurch zu umgehen, dass es die durch das Cloud Computing bereitgestellten und nahezu unendlichen Ressourcen nutzt, um hierdurch die Funktionalität mobiler Geräte zu erweitern und seinen Nutzern eine neue Klasse von Anwendungen bereitzustellen. Neuere Konzepte wie das Fog oder Edge Computing fokussieren in diesem Zusammenhang zusätzlich die Nutzung umliegender Ressourcen. Die Vorteile dieser neueren und stärker dezentralisierten Ansätze sind im Wesentlichen in einer höheren Ausfallsicherheit und geringeren Latenz zu finden.

3. Kontextverarbeitung und Adaption

Die im vorhergehenden Kapitel vorgestellten Nutzungsszenarien und Eigenschaften mobiler Geräte lassen erkennen, dass die Berücksichtigung des Kontextes dieser mobilen Geräte einen wesentlichen Aspekt in Bezug auf die Nutzung und Entwicklung mobiler Anwendungen darstellt. Mobile Anwendungen und mobile Geräte werden in diesem Zusammenhang oft als kontextbewusst (context-aware) beschrieben, oder es wird ein kontextbewusstes Verhalten von ihnen gefordert.

Entsprechend wird zunächst das Verständnis des Kontextbegriffes erläutert und im Anschluss die Struktur von Kontextdaten, der Prozess der Kontextverarbeitung und die Auswertung von Kontextdaten beschrieben. Dieser Bereich der Kontextsensitivität wird dann verlassen, um anschließend die Nutzung von Kontextinformationen zur Realisierung adaptiven Verhaltens in mobilen Anwendungen zu beschreiben. Hierzu werden Adaptionsgründe und Adaptionsformen vorgestellt, um damit die relevanten Grundlagen für die Entwicklung eines Prozesses zur Kontextadaption in Kapitel 5 erneut aufzugreifen. Diese Grundlagen werden bewusst zunächst abstrakt gehalten, um im Hinblick auf die Entwicklung eines möglichst generischen Prozesses zur Kontextadaption im Vorwege keine Einschränkungen hinsichtlich konkreter, verwendeter Kontextinformationen oder Verarbeitungsprozesse vorzunehmen.

3.1. Kontextsensitivität

Neben der begrenzten Speicher- und Rechenleistung besteht ein wesentliches Problem der Nutzung mobiler Anwendungen und Geräte im oft wechselnden Nutzungskontext derselbigen, wie in Kapitel 2 beschrieben wurde. Fragestellungen im Bereich des Mobile Computings und damit auch des mobilen Cloud Computings erfordern entsprechend eine Berücksichtigung des Kontextes, die bei stationären Geräten und Anwendungen in dieser Form nicht erforderlich ist [OXK+15]. Nur hierdurch kann allerdings das Verhalten der mobilen Anwendung die Interaktion mit dem Nutzer positiv beeinflussen und so zu einer sinnvollen Unterstützung, einer einfachen Benutzbarkeit und entsprechend insgesamt zu einem hohen Nutzen für ihre Anwender beitragen.

Was in diesem Zusammenhang unter dem Begriff des Kontextes zu verstehen ist, welche Arten von Kontext existieren und wie entsprechende Kontextdaten strukturiert sind, soll im nun folgenden Abschnitt erläutert werden. Um die existierenden Lösungsansätze, die im Verlauf dieser Arbeit vorgestellt werden, erläutern und bewerten zu können, erfolgt hierzu eine Erläuterung der unterschiedlichen Definitionen dessen, wie Kontextdaten klassifiziert werden können.

3.1.1. Definition des Kontextbegriffs

Es existieren eine Reihe von Definitionen, die versuchen zu beschreiben, was unter dem Begriff Kontext, dem Kontextbewusstsein und kontextbewusstem sowie auch kontextadaptivem Verhalten zu verstehen ist. Eine allgemeine Definition dessen, was unter dem Begriff des Kontextes zu verstehen ist, liefert hierzu das Wörterbuch Merriam-Webster [Mer16]:

> *1: the parts of a discourse that surround a word or passage and can throw light on its meaning*
>
> *2: the interrelated conditions in which something exists or occurs*

Hieraus lässt sich zunächst ableiten, dass die Begriffe *Zusammenhang* und *Umgebung* eine wichtige Rolle in Bezug auf den Begriff des Kontextes spielen. Im Hinblick auf die Informatik wurde der Begriff des Kontextes erstmalig von Mark Weiser in [Wei91] verwendet, dabei wurde jedoch nicht definiert, was dieser beschreibt. Die von Mark Weiser beschriebene Vision des Ubiquitous Computings besagt lediglich, dass Anwendungen und Geräte auf ihre Nutzer reagieren. Diese Fähigkeit zur Reaktion erfordert es, dass Anwendungen und Geräte sich ihrer Umwelt und damit ihres Kontextes bewusst sind [Wei91]. Eine Reihe weiterer früher Beispiele kontextbewusster Anwendungen haben sich in Bezug auf den Kontext zunächst auf nur eine spezifische Kontextinformation wie den Ort beschränkt (location awareness) [LHB11], was allerdings nur einen unvollständigen Ausschnitt dessen beschreibt, was unter dem Begriff des Kontextes verstanden wird. Eine frühe Definitionen, die den Begriff des Kontexts im Zusammenhang mit der Informatik versucht, näher abzugrenzen, ist die von William Schilit in [Sch95] vorgestellte:

> *„Three important aspects of context are: where you are, who you are with, and what resources are nearby [...]. Context encompasses more than just the user's location, because other things of interest are also mobile and changing. Context includes lighting, noise level, network connectivity, communication costs, communication bandwidth, and even the social situation, e.g., whether you are with your manager or with a co-worker." [Sch95]*

Diese Definition erweitert das Verständnis dessen, was unter dem Begriff des Kontexts in Bezug auf die Informatik zu verstehen ist, hin zu einer allgemeineren Beschreibung der Umgebung eines Nutzers. Diese Definition wird von Brown et al. in [BBC97] um den Aspekt der Zeit erweitert und unterteilt Kontextinformationen kontextbewusster Anwendungen in kontinuierliche (continuous) und diskrete Kontextinformationen. In ihrer Definition berücksichtigen Brown et al. somit, welche Kontextinformation mit welchem zeitlichen Intervall in Verbindung gebracht wird. Nähert sich dieses Intervall der Echtzeit an, so werden die Kontextinformationen als kontinuierlich betrachtet [BBC97]. Als Beispiel für kontinuierliche Kontextinformationen wird von den

Autoren ein Kompass-Sensor in Kombination mit einer Kartenanwendung angeführt.

Eine weitere Konkretisierung erfährt der Kontextbegriff im Jahr 1998 durch Salber et al. in [SDA99], die den Zustand eines Nutzers, Gruppen oder Objekte mit einbeziehen:

> *„Environmental information or context covers information that is part of an application's operating environment and that can be sensed by the application. This typically includes the location, identity, activity and state of people, groups and objects. „[SDA99]*

Von Dey in [Dey00] und Dey und Abowd in [DA99] werden diese bisherigen Definitionen hin zu einer sehr generischen Definition dessen erweitert, die sich zu einem etablierten Verständnis dessen entwickelt hat, was unter dem Begriff des Kontextes zu verstehen ist:

> *„Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.“ [DA99]*

Dieser Definition nach kann es sich bei Kontext um jede Art von Information aus dem Umfeld einer Anwendung handeln, die für die Interaktion mit dem Nutzer relevant und damit nützlich erscheint. Entsprechende Informationen erstrecken sich dabei auch auf die Anwendung und den Nutzer selbst. Eine solche Definition abstrahiert damit von konkreten technischen Gegebenheiten, Umweltbedingungen oder sozialen Zuständen und beschreibt diese abstrakt als Entitäten. Diese anerkannte Definition, dessen was unter Kontext zu verstehen ist, soll auch in dieser Arbeit gelten, wobei die einzelnen Entitäten im Folgenden als Kontextinformationen im Sinne der Definition von Dey und Abowd verstanden werden sollen.

3.1.2. Kontextbewusstsein

Die Betrachtung der Nutzungsszenarien mobiler Geräte macht offensichtlich, dass Informationen, die der Definition von Dey und Abowd nach als Kontextinformationen zu verstehen sind, sich durch die mobile Nutzung dieser Geräte und ihrer Anwendungen häufigen Veränderungen unterworfen sind. Der Annahme folgend, dass sich die Benutzbarkeit mobiler Geräte und ihrer Anwendungen erhöht, wenn diese sich den Veränderungen anpassen, ist es erforderlich, dass mobile Geräte und ihre Anwendungen ein Bewusstsein über den aktuellen Kontextzustand besitzen.

Zusammen mit ihrer initialen Definition dessen, was unter dem Begriff des Kontextes zu verstehen ist, definieren Schilit und Theimer in [ST94] das Adjektiv *kontextbewusst* (context-aware) wie folgt:

> *„Applications are context-aware when they adapt themselves to context“.*

Auch Dey und Abowd definieren hinsichtlich ihrer Beschreibung des Kontextbegriffs das Adjektiv kontextbewusst:

> *„A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task." [DA99]*

Obwohl sich das Adjektiv *context-aware* im englischen Sprachraum sowohl auf die Fähigkeit bezieht, den Kontext wahrzunehmen als auch auf ihn zu reagieren, soll hierzu abschließend noch differenziert werden, dass sich in dieser Arbeit die übliche deutsche Übersetzung, das Adjektiv „kontextbewusst", lediglich auf die Fähigkeit beschränkt, den Kontext wahrzunehmen, beispielsweise um ihn für eine zukünftige Auswertung zu speichern. Schließt das Kontextbewusstsein jedoch auch eine Anpassung auf selbigen ein, so soll dies als „kontextadaptiv" bezeichnet werden. Diese Anpassung kann dabei auf Basis des historischen, aktuellen oder auch des zukünftigen Kontextes einer mobilen Anwendung oder eines mobilen Gerätes erfolgen. Letztere wird von Nurmi et al. in [NMF+05] auch als die Proaktivität (proactiveness) des Kontextbewusstseins definiert.

3.1.3. Struktur von Kontextdaten

Im vorhergehenden Abschnitt wurde beschrieben, dass das Kontextbewusstsein mobiler Anwendungen und Geräte zu einer kontextabhängigen Adaption derselbigen führen kann. Diese Anpassung[1] kann es zum Ziel haben, die Nutzbarkeit mobiler Anwendungen zu erhöhen. Abbildung 3.1 veranschaulicht hierzu einen Ausschnitt der von einem mobilen Gerät durch Sensoren erfassten Kontextinformation.

Abbildung 3.1.: Ausgewählte Sensoren eines mobilen Gerätes

Welche Kontextinformationen für diese Anpassung berücksichtigt werden können und wie sie klassifiziert werden, sollen nun im folgenden Abschnitt beschrieben werden. Der Fokus liegt entsprechend auf dem für diese Arbeit relevanten Teilbereich des Kontextes mobiler Geräte und ihrer Anwendungen.

Typen von Kontextdaten Für eine weitergehende Klassifikation von Kontextinformationen bietet es sich zunächst an, sie anhand der Schnittstelle, über

[1] Die Begriffe Anpassung und Adaption sind synonym zu verstehen.

die sie erfasst werden, zu klassifizieren. Schnittstellen, die diese Daten (aus ihrer Umgebung) erfassen, werden auch als Sensoren bezeichnet. Diese Sensoren und die von ihnen wahrgenommenen Kontextdaten lassen sich nach Baldauf et al. in die folgenden drei Kategorien einteilen [BDR07]:

- Physisch wahrgenommener Kontext: Diese Kategorie von Kontextdaten wird durch *physische Sensoren* aus der realen Welt erfasst. Ein Beispiel ist die Temperatur, die über einen Temperatursensor wahrgenommen wird, die Helligkeit, die über eine Fotodiode wahrgenommen wird, oder die Beschleunigung, die über einen Beschleunigungssensor wahrgenommen wird.

- Virtuell wahrgenommener Kontext: Die Kategorie virtuell wahrgenommener Kontextdaten bezieht ihre Daten aus virtuellen Sensoren. Diese Kontextdaten werden über *virtuelle Sensoren* erhoben und umfassen beispielsweise die Termine aus dem elektronischen Terminkalender eines Nutzers, die über eine entsprechende Schnittstelle abrufbar sind.

- Abgeleiteter Kontext: Diese Kategorie von Kontextdaten wird über *logische Sensoren* erfasst, die die Daten verschiedener physischer und virtueller Sensoren aggregieren und diese um zusätzliche Informationen aus Datenbanken oder weiteren Quellen anreichern, um neue Informationen zu generieren. Beispielsweise kann ein logischer Sensor die Daten eines Beschleunigungssensors so anreichern und auswerten, dass die verschiedenen Aktivitätsausprägungen wie Gehen, Laufen oder Fahrradfahren daraus abgeleitet werden können.

Hierarchien von Kontextdaten Das vorhergehende Beispiel zeigt, dass Kontextdaten nicht nur nach der Art des Sensors, über den sie erfasst werden, klassifiziert werden können, sondern auch auf Basis des Grades der Verarbeitung unterschieden werden können. Insbesondere kann der Typ des abgeleiteten Kontextes weiter in verschiedene Hierarchien von Kontextdaten eingeordnet werden.

In diesem Zusammenhang unterscheiden Dey und Abowd in [DA99] Kontextdaten anhand der Stufe der Aggregation in einen primären und sekundären Kontext. Primärer Kontext (primary context) ist in diesem Zusammenhang jede Information, die ohne weiteren existierenden Kontext abgeleitet oder ohne jede Form von Sensordatenfusion erzeugt werden kann. Sekundärer Kontext (secondary context) charakterisiert sich entsprechend über jede Information, die aus einem primären Kontext ermittelt werden kann. Ein sekundärer Kontext kann entsprechend beispielsweise durch die Aggregation von Sensordaten erzeugt werden und entspricht damit dem von Baldauf et al. definierten abgeleiteten Kontext.

Am Beispiel der Rohdaten eines GPS-Sensors lassen sich so die gelieferten Werte in Bezug auf Höhen- und Breitengrad als der primäre Kontext beschreiben, wohingegen daraus abgeleitete Informationen wie der Abstand zwischen

zwei ermittelten Orten oder eine visuelle Darstellung dieser Information als sekundärer Kontext bezeichnet wird [PZCG14].

In diesem Zusammenhang werden von Dey in [Dey00] Kontextinformationen in Low-level- und in High-level-Informationen unterschieden. Erstere umfassen hierbei die Menge an Kontextinformationen, die sich direkt mithilfe physischer und virtueller Sensoren wahrnehmen lassen, wie beispielsweise die Temperatur oder die Spannung (elektrisches Potential) des Akkus eines mobilen Gerätes. Letztere beziehen sich hingegen auf den abgeleiteten Kontext, der sich aus der Verarbeitung von Low-level-Informationen ergibt. Wie allerdings von Sigg in [Sig08] festgestellt wird, ist die Abgrenzung dessen, was als Rohdaten, Low-level-Kontext oder High-Level-Kontext bezeichnet wird, nicht immer eindeutig. Generell merkt Sigg hierzu an, dass höhere Formen der Kontextrepräsentation öfter symbolische Präsentationen darstellen, niedrigere dagegen numerische, wie in Tabelle 3.1 veranschaulicht wird.

Folglich sollen in der vorliegenden Arbeit die Kontextdaten physischer und virtueller Sensoren als primärer Kontext oder Low-level-Kontext verstanden werden. Gleichzeitig sollen die Kontextdaten logischer Sensoren als abgeleiteter Kontext und damit als High-Level-Kontext verstanden und damit als hierarchische Struktur für Kontextdaten herangezogen werden.

High-level Kontext	Low-level Kontext	Rohdaten	Kontextdatenquelle
Laufen	14°C	1001111	Thermometer
Laufen	57.2°F	1001111	Thermometer
Telefonieren	64dB	109	Mikrofon
Lautstärkemessung	64dB	109	Mikrofon
am Strand	53.538671'N; 9.927731'E	GPRMC (..A,7153.8671,N,..)	GPS-Sensor
auf der Bank	34.122777'N; 118.303890'W	GPRMC (..7734.1227,N,..)	GPS-Sensor
Schreiben	z	0x7a	Tastatur [de]
Büro besetzt	z	0x7a	Tastatur [de]

Tabelle 3.1.: Hierarchien von Kontextdaten nach (Sig08)

Auf Basis der Untersuchung verschiedener Kontextklassifizierungen soll damit zusammenfassend festgehalten werden, dass als primäres Strukturierungsmerkmal für Kontextdaten oft hierarchische Strukturen herangezogen werden, wie es ebenso von Kühn et al. in [KKS13] festgestellt wurde.

Granularität und Qualität Bei Betrachtung der zuvor vorgestellten Hierarchien von Kontextdaten liegt es nahe, dass die verschiedenen Aggregationsebenen nur bis zu einem bestimmten Grad teilbar sind. In diesem Zusammenhang wird von Himberg in [Him04] und nachfolgend von Baldauf et al. in [BDR07] in ähnlicher Form der Begriff des Kontextatoms definiert, welches nach Himberg als Merkmal gemeinsam als kleinste unteilbare Menge an Kontextinformatio-

nen mit semantischer Bedeutung beschrieben wird [Him04]. Nach [HMNS13] stellt ein Kontextatom beispielsweise ein aus einem Sensordatenstrom abgeleitetes Merkmal dar, welches sich aus einer Attributgenerierung (feature extraction) ergibt und als kontinuierlicher normalisierter Wert darstellen lässt:

> *„We call the extracted features context atoms. Each atom describes the action of the user and/or state of the environment. [...] We define the context atom as a continuous variable from 0 to 1. A context atom is a single feature that has been extracted from the environment of the node.“ [HMNS13]*

In dieser Arbeit soll stellvertretend für die vorgenannte Definition der Begriff Kontextattribut gelten, der unter anderem in [vSvHW+06] gleichbedeutend mit dem Begriff des Kontextatoms von Himberg verwendet wird. Nach Baldauf et al. besitzt solch ein Kontextattribut zwei Ausprägungen, anhand derer es beschrieben werden kann: seinen Typ, zum Beispiel „Temperatur“, und seinen Wert, beispielsweise „20 °C“, wobei die Einheit vom Typ des Kontextatoms abhängig ist. In diesem Zusammenhang wird von Baldauf et al. angemerkt, dass diese Art von Kontextinformation für die Entwicklung kontextsensitiver Anwendungen oft nicht ausreichend ist und weitere Attribute wie die Zeit, die Quelle und die Sicherheit der Messung eines Kontextattributs als zusätzliche Information eines Kontextatoms oder Kontextattributs herangezogen werden sollten [BDR07]. Die Sicherheit über eine bestimmte Kontextinformation wird von Buchholz et al. als Qualität eines Kontexts bezeichnet (quality of context) und ist definiert als: *„any information that describes the quality of information that is used as context information.“* [BKS03]. Nach [vSvHW+06] kann diese Kontextqualität anhand der folgenden Kriterien, die im Sinne von Metainformationen die Qualität einer Kontextinformation definieren, näher charakterisiert werden:

- Genauigkeit (accuracy): Die Differenz zwischen dem Wert des Kontextattributs und dem realen Zustand, den es repräsentiert.
- Konfidenz (probability of correctness): Die von der Kontextquelle ermittelte Sicherheit, dass sie den richtigen Wert bereitstellt
- Vertrauenswürdigkeit (trustworthiness): Die vom Empfänger eines Kontextattributs ermittelte Sicherheit, dass das von der Kontextquelle bereitgestellte Kontextattribut den richtigen Wert hat.
- Aktualität (up-to-dateness): Das Alter einer Kontextinformation, üblicherweise repräsentiert durch einen Zeitstempel.

Diese Charakterisierung verdeutlicht, dass einzelne Kontextattribute in eine Reihe von Dimensionen eigeordnet werden können, um ihnen über die Grenzen einer Hierarchisierung und Typisierung hinweg zusätzliche Bedeutung zuzuordnen. Die Zuordnung von Kontextinformationen in diese Dimensionen schafft eine weitergehende Strukturierung von Kontextdaten, die nachfolgend

vorgestellt werden soll, um die im Verlauf dieser Arbeit verwendeten Kontextinformationen entsprechend einordnen zu können.

Dimensionen des Kontexts Im Zusammenhang mit der Strukturierung von Kontextdaten in Dimensionen bietet es sich hierzu an, zunächst die grundlegende Klassifikation von Schilit in [Sch95] heranzuziehen, die Kontextinformationen auf Basis der folgenden drei Fragen einordnet:

> ***Where you are:*** *This includes all location related information such as GPS coordinates, common names (e.g. coffee shop, university, police), specific names (e.g. Canberra city police), specific addresses, user preferences (e.g. user's favourite coffee shop).*
>
> ***Who you are with:*** *The information about the people present around the user.*
>
> ***What resources are nearby:*** *This includes information about resources available in the area where the user is located, such as machinery, smart objects, and utilities.*

Obwohl diese initiale Definition bereits eine hohe Passfähigkeit in Bezug auf das in dieser Arbeit primär betrachte Anwendungsgebiet mobiler Geräte aufweist, wird sie einer Reihe der Dimensionen der umfassenderen Kontextdefinition von Dey nicht gerecht. In diesem Zusammenhang findet sich in [KKS13] eine umfassende Klassifizierung der Dimensionen des Kontexts mobiler und ubiquitärer Systeme, die die Dimensionen Interaktionskontext, sozio-technischer Kontext, physischer Kontext, Taskkontext, räumlicher Kontext und temporaler Kontext unterscheidet.

Im Zusammenhang mit dem in dieser Arbeit untersuchten Kontext mobiler Geräte, im Hinblick auf ihre Kooperation mit der sie umgebenden Infrastruktur, fokussiert sich der relevante Kontext im Wesentlichen auf die letzten vier Dimensionen, die entsprechend im folgenden Abschnitt im Rahmen der Betrachtung domänenspezifischer Kontextmodelle mobiler Geräte weiter detailliert werden.

Kontext mobiler Geräte Die Untersuchungen verschiedener domänenspezifischer Kontextmodelle für mobile Geräte [VIP16, Mül10, PZCG14] zeigen, dass eine Vielzahl verschiedener Modelle existieren, von denen an dieser Stelle stellvertretend nur ein repräsentatives und im Anschluss ein im Hinblick auf diese Arbeit besonders relevantes vorgestellt werden sollen. Ferreira klassifiziert hierzu in [Fer13], basierend auf einer Definition von Dey und Abowd in [DA99], den Kontext mobiler Geräte anhand der folgenden Dimensionen:

> ***Who:*** *the unique identity of the entities (e.g. sensor or application).*

> ***What:*** *the characteristics of the entities that can be labeled, measured or inferred (e.g. a label for a geo-location coordinate, currently engaged physical activity for a person, etc.)*
>
> ***When:*** *the instance of time in which the event is occurring*
>
> ***Where:*** *the location (e.g. place, application, sensor) of the event*
>
> ***Why:*** *the intelligibility of the system or application, the user intent and accountability of the system, application and user*

Ein konkreteres und weitgehend dieselben Kontextdimensionen abdeckendes Modell wird von Mayrhofer et al. in [MRF03] vorgeschlagen und nimmt eine Kategorisierung des Kontexts anhand der folgenden Attribute vor:

- geografisch: der Ort, die Straße, der Raum
- physisch: die Helligkeit, die Lautstärke oder die Temperatur
- organisatorisch: die Zuordnung zu einer bestimmten Gruppe
- sozial: der Familienstand oder Beziehung zu einer Person
- emotional: der Puls oder der Blutdruck
- nutzerbezogen: das Nutzungsprofil des mobilen Geräts, wie „lautlos“
- taskabhängig: arbeitend oder schlafend
- aktionsabhängig: sitzend, laufend, stehend oder fahrradfahrend

In diesem Zusammenhang und bei Betrachtung der verschiedenen Definitionen soll festgehalten werden, dass sich räumliche und zeitliche Bezüge in vielen der vorgestellten Definitionen als wichtige Dimensionen wiederfinden, so wird auch von [BBH+10] festgestellt, dass der Aspekt der räumlichen Bezüge (spatial model) ein entscheidender Faktor in vielen Kontextdefinitionen ist, auch da er relativ einfach zu ermitteln ist. Diese Klassifizierung weist eine hohe Passfähigkeit zu den Dimensionen des Kontextes mobiler Geräte auf, der in dieser Arbeit betrachtet wird, und soll entsprechend im Folgenden als stellvertretend herangezogen werden.

Ferreira merkt in diesem Zusammenhang an, dass die so entstehende Menge an Kontextinformationen, die von mobilen Geräten erhoben werden, von ebendiesen mobilen Geräten nicht immer eigenständig und in gewünschter Qualität verarbeitet werden kann [Fer13]. Entsprechend soll, nachdem definiert wurde, wie Kontextinformationen beschrieben werden können und welcher Teil des Kontextes für mobile Geräte besonders relevant ist, im Folgenden erläutert werden, wie diese Informationen im Hinblick auf die Nutzung in mobilen Anwendungen effizient verarbeitet werden können.

3.2. Prozess der Kontextverarbeitung

Im Zusammenhang mit der Verarbeitung von Kontextdaten werden von Karin Henricksen in [Hen03] die Herausforderung vor allem durch einen schnell wechselnden Kontext, eine hohe Dynamik, oft und schnell wechselnde Anforderungen der Nutzer sowie durch Beschränkungen mobiler Geräte hinsichtlich ihrer Leistungsfähigkeit und ihrer beschränkten Ressourcen beschrieben. Aus diesen wiederkehrenden Anforderungen haben sich in den vergangenen Jahrzehnten wiederkehrende Vorgehensweisen etabliert, aus denen standardisierte Prozesse zur Verarbeitung von Kontextdaten entstanden sind [XP12]. Dies zeigt sich konkret in der Entwicklung entsprechender Systemunterstützungen wie *MUSIC* [RBD+09], *CASS* [FC04] und *SOCAM* [GPZ05] zur Entwicklung kontextsensitiver Anwendungen. In diesem Zusammenhang wird von Hynes et al. die Bereitstellung von Kontextdaten bereits als standardisierter Dienst (*Context as a Service*) bezeichnet [HRH09].

Bezüglich der einzelnen Prozessschritte der Verarbeitung von Kontextdaten lassen sich je nach Umsetzung zwar einzelne Unterschiede erkennen. Bei einer gewissen Abstraktion lassen sich allerdings die von Perera et al. in [PZCG14] identifizierten vier Schritte der Akquisition, Modellierung, dem Reasoning und der Distribution, die in Abbildung 3.2 veranschaulicht werden, in einer Vielzahl gängiger Prozesse zur Verarbeitung von Kontextdaten erkennen.

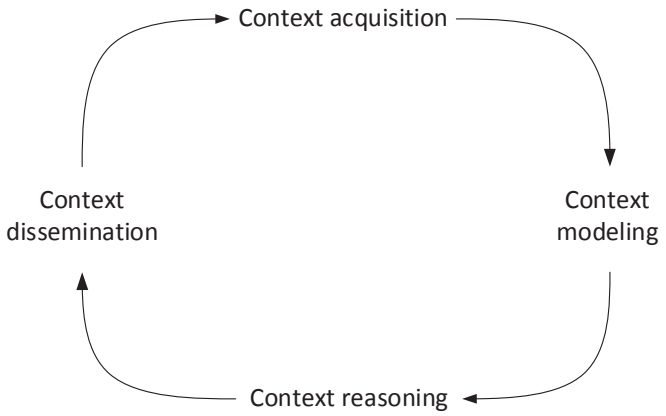

Abbildung 3.2.: Prozessschritte der Kontextverarbeitung nach (PZCG14)

Wesentliche Aspekte dieser einzelnen Prozessschritte, die in [PZCG14] vorgestellt werden und die für die Betrachtung von Kontextdaten im Rahmen der Themenstellung dieser Arbeit relevant sind, sollen nachfolgend vorgestellt werden.

3.2.1. Akquisition

In der Phase der Akquisition gilt es, den Kontext von verschiedenen Quellen zusammenzufassen. Neben der Differenzierung nach der Art des Sensortyps (vergleiche Abschnitt 3.1.3) können diese nach Perera et al. auch basierend auf

der Verantwortlichkeit, der Frequenz, der Quelle oder nach dem Akquisitionprozess (acquisition process) unterschieden werden. Diese unterscheiden sich nach [PZCG14] wie folgt:

Basierend auf der Verantwortlichkeit Kontext kann entweder aktiv angefragt werden (pull) oder bei einer Veränderung des entsprechenden Kontextattributs (publish and subscribe / push). Die zweite Variante erfordert es, dass ein Empfänger sich als Interessent für Veränderungen des entsprechenden Kontextattributs registriert, um bei einer Veränderung entsprechend benachrichtigt zu werden.

Basierend auf der Frequenz Hierbei kann zwischen einer zeitpunktbezogenen und einer intervallbasierten Kontextakquisition unterschieden werden. Zeitpunktbezogen, im exakten Moment des Ereignisses, beispielsweise in dem Moment, in dem ein Beschleunigungssensor eine Beschleunigung registriert. Oder intervallbasiert (periodically), wenn die Sensordaten periodisch abgefragt (pull) oder aktualisiert (push) werden. Ein Beispiel ist die regelmäßige Überprüfung verfügbarer Netzwerke zur Mobilkommunikation durch ein mobiles Gerät.

Basierend auf der Quelle Die Bereitstellung von Kontextdaten kann direkt durch den Zugriff auf die entsprechende Programmierschnittstelle (API) der Sensorhardware erfolgen. Die Bereitstellung kann jedoch ebenfalls durch eine entsprechende Systemunterstützung in Form einer Middleware erfolgen, die im Gegensatz zum direkten Sensorzugriff eine Abstraktion von Low-level-Schnittstellen bietet und hierdurch die Wiederverwendbarkeit erhöht. Die dritte Variante stellt ein sogenannter Kontext-Server dar, wie er ebenso in der Klassifikation der verschiedenen Architekturen des Sensorzugriffs von Baldauf et al. in [BDR07] vorgeschlagen wird. Dieser kann ressourcenbeschränkte Geräte unterstützen, die benötigten Kontextinformationen zu beziehen, wenn diese sie nicht selbst akquirieren können.

Basierend auf dem Sensortyp Wie in Abschnitt 3.1.3 beschrieben wurde, können die Sensortypen zur Akquisition von Kontextdaten in physische, virtuelle und logische Sensoren unterschieden werden.

Basierend auf dem Akquisitionsprozess In diesem Fall wird unterschieden, ob eine Kontextinformation direkt durch einen Sensor bereitgestellt wird oder ob die Kontextinformation mithilfe weiterer Verarbeitungsschritte aus existierenden Kontextdaten ermittelt wird.

In Bezug auf diese Arbeit ist dabei insbesondere die Akquisition basierend auf der Quelle relevant, da die hohe Heterogenität mobiler Geräte es erfordert, unterschiedlichste Sensoren mithilfe wiederverwendbarer Schnittstellen zu erfassen.

Je nach Prozessdefinition wird der Akquisition von Kontextdaten ebenfalls die Speicherung derselbigen zugerechnet. In [RCKZ13] wird die Speicherung

hingegen als zusätzlicher Schritt im Zusammenhang mit der Verarbeitung von Kontextdaten auf mobilen Geräten gesehen. Die Speicherung größerer Datenmengen auf ressourcenbeschränkten Geräten, die zusätzlich nur sporadisch mit der Infrastruktur verbunden sind, stellt erhöhte Anforderungen an diesen Prozessschritt, ist allerdings von enormer Bedeutung für die Ermittlung größerer Zusammenhänge in Kontextdaten, da dies häufig nur in Verbindung mit der Analyse oder Berücksichtigung vergangener Kontextinformationen möglich ist.

3.2.2. Modellierung und Repräsentationsformen

Im Anschluss an die Akquisition von Kontextinformationen können diese durch unterschiedliche Repräsentationen in Form verschiedener Kontextmodelle dargestellt werden. Unter der Modellierung ist hierbei die Nutzung generischer Datenmodelle zu verstehen, um zu beschreiben, wie Kontextinformationen repräsentiert und benutzbar gemacht werden [XP12]. Ein *Kontextmodell* definiert und speichert in diesem Zusammenhang die Werte von Kontextattributen in einer maschinenlesbaren Form und beschreibt ihre Beziehungen zueinander. Es ist nach von van Sinderen et al. definiert als:

> *„A context model is an information model designed to represent context information. It describes context attributes, their mutual relationships, and their values.“ [vSvHW+06]*

Neben dieser auf die systemtechnische Modellierung bezogene Definition existiert eine frühere Definition von Henricksen, die ein Kontextmodell basierend auf den Kontextinformationen, die es enthält, klassifiziert:

> *„A context model identifies a concrete subset of the context that is realistically attainable from sensors, applications and users and able to be exploited in the execution of the task. The context model that is employed by a given context-aware application is usually explicitly specified by the application developer, but may evolve over time.“ [Hen03]*

Hiernach identifiziert ein Kontextmodell eine Teilmenge von Kontextinformationen, die von Sensoren, Anwendungen und Nutzern wahrgenommen werden kann und der üblicherweise durch den Anwendungsentwickler einer kontextsensitiven Anwendung definiert ist. In diesem Zusammenhang können nach [BBH+10] Kontextmodelle in statische oder dynamische Modelle unterschieden werden, wobei statische Modelle darauf beschränkt sind, vorher definierte Strukturen von Kontextmodellen zu halten, wohingegen dynamische Modelle bei der Verwendung erweitert oder angepasst werden können.

Der Prozess, der Kontextdaten in eine Modellierung bringt, gliedert sich nach [PZCG14] dabei in zwei Schritte. In eine Modellierungsphase, in der neue Kontextinformationen in Form von Kontextattributen in Beziehung zueinander gesetzt und mit Qualitätsattributen versehen werden, und in eine zweite Phase, in der sowohl existierende als auch neue Kontextinformationen in Form des Modells zur Verwendung bekannt gemacht werden.

Im Laufe der Jahre haben sich hierfür gewisse Gruppen von Repräsentationsformen zur Kontextmodellierung etabliert, die im Folgenden angelehnt an [SLP04] und [XP12] vorgestellt werden:

Key-Value-Modelle (key value models) Dieser Ansatz verwendet einfache Schlüssel-Wert-Kombinationen, um Kontextdaten zu modellieren. Der Vorteil dieser Modellierung liegt in der Einfachheit der Implementierung in relationalen Datenbanksystemen. Jedoch erlaubt es dieser Ansatz nicht, komplexe semantische Strukturen und Zusammenhänge auszudrücken und direkt Schlussfolgerungen aus diesen Strukturen zu ziehen.

Auszeichnungsschemata (markup scheme) Bei diesem Ansatz werden mittels semistrukturierter Beschreibungssprachen wie der Extensible Markup Language (XML) Kontextattribute und ihre Werte anhand einer hierarchischen Struktur aus Markup Tags beschrieben. Diese Form der Kontextmodellierung wird vor allem dort angewendet, wo versucht wird, die hohe Dynamik und Komplexität von Kontextinformationen adäquat zu beschreiben, da sie zur Laufzeit erweiterbar ist.

Grafische Modelle (graphical models) Grafische Modelle bestehen aus einer Menge von grafischen Elementen, die genutzt werden, um Kontextinformationen zu repräsentieren. Ein in diesem Zusammenhang weitverbreiteter Ansatz ist die Verwendung der Unified Modeling Language (UML). Ein Beispiel hierfür findet sich in der von Henricksen vorgeschlagenen Erweiterung des Object-Role Modeling in [HIR03].

Objektorientierte Modelle (object oriented models) Objektorientierte Modelle versuchen die etablierten und erfolgreichen Konzepte der Objektorientierung, unter anderem Kapselung und Wiederverwendbarkeit, auf das Problemfeld der Kontextmodellierung zu übertragen, um die Dynamik des Kontexts innerhalb von ubiquitären Systemen beherrschbar zu machen. Das objektorientierte Paradigma zur Verarbeitung von Kontextdaten ist dabei insbesondere für den Entwickler objektorientierter Anwendungen als leicht erlernbar und verständlich anzusehen.

Logikbasierte Modelle (Logic based Models) Bei dieser Modellierungsform wird Kontext als Menge von Elementen innerhalb eines logikbasierten Systems repräsentiert, welches Ausdrücke, Fakten und Regeln definiert. Die Aktualisierung der Kontextinformation findet hier üblicherweise durch eine Anpassung der Faktenbasis statt. In diesem logikbasierten System können dann durch Schließen (reasoning) aus der Wissensbasis direkt ermittelte oder abgeleitete höherwertige (High-level)-Kontextinformationen bereitgestellt werden.

Ontologiebasierte Modelle (ontology based) Ontologien versuchen als strukturierendes Mittel die formal geordnete Darstellung und den Sinnzusammenhang zwischen einer Menge von Begrifflichkeiten innerhalb

eines bestimmten Bereichs zusammenzufassen. Sie werden als eine vielversprechende Art der Kontextrepräsentation angesehen, wenn es um die Abbildung vielfältiger Zusammenhänge innerhalb maschinenlesbarer Datenstrukturen geht.

Für eine weitergehende Bewertung der unterschiedlichen Modellierungsansätze wird auf [PZCG14] verwiesen. Zusammenfassend soll an dieser Stelle jedoch festgehalten werden, dass die Auswahl eines geeigneten Modellierungsansatzes stark von der konkreten Problemstellung abhängig ist und im Wesentlichen durch den nun vorgestellten nächsten Prozessschritt, dem Reasoning, geprägt ist.

3.2.3. Reasoning

Unter dem Begriff des Reasonings wird die Ableitung und Generierung neuen Wissens verstanden [VIP16]. In Bezug auf die Verarbeitung von Kontextdaten erfolgt in diesem Prozessschritt die Analyse der Kontextdaten, um hieraus neue Informationen und letztlich neues Wissen über die Zusammenhänge in den Daten herzustellen.

Der Prozessschritt des Reasonings weist entsprechend eine hohe Ähnlichkeit zu etablierten Prozessen für Mustererkennung und Wissensgenerierung wie *CRISP-DM* [She00] oder *KDD* [FPSS96] auf; es existieren jedoch auch domänenspezifische Unterschiede.

Die allgemeine Erfordernis des Reasonings entsteht aus zwei Eigenschaften von Low-level-Kontextdaten: Ihrer Ungenauigkeit, die sich beispielsweise in einer Doppeldeutigkeit (ambiguity) zeigt, und ihrer Unsicherheit (uncertaincy) [PZCG14]. Der Prozess des Reasonings versucht diese Probleme aufzulösen und entsprechende Muster zu erkennen. Er besteht aus drei Schritten, der Vorverarbeitung von Kontextdaten, der Sensordatenfusion und der Kontext-Inferenz, die im Folgenden jeweils kurz beschrieben werden:

Vorverarbeitung von Kontextdaten Die Vorverarbeitung stellt einen wesentlichen Schritt dar, wenn aus bestehenden Daten auf neues Wissen geschlossen werden soll. Die Vorverarbeitung von insbesondere Sensordaten bedarf dabei zunächst einer Korrektur fehlender Werte (imputation of missing values), der Entfernung unüblicher und unrichtiger Werte (outlier detection) und gegebenenfalls auch einer Konsistenzprüfung. Dieser Verarbeitungsschritt entspricht im KDD-Prozess dem Teilschritt der Vorverarbeitung.

Sensordatenfusion Ein weiterer, speziell für die Verarbeitung von Kontextdaten relevanter Schritt ist die Zusammenführung einzelner Sensordaten hin zu einer genaueren und vollständigeren Sicht; ein Schritt, der insbesondere bei Sensordaten als relevant angesehen wird, um die Genauigkeit gegenüber einem einzelnen Sensor zu erhöhen [HL97]. Dieser Verarbeitungsschritt entspricht im KDD-Prozess weitestgehend dem Teilschritt der Transformation.

Kontext-Inferenz In diesem Schritt werden aus Low-level-Informationen neue High-level-Kontextinformationen erzeugt. Ein Beispiel hierfür findet sich nach [PZCG14] in der Verarbeitung von GPS-Informationen in Bezug auf den Höhen- und Breitengrad hin zu der Information, dass es sich um ein Café handelt. Ein weiterer Schritt kann nun die Auswertung einer Historie beinhalten, anhand derer festgestellt wird, dass es sich um einen vielbesuchten Ort dieses Nutzers handelt. Als ein Teil der Kontext-Inferenz[2] findet innerhalb dieses Verarbeitungsschritts aus Sicht des KDD-Prozesses das Data Minings (die Mustererkennung) statt.

Diese Mustererkennung kann mithilfe unterschiedlicher Verfahren erfolgen. Bei der Mustererkennung in Kontextdaten mobiler Geräte wurden von Perera et al. in [PZCG14] sechs verschiedene Kategorien identifiziert, die sich hierfür besonders gut eignen und entsprechend oft in existierenden Lösungsansätzen angewendet wurden:

Überwachtes Lernen (supervised learning) Beim überwachten Lernen handelt es sich um einen weitverbreiteten Ansatz im Bereich des statistischen Schließens. Hierbei versucht ein Lernalgorithmus eine Abbildung zu finden, die möglichst gut jedem Eingabewert den vermuteten Ausgabewert zuordnet. Nach diesem Lernprozess, auch als Training bezeichnet, sollte das so entstandene Modell fähig sein, für neue und unbekannte Beispiele, die Testmenge, eine korrekte Ausgabe zu liefern. Die Abweichung zum tatsächlichen Ausgabewert wird im Fall einer falschen Ausgabe mithilfe eines Fehlermaßes erfasst. Das Ziel ist es entsprechend, das Fehlermaß auf der Testmenge, mit der nicht trainiert wird, zu minimieren. Für die Aufteilung von Trainings- und Testmenge kommen häufig Kreuzvalidierungsverfahren zur Anwendung. Erfolgt das Lernen allerdings mit dem Ziel zukünftige Ereignisse zu erkennen, gilt es dies bei der Aufteilung dahingehend zu berücksichtigen, dass beim Lernen keine Information über diese zukünftigen Ereignisse verwendet wird. Entsprechend kann hierzu die in Abbildung 3.3 gezeigte Aufteilung genutzt werden.

Abbildung 3.3.: Exemplarische Selektion von Trainings- und Testmenge

Unüberwachtes Lernen (unsupervised learning) Unüberwachtes Lernen unterscheidet sich vom überwachten Lernen dadurch, dass die Ausgabewerte nicht im Voraus bekannt sind. Das Lernverfahren versucht hier in den

[2] Obwohl der mit dem Begriff der Inferenz verbundene Prozess einer Schlussfolgerung deutlich weiter gefasst ist, bezieht sich dieser im Umfeld der Verarbeitung von Kontextdaten vornehmlich auf die Mustererkennung (vergleiche [PZCG14]) und wird im Folgenden entsprechend verwendet.

Eingabewerten Muster zu erkennen und hieraus beispielsweise Klassenzugehörigkeiten abzuleiten. Das so entstandene Modell kann die erlernten Muster dann auf neue, zuvor unbekannte Eingabewerte anwenden.

Regelsysteme Regelsysteme bilden die einfachste Art des Schließens ab, indem sie sich auf *If-then-else*-Konstrukte beschränken. Nach [LD10] stellen sie zudem die populärste Gruppe von Verfahren dar. Regelbasierte Systeme werden dabei oft im Zusammenhang mit ontologischem Schließen verwendet.

Fuzzylogik Fuzzylogik oder unscharfe Logik erlaubt näherungsweise Aussagen und dient der Modellierung von Unsicherheit, um Zustände umgangssprachlich zu beschreiben, und ähnelt dem probabilistischen Schließen. Die Konfidenz drückt in diesem Fall den Grad der Zugehörigkeit zu einer Gruppe aus. Fuzzylogik wird seit längerer Zeit in Kombination mit anderen Verfahren verwendet, um Unsicherheit zu modellieren und so am Beispiel der Temperatur die Zustände „kalt“, „relativ warm“ und „sehr warm“ anstatt einer konkreten Temperaturangabe in Grad Celsius abzubilden.

Ontologisches Schließen Ontologiebasierte Systeme basieren auf der Beschreibung von Zusammenhängen in einer Wissensbasis, wie sie im Zusammenhang mit der ontologiebasierten Modellierung erwähnt wurden. Ontologisches Schließen bietet sich entsprechend vor allem in Zusammenhang mit ontologischer Modellierung von Kontextinformationen an.

Probabilistisches Reasoning Bei diesen Verfahren werden Schlüsse und Entscheidungen auf Basis von Wahrscheinlichkeiten getroffen, die den einzelnen Fakten zugehörig sind. Ein prominentes Beispiel sind versteckte Markov-Ketten, bei denen das Ereignis, auf das es zu schließen gilt, unbekannt ist und nur mithilfe von Beobachtungen und anhand von Wahrscheinlichkeiten Schlüsse abgeleitet werden. Sie kommen insbesondere bei der Aktivitätserkennung (activity recognition) zum Einsatz [SBI+15].

Distribution Die Distribution von Kontextdaten stellt den letzten Schritt innerhalb des Verarbeitungsprozesses von Kontextdaten dar. Die Kontextbereitstellung wird in diesem Schritt oft von Konsumenten oder Empfänger der Kontextdaten aus initiiert. Entsprechend stellt er das Gegenstück zur Kontextakquisition dar (vergleiche Abschnitt 3.2.1) und bietet dieselben Kommunikationsstile an (pull und push).

3.3. Kontextadaption

Neben der Definition dessen, was unter kontextbewusstem Verhalten zu verstehen ist (vergleiche unter anderem die Definition von Dey und Abowd in Abschnitt 3.1.2) existiert eine weitere Definition von Kakousis et al., die sich we-

niger auf die Wahrnehmung und stärker auf den Anpassungsprozess und das Ziel kontextadaptiven Verhaltens bezieht:

> *„Adaptation in ubiquitous computing is understood as the reactive process triggered by a specific event or a set of events in the context, with an ultimate goal to improve the QoS [Quality of Service] perceived by the end-user.“ [KPP10]*

Kontextadaption oder kontextadaptives Verhalten beschreibt nach [KPP10] den Prozess und die Fähigkeit eines Systems, sich an seinen Kontext anzupassen, mit dem Ziel die Dienstqualität für den Endanwender zu erhöhen. Zur Realisierung einer einfachen Kontextadaption wie der Anpassung der Bildschirmhelligkeit an die Lichtverhältnisse der Umgebung reicht es entsprechend eine Teilmenge der im Prozess der Kontextverarbeitung beschriebenen Schritte zu durchlaufen. Es existieren jedoch auch weitaus komplexere Adaptionsszenarien, die weitergehende Konzepte erfordern:

Eine der wesentlichen Basistechnologien zur Realisierung solch eines komplexen adaptiven Verhaltens ist laut [KPP10] das *Autonomic Computing*. Dieser im Jahre 2003 von der Firma IBM erstmalig verwendete Begriff beschreibt ein Paradigma, das sich nach[KC03] aus vier Prinzipien zusammensetzt, die auch als Self-CHOP oder abweichend als Self-Management-Eigenschaften bezeichnet werden und wie folgt beschrieben werden:

- Selbstkonfiguration (self configuration): Die Fähigkeit eines Systems, sich eigenständig zu konfigurieren. Wird beispielsweise eine neue Komponente in ein Gesamtsystem eingebracht, so registriert sich diese zusammen mit ihren Fähigkeiten, sodass andere Teile des Gesamtsystems diese Komponente benutzen oder ihr eigenes Verhalten auf das Vorhandensein dieser neuen Komponente abstimmen.

- Selbstverbesserung (self-optimization): Die Fähigkeit eines Systems, sich selbst zu optimieren. Selbstverbessernde Systeme versuchen proaktiv ihre Funktionsweise als Gesamtsystem zu optimieren, um beispielsweise die Performance des Gesamtsystems zu erhöhen.

- Selbstheilung (self-healing): Die Fähigkeit eines Systems, Fehler zu erkennen, die Ursache zu diagnostizieren und entsprechende kompensierende Maßnahmen zur Fehlerbehandlung zu ergreifen. In diesem Zusammenhang kann ein selbstheilendes System Fehler in der Hardware oder Software mithilfe einer Diagnose-Komponente erkennen und entsprechende kompensierende Strategien auslösen, beispielsweise das Einspielen eines Updates, welches den Fehler behebt.

- Selbstschutz (self-protection): Die Fähigkeit eines Systems, sich vor Angriffen zu schützen und sich selbst proaktiv gegen zukünftige Angriffe durch Anpassung vorzubereiten.

Die vorgenannten Fähigkeiten erlauben so eine Form der Selbstorganisation, die sich nicht nur auf ein Teilsystem, sondern auf das Gesamtsystem bezieht

[KC03]. Diese Selbstorganisation läuft nach [KC03] in Form eines Regelkreises ab, der die Schritte des Monitorings, der Analyse, der Planung und der Ausführung (execute) einer Anpassung mithilfe einer Wissensbasis realisiert und der entsprechend als MAPE-K-Schleife (loop) bezeichnet wird. In [KPP10] wird dieser Regelkreis auf die Problemstellung des kontextabhängigen Adaptionsbedarfes mobiler Geräte übertragen und als Regelkreis der Adaption im Ubiquitous Computing beschrieben, der in Abbildung 3.4 veranschaulicht wird.

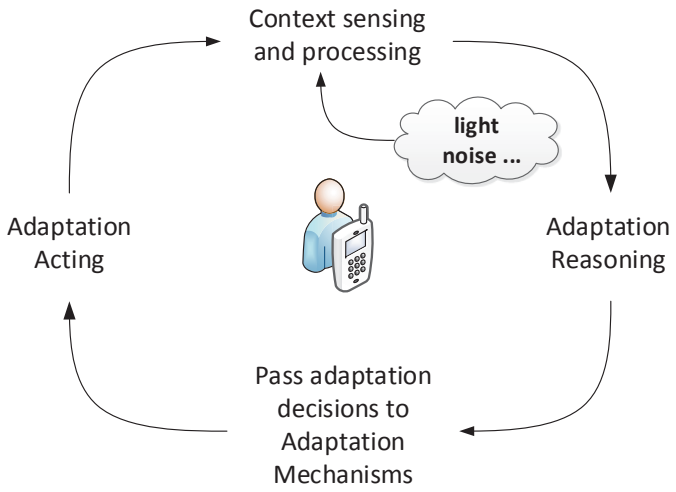

Abbildung 3.4.: Regelkreis der Adaption nach (KPP10)

Eine Adaption besteht laut Kakousis et al. in diesem Regelkreis aus drei wesentlichen Schritten:

- Der Wahrnehmung und Verarbeitung von Kontextdaten, wie sie bereits in den vorhergehenden Abschnitten dieses Kapitels beschrieben wurde. In dieser Phase beobachtet ein System seinen Kontext und verarbeitet diese Daten hin zu höherwertigen Informationen.
- Der Erkennung eines Adaptionsbedarfes: In dieser Phase entscheidet ein selbstadaptives System auf Basis der Kontextinformationen über einen Anpassungsbedarf, der häufig in Form von Aktionen (action based) oder Adaptionszielen (goal based) beschrieben wird [VB14, HSMK09] und eine Adaptionsentscheidung zum Ergebnis hat.
- Die Durchführung der Adaption: In dieser Phase nutzt ein selbstadaptives System einen Adaptionsmechanismus, um die zuvor getroffene Adaptionsentscheidung umzusetzen.

Entsprechend sollen in den folgenden Abschnitten die Erkennung eines Adaptionsbedarfes und die Durchführung der Adaption näher beschrieben werden. Hierzu werden die Gründe beschrieben, warum sich ein System an seinen Kontext anpasst und welche verschiedenen Formen der Adaption hierfür üblicherweise Anwendung finden. Abschließend wird ein kurzer Überblick zu adaptiven Architekturen gegeben.

3.3.1. Adaptionsgründe

Die Gründe dafür, dass sich ein System oder eine Anwendung adaptiv verhält, um sich an seine Umgebung und seinen Nutzungskontext anzupassen, sind vielfältig. Dementsprechend existieren verschiedene Verfahren zur Erkennung eines Adaptionsgrundes und der Planung eines daraus resultierenden Adaptionsbedarfes. In [KPP10] werden hierzu die folgenden vier Kategorien unterschieden:

Aktionsbasierte Anpassung (action-based adaptation) Die aktionsbasierte oder regelbasierte Anpassung stellt nach [KPP10] die prominenteste Kategorie von Verfahren dar, wenn es um die Definition von Verhaltensmustern innerhalb kontextadaptiver Systeme geht. Hier werden Anpassungen eines Systems durch *Zustände* (states) und *Aktionen* (actions) definiert. Ein System in einem Zustand S und die Erfüllung einer Bedingung p löst eine Aktion aus, die das System in einen neuen Zustand S' überführt. Die Realisierung dieses Verhaltens lässt sich nach [KD07] durch eine Menge von Regeln in der Form *IF(Condition) – Then(Action)* definieren, die bestimmen, wie sich das System in einem bestimmten Zustand und damit in einer bestimmten, durch seinen Kontext definierten Situation verhält.

Regelbasierte Systeme stellen damit prinzipiell ein intuitives und einfaches Instrument zur Realisierung von kontextadaptivem Verhalten dar [KPP10]. Zusätzlich ist ihre Auswertung oft auch auf ressourcenbeschränkten mobilen Geräten möglich, da lediglich die Erfüllung der die Aktion auslösenden Bedingung p geprüft werden muss. Die Einfachheit dieses Instruments schränkt jedoch in gleichem Maße ihre Nutzbarkeit innerhalb mobiler Umgebungen ein. Einerseits, da sich der beobachtete Kontext oft ändert, und andererseits, da es für den Entwickler oft schwierig ist alle möglichen Kontextzustände vorherzusehen und entsprechende Regeln zu entwickeln, die diesen Zustand abdecken und eine gewünschte Adaption ermöglichen [WC13].

Zielbasierte Anpassung (goal-based adaptation) Eine weitere Kategorie von Verfahren zur Abbildung von Adaptionsbedarfen bildet die Gruppe der zielbasierten Anpassungen. Anstatt regelbasiert einzelne Kontextzustände zu beschreiben, die eine Adaption auslösen, wird in dieser Kategorie von Adaptionsverfahren ein gewünschter Ziel-Zustand des Systems beschrieben. Dieser Ansatz verlagert die Komplexität der Zielformulierung vom Entwickler in das System, welches hierfür entsprechende Fähigkeiten zum Planen und logischen Schließen vorhalten muss. Konfliktäre und konkurrierende Ziele erschweren alerdings die Nutzung dieses Ansatzes, weswegen von Kephart und Das in [KD07] Nutzenfunktionen als der logische Evolutionsschritt zur zielbasierten Anpassung gesehen werden.

Nutzenfunktionen (utility functions) Als Erweiterung der zielbasierten Anpassung verwenden Nutzenfunktionen Messgrößen zur Bewertung einer Zielerreichung. In diesem Zusammenhang unterstellen sie Kosten der jeweiligen Alternative, um zur Laufzeit zwischen alternativen Adaptionsstrategien die Strategie mit dem maximalen Nutzen ermitteln zu können. So können im Gegensatz zur zielbasierten Anpassung unterschiedliche Ziele gleichzeitig vefolgt werden.

Nutzenfunktionen eignen sich dabei insbesondere im Umfeld des Ubiquitous Computings, da sie für jeden Kontextzustand eine effiziente Bewertung des Nutzens einer jeweiligen Adaptionsstrategie vornehmen können [GBE+09].

Wissensbasierte Anpassung (adaptation reasoning by experience) Neben den bisher vorgestellten drei Kategorien von Verfahren, die allesamt auf (durch den Entwickler) vordefinierten Verhaltensmustern basieren, existiert eine weitere Kategorie, die keinerlei Form der Beschreibung von Adaptionsgründen oder Adaptionszielen bedarf.

Bei der wissensbasierten Anpassung werden die zur Laufzeit gesammelten Kontextinformationen verwendet, um einen Anpassungsbedarf und eine passende Aktion zur Erfüllung des Anpassungsbedarfs zu ermitteln. Der Vorteil dieses Ansatzes liegt laut [KPP10] darin, dass das adaptive Verhalten der Anwendung nicht wie bei den anderen Verfahren zur Entwurfszeit der Anwendung festgelegt wird, sondern sich individuell auf den Kontext des jeweiligen Gerätes oder seiner Nutzer bezieht. Solch ein wissensbasiertes System lernt die Qualität seiner bisherigen Adaptionsentscheidungen und leitet daraus neue Adaptionsentscheidungen ab, die den Nutzen einer mobilen Anwendung für den Anwender maximieren soll. Die Realisierung solcher Systeme erfolgt oft mithilfe von Verfahren des fallbasierten Schließens (case-based reasoning) oder als verstärkendes Lernen (reinforcement learning realisiert), einer Form des maschinellen Lernens. Die richtige Art Verstärkung kann beispielsweise aus vergangenen, erfolgreichen Adaptionsentscheidungen abgeleitet werden. Hierzu gilt es allerdings den Erfolg einer solchen Adaptionsentscheidung anhand der Erfüllung oder Nichterfüllung bestimmter Kriterien zu definieren. Beide Verfahren basieren auf der Annahme, dass ähnlichen Problemen durch ähnliche Lösungen begegnet werden kann. Hierzu nutzt ein entsprechendes System bestehendes Wissen über Kontextzustände und vergangene Adaptionsentscheidungen, um hieraus die passende Adaptionsentscheidung für einen neuen Kontextzustand abzuleiten.

Obwohl diese Verfahren mitunter suboptimale Adaptionsentscheidung liefern und stets eine ausreichend große Historie vergangener Kontextzustände und Adaptionsentscheidungen benötigen, haben sie einen entscheidenden Vorteil. Der Entwickler einer mobilen Anwendungen kann kaum alle zukünftigen Kontextzustände, denen seine Anwendung ausgesetzt sein wird, vorhersehen

[WC13]. Eine wissensbasierte Anpassung kann ihn allerdings von dieser Aufgabe befreien, sofern sie in der Lage ist die Zusammenhänge zwischen dem Kontextzustand und dem Ergebnis einer vergangenen Adaptionsentscheidung zu erkennen und dieses Wissen erfolgreich auf aktuelle und zukünftige Kontextzustände überträgt.

3.3.2. Adaptionsformen

Adaption kann auf eine Reihe von verschiedenen Arten, aber auch auf verschiedenen Ebenen begriffen und durchgeführt werden, entsprechend werden die in diesem Zusammenhang relevantesten Formen der Adaption im Folgenden vorgestellt.

In diesem Zusammenhang kann Adaption zunächst danach unterschieden werden, ob sie statisch oder dynamisch stattfindet. Statische Adaption beschreibt die Anpassung einer Anwendung zur Entwurfszeit [MSKC04]. Das adaptive Verhalten einer solchen Anwendung ist entsprechend auf Anpassungen der Softwarearchitektur beschränkt und wird auch als Entwurfszeit-Adaptivität bezeichnet [Gei08]. Dynamische Adaption hingegen verlagert die Anpassung des Verhaltens einer Anwendung in die Laufzeit derselbigen. Sie bietet entsprechend weitaus bessere Möglichkeiten, auf den häufig wechselnden Kontext mobiler Geräte angemessen reagieren zu können. Diese auch als Laufzeitadaptivität bezeichnete Fähigkeit [Gei08] lässt sich nach [KPP10] weitergehend in Reaktivität und Produktivität unterteilen. Reaktivität bezeichnet in diesem Zusammenhang die Fähigkeit eines Systems, sich im Anschluss an eine Veränderung des Kontextzustandes anzupassen. Obwohl sich ein Großteil existierender kontextbewusster Systeme bereits dynamisch an seinen Nutzungskontext anpassen kann, so beschränkt sich die Anpassung jedoch häufig noch auf einzelne Kontextattribute, fast ausschließlich reaktiv zu arbeiten [May04, Sig08].

Nur sehr wenige kontextadaptive Anwendungen oder Systeme bieten entsprechend eine proaktive Anpassung an ihren (zukünftigen) Nutzungskontext, welche allerdings von Mayrhofer als entscheidendes und wesentliches Kriterium für den erfolgreichen Einsatz dieser Lösungen angesehen wird [May04]. Beispielsweise liefern in einem Auto die Vorhersage von Aquaplaning und die proaktive Reduktion der Geschwindigkeit einen deutlich höheren Mehrwert für den Nutzer als das reaktive Verhalten eines elektronischen Stabilitätsprogramms, welches versucht, den bereits außer Kontrolle geratenen Wagen zu stabilisieren.

Neben der vorgenannten zeitlichen Dimension der Anpassung können die dynamischen Adaptionsformen kontextadaptiver Systeme zusätzlich dahingehend unterschieden werden, mithilfe welcher softwaretechnischer Prinzipien die Adaption realisiert wird. Hierbei wird zwischen der parametrischen und der kompositionellen Adaption unterschieden.

Parametrische Adaption Parametrische Adaption beschreibt die Anpassung des Kontrollflusses einer Anwendung oder eines Systems. Ein bekanntes Beispiel ist die Implementierung der Flusskontrolle innerhalb des Transmission Control Protocol (TCP), in welchen die Größe des Übertragungsfensters an die Last- und Fehlersituation im Netzwerk angepasst wird. In diesem Fall beeinflusst der Zustand der Umgebung, beispielsweise die aktuelle Last im Netzwerk, die internen Parameter einer Anwendung oder eines Systems, in diesem Fall die Größe des Übertragungsfensters.

Parametrische Adaption verändert dabei nicht die Struktur einer Anwendung oder eines Systems, sondern verändert lediglich einzelne Stellgrößen, die in Abhängigkeit von externen Zuständen gesetzt werden. Die Adaptionsmöglichkeiten werden hierbei vom Entwickler zur Entwurfszeit festgelegt [Gei08]. Die fehlende Fähigkeit zur Anpassung der Struktur wird als die wesentliche Einschränkung in Bezug auf den Grad der Adaptionsfähigkeit dieser Adaptionsform angesehen [MSKC04].

Kompositionelle Adaption Im Gegensatz zur parametrischen Adaption erlaubt es die kompositionelle Adaption durch das Austauschen, Hinzufügen und Entfernen von Teilen der Geschäftslogik einer Anwendung zur Laufzeit, um das Verhalten im Hinblick auf den aktuellen Nutzungs- und Ausführungskontext einer Anwendung hin zu optimieren. Hierdurch kann neues Verhalten hinzugefügt oder bestehendes Verhalten angepasst werden, wodurch ein weitaus höherer Grad Adaptionsfähigkeit erreicht wird [MSKC04]. Beispielsweise kann bei einer Anwendung auf einem mobilen Gerät mithilfe der kompositionellen Adaption eine für die Kommunikation verantwortliche Komponente zur Laufzeit ausgetauscht werden, sobald sich die zur Verfügung stehenden Netzwerkverbindungen ändern.

Diese hohe Flexibilität erfordert nach [MSKC04] drei wesentliche Basistechnologien, die im Folgenden jeweils kurz beschrieben werden sollen: einen komponentenbasierten Entwurf (component based design), die Trennung von Geschäftslogik und Adaptionslogik (separation of concerns) und die Fähigkeit zur Introspektion (computational reflection).

Komponentenbasierter Entwurf Die kompositionelle Adaption stellt eine Reihe von Anforderungen an die Softwarearchitektur, die Laufzeitumgebung und den Entwurf einer adaptiven Anwendung [Gei08]. Die grundlegende Fähigkeit zur Adaption wird wie zuvor beschrieben erreicht, indem einzelne Komponenten zur Laufzeit ausgetauscht werden oder zwischen ihnen umgeschaltet wird und hierdurch das Verhalten einer Anwendung beeinflusst wird, ein Ansatz der ursprünglich von Kramer und Magee in [KM85] vorgeschlagen wurde.

Dies erfordert es einerseits, dass die Funktionalität einer adaptiven Anwendung durch den Entwickler in entsprechende Komponenten auf-

[2] Je nachdem ob die eigesetzte Ausführungsumgebung dies unterstützt, sind eine oder beide Varianten denkbar.

geteilt wird, und andererseits, dass das zur Entwicklung verwendete Komponenten-Framework und die zur Ausführung benötigte Laufzeitumgebung das dynamische Entfernen und das Neu-Binden von Komponenten zur Laufzeit erlaubt. Beispiele für entsprechende komponentenbasierten Plattformen sind .NET[3] und J2EE[4].

Trennung von Geschäftslogik und Adaptionslogik Die Trennung des funktionalen Verhaltens der Geschäftslogik von zusätzlichem Programmcode, der für die Adaptionsbedarfe, beispielsweise die Fehlerbehandlung, verantwortlich ist, wird als zweite wichtige Basistechnologie im Zusammenhang mit der kompositionellen Adaption gesehen [MSKC04]. Diese Trennung wird deswegen als vorteilhaft betrachtet, da sich der für die Adaption verantwortliche Programmcode oft über viele Stellen der Geschäftslogik einer Anwendung verteilt und sich mit dieser vermischt, wodurch die Entwicklung und Wartung einer adaptiven Anwendung oft komplex wird.

Ein Weg, diese Trennung zu realisieren, bietet die aspektorientierte Programmierung [KLM+97], die es dem Entwickler ermöglicht, Geschäftslogik und Adaptionslogik getrennt voneinander zu entwickeln, welche erst zur Laufzeit der Anwendung miteinander verbunden werden. Diese Trennung stellt ein wesentliches Unterscheidungsmerkmal zur parametrischen Adaption dar, bei der diese Trennung nicht erfolgt.

Fähigkeit zur Introspektion Diese dritte Basistechnologie der kompositionellen Adaption beschreibt die Fähigkeit einer Anwendung, ihren eigenen Zustand und ihr eigenes Verhalten zur Laufzeit zu beobachten und gegebenenfalls zu verändern. So kann eine Anwendung mithilfe virtueller Sensoren (vergleiche Abschnitt 3.1.3) ihren internen Zustand wahrnehmen und mithilfe einer entsprechenden Adaptionslogik eine Anpassung des Verhaltens durch den Austausch einzelner Komponenten herbeiführen.

Zusammen erlauben die drei genannten Aspekte die Entwicklung von Anwendungen, die sich an Ihren Ausführungskontext anpassen können. Die Realisierung solcher Anwendungen erfordert allerdings die Unterstützung durch eine entsprechende Infrastruktur auf Ebene des Betriebssystems oder einer Middleware, die die entsprechenden Basisfunktionalitäten, beispielsweise die Funktionalität zur Wahrnehmung des Kontextes, zur Ausführung (kontext-) adaptiver Anwendungen bereitstellt.

3.4. Zusammenfassung

Kontextsensitivität bezeichnet die Fähigkeit einer Anwendung oder eines Systems, Informationen über sich selbst oder ihre Umgebung zu beziehen. Sie

[3] https://www.microsoft.com/net
[4] http://docs.oracle.com/javaee/7/index.html

stellt ein entscheidendes Qualitätsmerkmal im Hinblick auf die Nutzung mobiler Geräte dar, da sie es mobilen Anwendungen erlaubt, sich ihres aktuellen Nutzungskontexts durch Auswertung von Kontextinformationen wie Sensordaten „bewusst“ werden zu können und auf Änderungen zu reagieren, ohne dass ein expliziter Eingriff durch den Nutzer notwendig wird. Typische Beispiele für kontextsensitive Anwendungen sind digitale persönliche Assistenten, standortbezogene Dienste, Kommunikationsdienste, Informationsdienste und Navigationsdienste [BRS06].

In diesem Zusammenhang wurden zunächst unterschiedliche Definitionen dessen, was unter dem Begriff des Kontextes zu verstehen ist, diskutiert. Nachfolgend wurde ein etablierter Prozess für die Verarbeitung von Kontextdaten vorgestellt und hierzu die relevanten Aspekte der Akquisition, Modellierung, Auswertung und Distribution von Kontextdaten behandelt. Anschließend wurden die Verwendung von Kontextdaten im Hinblick auf die Realisierung (kontext-) adaptiven Verhaltens von Anwendungen vorgestellt und in diesem Zusammenhang relevante Adaptionsgründe und Adaptionsformen beschrieben.

4. Anforderungsanalyse und existierende Ansätze

Im Anschluss an die Erarbeitung der Grundlagen sollen im folgenden Abschnitt zunächst Klassen von Anwendungsfällen vorgestellt werden, die von einer Kooperation mit ihrer Infrastruktur im Sinne eines mobilen Cloud Computings profitieren können. Auf dieser Grundlage wird eine Anforderungsanalyse durchgeführt, die die Anforderungen an eine mobile Cloud-Kooperation in einem umfassenden Kriterienkatalog abbildet. Auf Basis des entwickelten Kriterienkatalogs werden im Anschluss eine Reihe existierender Ansätze aus verschiedenen Forschungsrichtungen ausgewählt und gegen den Kriterienkatalog evaluiert.

Um ein besseres Verständnis über existierende Ansätze zu erlangen, werden nachfolgend prominente Lösungsansätze tiefergreifend analysiert, die eine hohe Verwandtschaft mit der in dieser Arbeit untersuchten Problemstellung aufweisen.

4.1. Anwendungsfälle

Im Hinblick darauf, dass eine Vielzahl von Forschungsarbeiten als konkretes Anwendungsbeispiel für mobiles Cloud Computing die Gesichts- oder Objekterkennung in Bildern anführt (vergleiche [SMF+12, SMM15, ZLJF12, CLK+11, CBC+10]), soll an dieser Stelle festgehalten werden, dass sich mobiles Cloud Computing bei Weitem nicht auf diesen Anwendungsbereich beschränkt. Entsprechend wird die Analyse verwandter Arbeiten um eine Analyse bereits existierender und zukünftiger mobiler Anwendungen ergänzt, um eine möglichst breite Abdeckung der möglichen Klassen von Anwendungsbeispielen zu erreichen, die von einer Kooperation von mobilen Geräten mit ihrer Infrastruktur profitieren können. Die Anwendungsbereiche sollen dabei anhand ihrer Nutzung, insbesondere der Dauer und der Häufigkeit der Nutzung, der mit der Infrastruktur ausgetauschten Datenmenge sowie der Anforderung zur (echt-) zeitnahen Kooperation mit der Infrastruktur, beschrieben werden.

4.1.1. Verarbeitung von Sensordaten

Der im ersten Abschnitt dieser Arbeit aufgezeigte Trend der stark steigenden Verbreitung mobiler Geräte hat bereits dazu geführt, dass eine Vielzahl von Nutzern mobiler Geräte eine ständig größer werdende Anzahl von Sensoren mit sich führen. Ebenso wurde begonnen, Sensorknoten, wie sie in Abbildung 4.1 veranschaulicht sind, in der Umwelt auszubringen, um bestimmte Umgebungen oder großflächige Umweltzustände zu beobachten.

[1] Quelle: http://www.redcad.org/members/benhalima/azem/images/mica2.jpg

Abbildung 4.1.: Sensorknoten eines drahtlosen Sensornetzwerks[1]

Auf die reine Erfassung von Sensordaten folgen typischerweise eine Reihe weiterer Verarbeitungsschritte, die die Auswertung, Aggregation und Weiterleitung der gewonnenen Informationen umfassen [PJZ+14, PZCG14]. Die oft leistungsbeschränkten Sensorknoten sind allerdings durch ihre begrenzten Rechen- und Speicherressourcen oft nicht in der Lage, diese Verarbeitungsschritte selbstständig durchzuführen. Selbst wenn die vorgenannten Ressourcen ausreichend sind, so ist der begrenzte Energievorrat dieser Geräte oft ein weiteres einschränkendes Kriterium für die lokale Verarbeitung dieser Daten.

Die vor Ort gesammelten Sensordaten werden aus diesem Grund oft zunächst nur weitergeleitet und zentral gespeichert. Alle gesammelten Rohdaten weiterzuleiten, erfordert jedoch nicht nur eine stabile Verbindung mit hoher Bandbreite und setzt insbesondere in Richtung der aggregierenden Netzwerkknoten immer höhere Bandbreiten voraus. Bei Betrachtung von Szenarien wie der (Echtzeit-)Analyse von Umweltdaten, wie sie beispielsweise im Rahmen der Erkennung von Waldbränden, Tsunamis oder Erdbeben benötigt wird, wird offensichtlich, dass eine dezentrale Verarbeitung dieser Daten nicht nur die zu übertragende Datenmenge verringert, sondern auch die beschriebenen Szenarien überhaupt erst umsetzbar macht. Hierzu bieten sich, wie in Abbildung 4.2 veranschaulicht ist, die im Zusammenhang mit dem mobilen Edge Computing vorgestellten Edge Nodes an.

Diese Klasse der Anwendungen zur Verarbeitung von Sensordaten lässt sich entsprechend dadurch charakterisieren, dass stark leistungsbeschränkte Geräte wie Sensorknoten üblicherweise unidirektional und in regelmäßigen Zeitabständen über einen langen Zeitraum hinweg kleinere Datenmengen übertragen. Die Nutzungsdauer ist somit konstant über unter Umständen mehrere Jahre, kann aber auch stärker ereignisorientiert ausgerichtet sein, sodass zu bestimmten Ereignissen unvorhergesehen hohe Datenmengen übertragen werden, was das Vorhalten entsprechender Ressourcen zur Verarbeitung in der Infrastruktur schwieriger vorhersehbar macht.

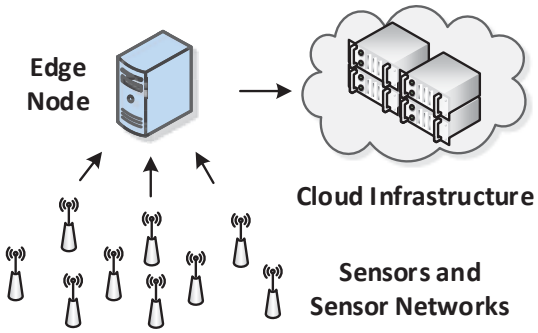

Abbildung 4.2.: Verarbeitung von Sensordaten in mobilen Clouds

Obwohl die übertragene Datenmenge pro Sensorknoten möglicherweise vergleichsweise klein ist, ergeben sich aus der oft großen Anzahl beteiligter Geräte große Datenmenge, die von einer dezentralen Verarbeitung, Aggregation oder Analyse innerhalb einer Edge-Cloud-Infrastruktur profitieren können, vor allem weil sie die Anforderungen an die immer größer werdende Bandbreite hin zur zentralen Infrastruktur entlasten können. Diese Verarbeitung kann, muss jedoch nicht zeitkritisch sein. Beispiele für die Verarbeitung von Sensordaten mithilfe des mobilen Cloud Computings finden sich in [JK11, LML+10, MMR+13].

4.1.2. Analyse von Audiodaten

Mobile Geräte sind durch ihre verbauten Sensoren nicht nur in der Lage, ihre Umgebung mit Sensoren zu erfassen, sondern auf dieser Basis auch eine Reihe von Diensten bereitzustellen, von denen im Folgenden die Klasse von Anwendungsfällen vorgestellt wird, die sich auf die Analyse von Audiodaten bezieht. Hierfür bieten mobile Geräte ein oder mehrere qualitativ hochwertige Mikrofone, die es erlauben, Sprache und Umgebungsgeräusche aufzunehmen und zu verarbeiten. Ein bekannter Anwendungsfall bezieht sich auf den von Microsoft vorgestellten Dienst zur Simultanübersetzung, wie er in der Videokonferenzanwendung Skype[2] umgesetzt wurde. Hierfür wird die Stimme der Sprecher durch eine in Echtzeit berechnete Simultanübersetzung in der Zielsprache des Empfängers ergänzt, die in der Videokonferenz eingebettet als Untertitel angezeigt wird, wie in Abbildung 4.3 zu sehen ist. Ein Anwendungsfall, der einen entsprechenden Übersetzungsdienst in der Infrastruktur benötigt, die für die Übertragung und Vermittlung der Videokonferenz vorhanden ist.

Ein weiterer Anwendungsfall, der, wahrscheinlich um stets betriebsbereit zu sein, nicht auf diese Cloud-Unterstützung zurückgreift, ist der kürzlich vorgestellte ”Pilot“ der in Abbildung 4.4 gezeigt wird: ein Universalübersetzer, der als kleines Gerät in der Ohrmuschel des Nutzers die in der Umgebung des Nutzers hörbare Sprache aufzeichnet und in die Zielsprache des Nutzers übersetzt. Die Ressourcenbeschränkung dieses mobilen Gerätes erlaubt allerdings

[2] http://www.skype.com

Abbildung 4.3.: Simulatübersetzung einer Videokonferenz (Mic16b)

nur eine eingeschränkte Dienstqualität für die Übersetzung. Auch in diesem Fall kann eine mobile Cloud oder Edge-Cloud-Infrastruktur das mobile Gerät entlasten und die Funktionalität zur Übersetzung zusätzlich in einer verbesserten Dienstqualität sicherstellen.

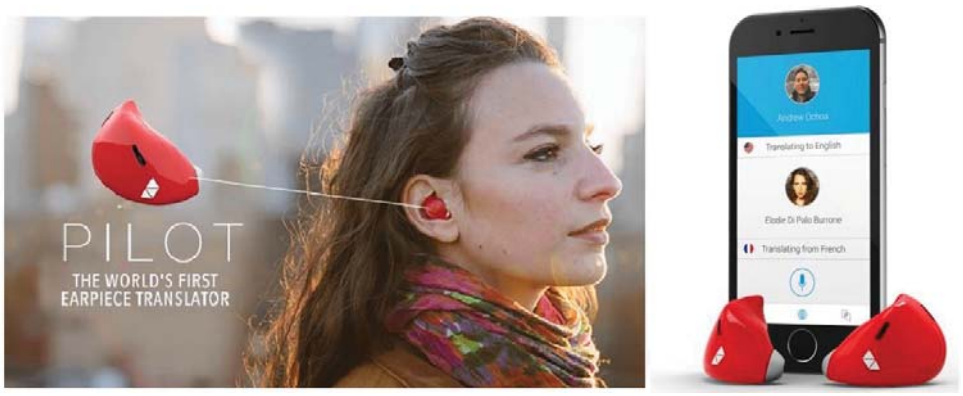

Abbildung 4.4.: Simultanübersetzer „The Pilot" (Ind16)

Diese Gruppe von Anwendungen ist dadurch charakterisiert, dass moderate Datenmengen anfallen, die Verarbeitung dieser Datenmengen aber vergleichsweise rechen- und speicherintensiv ist, weswegen die zu verarbeitenden Daten nach Möglichkeit in die Infrastruktur übertragen werden, um diese dort in entsprechender Qualität zu verarbeiten. Die Verarbeitung ist in den gezeigten Fällen zeitkritisch und latenzsensitiv, sodass bereits geringe Verzögerungen eine subjektive Einschränkung der Dienstqualität aus Nutzerwahrnehmung entstehen lassen kann. In Bezug auf die Datenübertragung werden vornehmlich Daten vom mobilen Gerät in Richtung der Infrastruktur übertragen. Die Nutzungsdauer dieser Dienste variiert dabei üblicherweise zwischen einigen Minuten und einigen Stunden.

4.1.3. Video- und Bildverarbeitung

Das Vorhandensein der in vielen mobilen Geräten verfügbaren Kameras bietet die Basis für eine weitverbreitete Klasse von Anwendungsfällen, der Verarbeitung von Video- und Bilddaten. So erwarten sowohl professionelle Nutzer wie Journalisten als auch private Nutzer, die ihre Inhalte beispielsweise in sozialen Netzwerken zur Verfügung stellen möchten, dass die weitere Bearbeitung ihrer Bild- und Videoinhalte direkt auf dem mobilen Gerät erfolgen kann. Typische Schritte innerhalb dieses Verarbeitungsprozesses können die Reduzierung des Bildrauschens, die Anwendung von Farbfiltern und die Erkennung oder Freistellung von Objekten sein. Zusätzlich existieren eine Reihe weiterer, komplexerer Aufgaben, welche zukünftig von den Nutzern mobiler Geräte vorausgesetzt werden, die sich beispielsweise auf die Gesichts- oder Objekterkennung beziehen, wie durch die Portierung entsprechender Bibliotheken für mobile Geräte zu erkennen ist [Ope16]. Ein weiterer Anwendungsfall betrifft die Erstellung von 360°-Panoramaaufnahmen. Zusammengefasst handelt es sich um Anwendungsfälle, die von einer Auslagerung bestimmter Funktionalitäten einer entsprechenden mobilen Anwendung in die umliegende Infrastruktur profitieren können, um die Ressourcen mobiler Geräte zu entlasten und gleichzeitig die Dienstqualität der mobilen Anwendung zu erhöhen.

In Bezug auf die Videoverarbeitung lassen sich die Ressourcen mobiler Geräte ebenso entlasten. Insbesondere die im Vergleich zur Bildverarbeitung erhöhte Datenmenge kann die begrenzten Ressourcen der Bandbreite eines mobilen Geräts aus- oder überlasten. Im Hinblick auf die Nutzung einer Edge-Cloud-Infrastruktur kann so die Übertragung eines Videodatenstreams in eine zentrale Infrastruktur vermieden werden und erste Verarbeitungsschritte wie die Erkennung bestimmter Muster bereits in einer Edge Cloud realisiert und nur die Ergebnisse weitergegeben oder zurück auf das mobile Gerät transferiert werden. Insbesondere bei der Verarbeitung von Videodaten kann es sinnvoll sein, bestimmte Schritte wie eine initiale Reduzierung der Datenmenge auf einem Surrogate durchzuführen.

Ebenso kann diese kooperative Ausführung auch in Szenarien sinnvoll sein, bei denen durch eine Auslagerung bestimmter Funktionalität in die Infrastruktur zugleich eine höhere Dienstqualität erreicht wird, eine lokale Ausführung im Sinne eines Fallbacks jedoch die Dienstverfügbarkeit aufrechterhält. Ein Beispiel ist die in Abbildung 4.5 gezeigte Anwendung zur Erkennung und Übersetzung von Texten.

Die Anwendungsszenarien der Video- und Bildverarbeitung lassen sich entsprechend dadurch charakterisieren, dass vergleichsweise große Datenmengen anfallen und die Verarbeitung dieser Daten rechen- und speicherintensiv ist. In Bezug auf die Datenübertragung werden dabei vornehmlich Daten vom mobilen Gerät in Richtung der Infrastruktur übertragen. Die Anforderungen im Hinblick auf die Verarbeitung dieser Daten in Echtzeit erscheinen nur bei wenigen Anwendungsfällen gegeben.

Abbildung 4.5.: Bildverarbeitung mit optionaler Cloud-Unterstützung

4.1.4. Mobiles (Cloud) Gaming

Im Bereich des Mobile Gaming existieren bereits eine Reihe von Anbietern und Lösungen, die auf dem Konzept aufbauen, mobile Geräte als Terminals anzusehen, welche die Eingaben in die Infrastruktur übertragen, in der die Geschäftslogik des Spiels ausgeführt wird und die die Bildschirmausgabe in Form eines Streams an das mobile Gerät zur Anzeige transferiert wird. Existierende Lösungen wie Nvidia Shield [Nvi15] nutzen dieses Konzept, um die Ressourcen eines mobilen Gerätes zu entlasten. Jedoch ist die Dienstqualität dieses Konzepts insbesondere von der Latenz der Netzwerkverbindung des mobilen Geräts abhängig. Eine Reihe von Forschungsarbeiten [CCT+11, CWSR14, WD12] fokussieren sich entsprechend auf Lösungen, die die Latenz minimieren. In diesem Zusammenhang werden 120 ms als die maximale Reaktionszeit zwischen der Eingabe von Steuerbefehlen und der Anpassung der Ausgabe angesehen, die zu keiner durch den Nutzer wahrnehmbaren Einschränkung der Dienstqualität führt [MGG+14].

Im Bereich Mobile Gaming ist damit insbesondere eine Konstante und hohe Datenübertragung von der Infrastruktur hin zum mobilen Gerät zu erwarten, bei der in beide Übertragungsrichtungen hohe Anforderungen an die Latenz der Verbindung gestellt werden. Das Nutzungsmuster dieser Anwendungen kann als regelmäßig betrachtet werden und eine Nutzungsdauer von einigen Stunden täglich ist als realistisch anzusehen.

4.1.5. Augmented Reality

Eine weitere Gruppe von Anwendungen ist dem Themengebiet der Augmented Reality zuzuordnen. Ein Beispiel für diese Erweiterung der physischen Realität findet sich beispielsweise im Konzept eines virtuellen Einrichtungsassistenten, der eine Reihe von Bildern eines Raums als Grundlage für eine

3D- Modellierung verwendet, um sowohl ein virtuelles Verschieben bereits real im Raum existierender Möbelstücke als auch ein Hinzufügen neuer Möbelstücke aus dem Katalog eines Händlers erlaubt. Dieses Gesamtergebnis wird anschließend in den von der Kamera des mobilen Geräts aufgenommenen Videodatenstrom integriert, vergleiche Abbildung 2.5 aus Abschnitt 2.2.1. Beispiele für eine konkrete technische Umsetzung finden sich im Projekt Irides [Mic15], in dem die rechenintensive Aufbereitung der grafischen Inhalte auf einem entsprechend leistungsfähigen Surrogate erfolgt, wie in Abbildung 4.6 veranschaulicht wird.

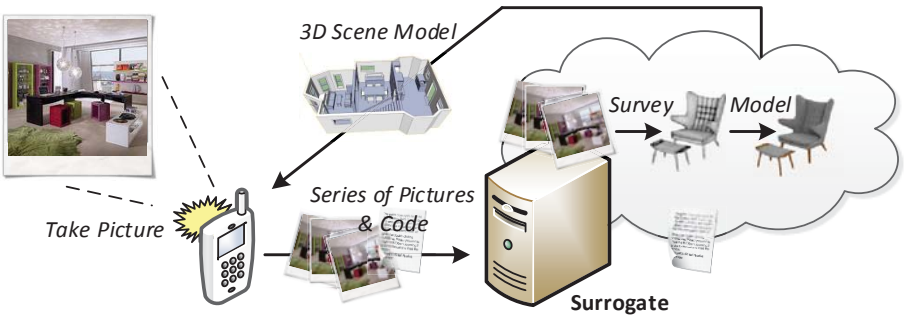

Abbildung 4.6.: Augmented Reality: Einrichtungsassistent

Charakteristisch für diese Klasse von Anwendungsfällen ist die Vereinigung der Anforderungen aus den beiden vorgenannten Kategorien des mobilen Cloud Gamings und der Video- und Bildverarbeitung, ergänzt um noch zusätzlich erhöhte Anforderungen in Bezug auf die Latenz. Das Nutzungsmuster kann im Bereich von wenigen Minuten bis einigen Stunden liegen. Die Periodizität kann je nach Anwendungsfall stark variieren.

4.1.6. Portierung existierender Anwendungen

Die aufgeführten Klassen von Anwendungsfällen stellen nur einen Ausschnitt der Einsatzmöglichkeiten einer kontextadaptiven Anwendungsarchitektur für das mobile Cloud Computing dar. Im praktischen Einsatz sind dabei eine Reihe weiterer Anwendungsdomänen denkbar, die die Portierung klassischer Anwendungen für stationäre Geräte auf mobile Geräte umfasst. Im Fokus unterschiedlicher Forschungsarbeiten wie [LLS11, WSW13, TS12, ZTC14] stehen dabei oft Webbrowser. Hier wird eine leistungsstarke Infrastruktur genutzt, die Inhalte insoweit für ein mobiles Gerät aufzubereiten, als diese effizient dorthin übermittelt und zur Anzeige gebracht werden können.

4.2. Anforderungsanalyse

Mit Blick auf die zuvor vorgestellten Klassen von Anwendungsfällen kann festgehalten werden, dass existierende Lösungen, die die Marktreife erreicht haben, oft proprietärer sind und keine von ihnen auf eine allgemeine kontextadaptive Anwendungsarchitektur oder eine entsprechende Systemunterstützung

aufsetzen. Entsprechend sind Entwickler von mobilen Anwendungen, die die Potenziale des mobilen Cloud Computings nutzen, nicht nur auf Fähigkeiten im Umgang mit den klassischen Problemstellungen des Mobile Computings angewiesen. Ebenso müssen diese Entwickler zur Realisierung entsprechender kontextadaptiver mobiler Anwendungen sowohl mit den klassischen Problemstellungen verteilter Systeme als auch den Problemstellungen und Lösungsansätzen im Bereich kontextadaptiver Systeme vertraut sein.

Dies lässt eine angemessene Unterstützung der Entwickler unumgänglich erscheinen, wenn es darum geht, die Potenziale kontextadaptiver Anwendungsarchitekturen für die breite Masse der Entwickler mobiler Anwendungen nutzbar zu machen. Diese Unterstützung betrifft damit einerseits den Entwicklungsprozess hin zu einer adaptionsfähigen Architektur mobiler Anwendungen und andererseits die Bereitstellung einer entsprechenden Systemunterstützung für die kontextadaptive Ausführung dieser Anwendungen, die im Folgenden unter den Begriff einer *Architektur für mobiles Cloud Computing* verstanden wird. Die Anforderungen an eine solche Architektur sollen im nun folgenden Abschnitt ermittelt und vorgestellt werden.

4.2.1. Ermittlung der Anforderungen

Das Gebiet des mobilen Cloud Computings verbindet das Forschungsgebiet der verteilten Systeme mit dem Forschungsgebiet der kontextbewussten Systeme. Entsprechend ist es das Ziel des hier entwickelten Anforderungskataloges auf die speziellen sich aus dieser Kombination ergebenden Problemstellungen einzugehen. Der im folgenden vorgestellte Kriterienkatalog geht entsprechend explizit auf die speziellen Anforderungen an Anwendungen im Bereich des mobilen Cloud Computings ein, um die Entwickler entsprechender Anwendungen bei der Erstellung eigener Lösungen zu unterstützten. Hierzu betrachtet er beispielsweise das Zusammenwirken mobiler Geräte mit ihrer Infrastruktur. Dieses erfolgt allerdings weder in Form eines klassischen Client-Server-Modells, noch in Form eines Peer-to-Peer-basierten Ansatzes, woraus sich spezielle Anforderungen im Hinblick auf die Robustheit und Fehlertoleranz einer solchen Architektur ergeben.

Aus Sicht der Forschungsmethodik erfolgt hierzu eine strukturierte Analyse verwandter Arbeiten, bei der die aus den relevanten Forschungsarbeiten erwähnten konkreten, lösungsspezifischen qualitativen und quantitativen Kriterien für die Erfolgsbewertung auf übergeordnete und technologieneutrale Anforderungen hin abstrahiert werden. Ergänzend zu diesen für das Mobile Cloud Computing spezifischen Anforderungen werden zusätzlich die allgemeinen Kriterien für Softwarequalität berücksichtigt und in Form der ISO/IEC 25010 [Int11] Kriterien herangezogen, um sicherzustellen, dass die Anforderungen den dort genannten Kriterien der Zuverlässigkeit, Nutzbarkeit, Effizienz, Wartbarkeit und Portierbarkeit entsprechen. Zur Vereinigung dieser zwei unterschiedlichen Kriterienkataloge werden die aus den verwandten Arbeiten ermittelten konkreten Anforderungen in den für die Problemstellungen des

mobilen Cloud Computings angepassten ISO-/IEC-Kriterienkatalog eingeordnet. Das Ergebnis umfasst einen Anforderungskatalog der die Bereiche Verfügbarkeit, Portierbarkeit, Skalierbarkeit, Nutzbarkeit, Wartbarkeit und Sicherheit abdeckt, die im Folgenden nacheinander vorgestellt werden. Als erklärendes Beispiel und stellvertretend für die verschiedenen Nutzungsszenarien im mobilen Cloud Computing soll in der folgenden Anforderungsanalyse der Einfachheit halber ein repräsentativer Anwendungsfall, die Auslagerung einer Funktionalität von einem mobilen Gerät auf ein Surrogate, als beschreibendes Beispiel dienen.

4.2.2. Verfügbarkeit

Idealerweise sollte mobiles Cloud Computing auf Basis einer zuverlässigen und stabilen Verbindung mit geringer Latenz und hoher Bandbreite zwischen allen beteiligten Geräten stattfinden. Zusätzlich sollten diese Eigenschaften sich im Verlaufe der Nutzung nur unwesentlich ändern und die Verbindungen eine entsprechend hohe Zuverlässigkeit aufweisen. Werden hierzu allerdings die im Mobile Computing überwiegend vorherrschenden drahtlosen Kommunikationsverbindungen genutzt, so zeigt sich, dass diese Anforderungen nur bedingt oder mit starken Einschränkungen umsetzbar sind. Die Mobilität der Nutzer bringt somit weitere Einschränkungen in Bezug auf die Qualität der Bandbreite, Latenz und Verfügbarkeit einer Verbindung zur Infrastruktur und zu weiteren mobilen Geräten mit sich. Um den aufgezeigten Problemen zu begegnen, wird versucht, mobile Clouds in ihrer Ausdehnung zu beschränken und hierzu die Verbindung zwischen einem mobilen Gerät, als Quelle der Auslagerung, und seines Surrogates, als Ziel der Auslagerung, aufrechtzuerhalten und hierfür direkte Verbindungen zwischen den beteiligten Geräten wie Ad-hoc-Netzwerken zu nutzen. Ebenso wird versucht, die verschiedenen, verfügbaren drahtlosen Schnittstellen für die Mobilkommunikation simultan zu nutzen [YHK+12, HHY15, HAHZ15]. Diese Verfahren bringen allerdings eine erhöhte Komplexität in Bezug auf die Adressierung und die Verteilung der Datentransfers auf die einzelnen Schnittstellen mit sich. Zusätzlich lässt sich diese Art der abwechselnden oder simultanen Kommunikation über verschiedene Schnittstellen nicht transparent durch den *Internet Protocol (IP)-Layer* verbergen, wodurch eine transparente Nutzung durch Entwickler ein zusätzliches *Overlay-Netzwerk* erforderlich macht, um die zusätzliche Komplexität vor dem Entwickler zu verbergen. Durch die gezeigten Ansätze lassen sich die Auswirkungen der wechselnden Qualität drahtloser Kommunikationsverbindungen zwar reduzieren, jedoch keinesfalls ganz kompensieren. Hieraus ergibt sich die Anforderung an eine angemessene Architektur, diese Einschränkungen, beispielsweise einen Verbindungsabbruch[3], gegenüber einer mobilen Anwendung nicht sichtbar zu machen, sondern entsprechende Mechanismen zur Fehlerbe-

[3] Wie in Kapitel 2 beschrieben, wird die Konnektivität stationärer Geräte innerhalb eines Netzwerks untereinander generell als stabil angesehen. Mobile Geräte hingegen unterliegen einer häufig wechselnden Konnektivität ihrer Verbindung zur Infrastruktur und zu weiteren mobilen Geräten.

handlung bereitzustellen und somit gegenüber einer mobilen Anwendung eine transparente Fehlerbehandlung durchzuführen. Da solch ein Verbindungsabbruch in einer mobilen Cloud üblicherweise ein Fehlen einer benötigten Funktionalität nach sich zieht, sollte eine Architektur für mobiles Cloud Computing diesem Fehler durch eine Replikation der Funktionalität auf dem mobilen Gerät begegnen (A1).

Ein Rückgriff (fallback) auf die Funktionalität auf dem mobilen Gerät sollte jedoch nicht die einzige Strategie sein. Die Suche nach weiteren gleichwertigen Surrogates und eine sinnvolle Verteilung (load balancing) der Geschäftslogik einer mobilen Anwendung zwischen diesen sollten von einer entsprechenden Architektur für mobiles Cloud Computing umgesetzt werden, um beispielsweise eine Überlastung bestimmter Surrogates zu vermeiden. In keinem Fall sollte dabei der fachliche Kontrollfluss der mobilen Anwendung, also die Geschäftslogik, hierdurch beeinflusst werden (A2). Weitere Adaptionsstrategien, um die Nutzbarkeit und Verfügbarkeit einer mobilen Anwendung zu erhalten, können bei zeitlich unkritischer Funktionalität auch die Verzögerung der Ausführung dieser Funktionalität umfassen, bis erneut ein Ausführungskontext vorhanden ist, der eine erfolgreiche Ausführung erwarten lässt. Hierzu bedarf es einer Vorhersage dieses zukünftigen Ausführungskontextes, insbesondere im Hinblick auf die Konnektivität zu den aktuell und möglicherweise zukünftig benötigten Ressourcen (A3).

Insbesondere wenn ressourcenbeschränkte Geräte als Teil der mobilen Cloud oder eines mobilen Ad-hoc-Netzwerkes als Surrogate auftreten und damit Ziel einer Auslagerung von Berechnungen werden, gilt es, mit diesen Ressourcen effizient umzugehen und mit der sowohl erfolgreichen als auch nicht erfolgreichen Ausführung von Funktionalität möglichst nicht unnötig lange die Ressourcen dieser Surrogates zu belasten. Dies umfasst auch Ausführungsergebnisse, die wahrscheinlich nicht oder nicht rechtzeitig an die Quelle der Auslagerung zurückgesendet werden können. Hierbei gilt es allerdings zu beachten, dass insbesondere energetisch oder zeitlich teuer ermittelte Ergebnisse entsprechend lange vorgehalten werden, um zu vermeiden, dass diese erneut auf anderen Surrogates oder dem auslagernden mobilen Gerät selbst erneut erzeugt werden müssen (A4).

- A1 Fehlerbehandlung: Verbindungsabbrüche und Übertragungsfehler zwischen mobilen Geräten und Surrogate sollten abgefangen und eine lokale Ausführung der ausgelagerten Funktionalität initiiert werden.

- A2 Erweitertes Fehlermanagement: Verbindungsabbrüche zu und Überlastsituationen auf Surrogates sollten erkannt und durch die Architektur für mobiles Cloud Computing kompensiert werden, ohne dass der fachliche Kontrollfluss der mobilen Anwendung beeinflusst wird.

- A3 Prognose der Konnektivität: Zukünftige Verbindungen zu Surrogates und deren Qualität sollten in einer Adaptionsstrategie und in der zukünf-

tigen Auslagerung von Funktionalität auf Surrogates Berücksichtigung finden.

- A4 Effizienz: Verbindungsabbrüche zu Surrogates sollten nicht zu blockierten Ressourcen, die den begrenzten Speicher blockieren oder weiteren negativen Seiteneffekten führen.

4.2.3. Portierbarkeit

Die Entscheidung über die Auslagerung einer bestimmten Funktionalität wird als eine der wesentlichen Problemstellungen im mobilen Cloud Computing angesehen [OMK+14, LAS+15]. Um die Kooperation zwischen verschiedenen mobilen und stationären Geräten zu ermöglichen, bedarf es der Fähigkeit zur Verschiebung bestimmter Funktionalitäten einer mobilen Anwendung von einem mobilen Gerät auf seine Surrogates. Hierfür wiederum ist es notwendig, eine mobile Anwendung so zu entwickeln, dass sich ihre Funktionalität überhaupt verteilen lässt. Diese Verteilbarkeit erfordert die Partitionierung der Funktionalität einer mobilen Anwendung, eine Anforderung, die als die generelle Auslagerungsfähigkeit (P1) beschrieben wird.

Eine gute Adaptionsstrategie wie die im Beispiel genannte Auslagerungsentscheidung ist allerdings stark vom aktuellen Ausführungskontext abhängig [OXK+15] und sollte entsprechend auf Basis von Kontextattributen, beispielsweise der verfügbaren Bandbreite, des zu erwartenden Energieverbrauchs, der Leistungsfähigkeit der Surrogates und weiterer Parameter, getroffen werden. Hierfür bedarf es einer Kostenfunktion, die im Hinblick auf das oder die Adaptionsziele den zu erwartenden Nutzen oder Vorteil gegenüber weiteren Adaptionsstrategien ermittelt. Die hierfür notwendige Ermittlung der relevanten Parameter wie Kontextinformationen und die sich daran anschließende Berechnung der Kostenfunktion einer mobilen Anwendung kann allerdings ebenfalls eine ressourcen- und insbesondere rechenintensive Aufgabe darstellen. Dies macht es erforderlich, (Optimierungs-)Routinen möglichst effizient zu gestalten und gegebenenfalls eine gute Approximation, wie in [WZL16] und [LAS+15] gezeigt, zu verwenden, um die durch die Adaptionsstrategie zu erwartende Ressourceneinsparung nicht durch die Ermittlung und Beurteilung derselbigen obsolet zu machen. Dies erfordert es, wenn das gewählte Adaptionsziel eine Zeiteinsparung umfasst, ebenso die für die Ermittlung der Adaptionsstrategie notwendige Rechenzeit zu minimieren. Diese Unterstützung durch eine kontextadaptive Anwendungsarchitektur wird als die grundlegende Fähigkeit zur effizienten Kontextadaption beschrieben (P2).

Erweiterte Verfahren zur Beurteilung einer Adaptionsstrategie wie in [CIM+11] verlagern den vom aktuellen Ausführungskontext unabhängigen Teil der Ermittlung geeigneter Adaptionsstrategien von der Laufzeit der mobilen Anwendung (runtime) in die Entwicklungszeit der mobilen Anwendung (compile time) und kombinieren diese Information mit dem kontextabhängigen Teil der Beurteilung einer Kostenfunktion, wodurch sich die Zeitdauer für die Beurteilung der Adaptionsstrategie zur Laufzeit deutlich verkürzt. Im Zusam-

menhang mit dem Anwendungsbeispiel der Auslagerung ergibt sich hierdurch eine deutlich größere Anzahl von Situationen, in denen es vorteilhaft ist, eine Funktionalität einer mobilen Anwendung, im Hinblick auf das Adaptionsziel einer Zeitersparnis, in die Infrastruktur auszulagern. Wird in diesem Zusammenhang von einer kontextadaptiven Anwendungsarchitektur nicht nur der aktuelle, sondern mithilfe von Kontextprognosen (vergleiche A3) oder Mobility Models[4] auch ein zukünftiger Ausführungskontext antizipiert, so wird dies als die erweiterte Adaptionsfähigkeit beschrieben (P3).

Ein weiterer Aspekt der Portabilität einer kontextadaptiven Anwendungsarchitektur für mobiles Cloud Computing betrifft die Fähigkeit zur Koexistenz mit weiteren Anwendungen auf demselben mobilen Gerät. Für diese Fähigkeit zur Koexistenz ist es erforderlich, dass eine faire Ressourcenteilung zwischen den laufenden Anwendungen ermöglicht wird. Diese faire Ressourcenteilung gilt es gleichzeitig auf die Verteilung von ausgelagerten Aufgaben zwischen den beteiligten Surrogates auszudehnen (P4).

Ein weiterer Aspekt betrifft das Deployment einer mobilen Anwendung auf die an der kooperativen Ausführung beteiligten Surrogates. Übliche Technologien für die entfernte Ausführung von Diensten wie entfernte Methodenaufrufen (remote method invocation) sind hierfür generell nur stark eingeschränkt nutzbar, da sie erfordern, dass der für die ausgelagerte Funktionalität auszuführende Code bereits auf dem jeweiligen Surrogate vorhanden ist. Im mobilen Cloud Computing ist hingegen ebenso die spontane Kooperation erforderlich, um in möglichst viele Situationen, in denen eine Kooperation möglich und vorteilhaft ist, diese auch nutzen zu können. Dies erfordert es, den für die Ausführung einer Funktionalität einer mobilen Anwendung auf einem Surrogate benötigten Programmcode zur Laufzeit der mobilen Anwendung und gegebenenfalls erst direkt vor der Ausführung der Funktionalität auf einem Surrogate bereitzustellen (P5).

Letztes wesentliches Kriterium für die Portabilität einer kontextadaptiven Anwendungsarchitektur ist die Offenheit, die es ermöglichen sollte, mit unterschiedlichen Dienstanbietern und unterschiedlichen Typen von Surrogates interagieren und kooperieren zu können, um Abhängigkeiten (vendor lock-in) zu bestimmten Dienstanbieter zu verhindern, was die Unterstützung offener Standards und Protokolle erforderlich macht (P6).

- P1 Generelle Auslagerungsfähigkeit: Die Fähigkeit, bestimmte Teile des Codes und der Daten einer Anwendung auf Surrogates auszuführen, um die Ressourcen eines mobilen Geräts dabei zu entlasten.
- P2 Grundlegende Adaptionsfähigkeit: Die Fähigkeit sich, unter Berücksichtigung des aktuellen Kontexts, an unterschiedliche Umgebungen im

[4] Ein Mobility Model repräsentiert die Bewegung eines Nutzers und seinen Ortswechsel im Verlauf der Zeit. Es kann genutzt werden, um zukünftige Aufenthaltsorte vorherzusagen, und erlaubt es damit, auf zukünftige Kontextzustände wie die Konnektivität eines mobilen Geräts zu schließen [BH04].

[4] Insbesondere bei der opportunistischen Nutzung von Ressourcen kann nicht davon ausgegangen werden, dass der zur Ausführung benötigte Code bereits auf dem Surrogate verfügbar ist.

Hinblick auf ein Adaptionsziel anzupassen und hierzu beispielsweise die Verlagerung oder Anpassung bestimmter Funktionalitäten einer mobilen Anwendung durchzuführen.

- P3 Erweiterte Adaptionsfähigkeit: Um fortgeschrittene Adaptionsstrategien zu ermöglichen, sollten historische, aktuelle und mögliche zukünftige Kontextzustände in der Auslagerungsentscheidung berücksichtigt werden, um dem schnell wechselnden Kontext in mobilen Umgebungen Rechnung zu tragen.

- P4 Koexistenz: Das Konzept der fairen Ressourcenaufteilung auf mobilen Geräten erfordert bei einer verteilten Ausführung die Erweiterung dieser fairen Ressourcenaufteilung auf die Surrogates, sodass diese mit anderen Anwendungen, gegebenenfalls auch mit solchen anderer Nutzer, sinnvoll koexistieren können und keine exklusive Verfügbarkeit von Ressourcen voraussetzen.

- P5 Deployment: Die Installation einer Anwendung auf einem mobilen Gerät sollte sich in ihrer Komplexität nicht von der Installation einer beliebigen mobilen, nicht für die verteilte Ausführung entwickelten, Anwendung unterscheiden. Zur Unterstützung der spontanen Kooperation ist es erforderlich, dass das Deployment der Anwendung auf einem Surrogate zur Laufzeit der Anwendung und direkt vor der Auslagerung einer Funktionalität ermöglicht wird.

- P6 Offenheit: Eine kontextadaptive Anwendungsarchitektur sollte in der Lage sein, mit verschiedenen Dienstanbietern und verschiedenen Typen von Surrogates durch offene Protokolle und Standards interagieren zu können.

4.2.4. Skalierbarkeit

Um die verteilte Ausführung von kontextadaptiven Anwendungen im mobilen Cloud Computing zu ermöglichen, ist es für eine entsprechende Architektur erforderlich, ein möglichst großes Spektrum von Surrogates zu unterstützen und sich an die Heterogenität dieser Surrogates möglichst gut anzupassen (vergleiche insbesondere P1). Ebenso ist es für eine entsprechende Architektur relevant, sich an den aktuellen und zukünftigen Ausführungskontext anzupassen (vergleiche P2 und P3). Zusammen mit weiteren Aufgaben für die Koordination der Auslagerung, zum Beispiel die regelmäßige Suche nach verfügbaren Surrogates, kann ein signifikanter Overhead entstehen, welcher eine zusätzliche Belastung der begrenzten Rechen-, Energie- und Bandbreitenressourcen eines mobilen Gerätes zur Folge hat. Der hierdurch entstehende Overhead einer kontextadaptiven Anwendungsarchitektur sollte entsprechend möglichst gering ausfallen. Es gilt hierbei also den zusätzlichen Rechenaufwand für die Partitionierung und die Auslagerungsentscheidung zu minimieren (S1).

Wenn neue Surrogates gefunden werden, müssen diese nicht nur mit dem notwendigen Programmcode für die Ausführung einer vom mobilen Gerät ausgelagerten Funktionalität versorgt werden (siehe P5), sondern es ist ebenso erforderlich, diese möglichst schnell in die verteilte Ausführung zu integrieren. Es gilt dazu, einerseits die gewählte Adaptionsstrategie erneut zu evaluieren und andererseits den Workload in Form der auszulagernden Funktionalität entsprechend der Auslagerungsstrategie zu verteilen, um eventuell vorhandene Überlastsituationen bestimmter Surrogates zu kompensieren. Ebenso gilt es hierbei, Surrogates mit speziellen Ressourcen, beispielsweise Sensoren, zu berücksichtigen und dies im Rahmen der Dienstsuche über Mechanismen zu unterstützen – wie beispielsweise Mechanismen zur Dienstsuche, die die Suche nach nichtfunktionalen Kriterien unterstützen (S2).

Ein weiterer Vorteil in der Nutzung von leistungsfähigen Surrogates besteht in der Möglichkeit der parallelen Ausführung dieser Funktionalität auf mehreren Surrogates. Die parallele Ausführung von nicht für die nebenläufigen Ausführungen geschriebenem Programmcode stellt jedoch ein generelles Problem dar und ist nur in den wenigsten Fällen problemlos möglich. Lediglich im Bereich von Kontrollstrukturen, die der iterativen Verarbeitung von Daten dienen, beispielsweise der „parallelfor“-Schleife in der Parallel Patterns Library[5] der Firma Microsoft, ist solch eine automatisierte Parallelverarbeitung denkbar. Dementsprechend ist es für eine kontextadaptive Anwendungsarchitektur von großem Vorteil, dass sie dem Entwickler entsprechende Strukturen vorgibt, um die nebenläufigen Ausführungen von Funktionalität zu begünstigen (S3).

Ein weiteres Kriterium für eine gute Skalierbarkeit einer kontextadaptiven Anwendungsarchitektur stellt die effiziente Verwaltung der Ressourcen des mobilen Geräts und auch der Surrogates in den Mittelpunkt. So erscheinen die Ressourcen der Cloud Service Provider zwar unerschöpflich, im Bereich des mobilen Cloud Computings gilt es jedoch gleichzeitig, mit den ebenfalls begrenzten Ressourcen bestimmter Surrogates wie Cloudlets möglichst effizient umzugehen und beispielsweise die Taskausführung zu pausieren und zu reaktivieren (S4).

- S1 Overhead: Mobile Anwendungen sollten, wenn sie lokal ausgeführt werden, eine mit konventionellen Anwendungen gleichen Typs vergleichbare Geschwindigkeit und Ressourcenbelastung aufweisen. Der Overhead der kontextadaptiven Architektur sollte somit auf dem mobilen Gerät sowie auch auf den Surrogates möglichst gering sein.
- S2 Dienstsuche und Integration: Eine kontextadaptive Anwendungsarchitektur sollte in der Lage sein, neue Ressourcen möglichst schnell zu finden und zu integrieren. Hierbei sollte eine Funktionalität einer mobilen Anwendung schnell auf die aktuell verfügbaren Ressourcen, entsprechend der gewählten Adaptionsstrategie, verteilt werden, um die opportunistische Nutzung von Ressourcen zu ermöglichen.

[5] https://msdn.microsoft.com/de-de/library/dd728073.aspx

- S3 Parallelisierung: Die parallele Ausführung durch die Verteilung einer mobilen Anwendung auf mehreren Surrogates sollte durch eine kontextadaptive Anwendungsarchitektur mithilfe etablierter Mechanismen für die nebenläufigen Ausführungen wie Threads unterstützt werden.
- S4 Effizienz: Die Möglichkeit zum Pausieren von ausgelagerter Funktionalität auf Surrogates ist erforderlich, um eine Verfügbarkeit des Surrogates auch bei hoher Auslastung desselbigen zu ermöglichen. Gleichzeitig sollte die Reaktivierung dieser Funktionalität möglichst schnell erfolgen, um auch Anwendungsfälle zu unterstützen, die eine geringere Latenz erfordern.

4.2.5. Nutzbarkeit

Aktuell existieren keine etablierten Architekturen, keine Standards und auch keine entsprechenden Infrastrukturen, die als Referenzimplementierung für die Entwicklung einer kontextadaptiven Anwendungsarchitektur und zur Unterstützung einer Vielzahl mobiler Anwendungen herangezogen werden können [SAGB14]. Da lediglich erste Standardisierungsbemühungen zu erkennen sind (vergleiche [ETS14, ETS15]), liegen für das Design und die Implementierung von Architekturen für das mobile Cloud Computing noch keine Erfahrungswerte vor. Um allerdings das Potenzial dieses Konzepts den Entwicklern mobiler Anwendungen zu erschließen, erscheint es sinnvoll, die zusätzlichen Anforderungen an die Entwickler, in Form spezifischer Kenntnisse über die Eigenheiten verteilter Systeme und domänenspezifischer Restriktionen und Problemstellungen des mobilen Cloud Computings, möglichst gering zu halten (U1).

Aktuelle Lösungen wie die Google AppEngine[6], die das Konzept des Platform as a Service (vergleiche Abschnitt 2.1.2) umsetzen, sind für eine spontane und opportunistische Interaktion mit mobilen Geräten nicht ausgelegt und stellen hier keine sinnvolle Unterstützung dar. Die Nutzung eines solchen PaaS-Dienstes stellt ebenso zusätzliche Anforderungen an das Wissen und die Fähigkeiten eines Entwicklers mobiler Anwendungen. Dementsprechend sollte eine angemessene Anwendungsarchitektur für mobiles Cloud Computing eine geeignete Abstraktion zu den domänenspezifischen Problemstellungen des mobilen Cloud Computings wie eine wechselnde Verbindungsqualität durch entsprechende Transparenzeigenschaften bereitstellen (U2).

Zusätzlich sollte sich eine kontextadaptive Anwendungsarchitektur in die üblicherweise für die Entwicklung mobiler Anwendungen genutzten Methoden und Werkzeuge integrieren, um einen einfachen und ganzheitlichen Entwicklungsprozess sicherzustellen (U3). Ebenso sollte sich die Konfiguration einer entsprechenden Architektur möglichst auf die Auswahl verständlicher Adaptionsziele beschränken, um hierdurch aus Sicht der Endanwender verständliche Optimierungsziele, zum Beispiel die Einsparung von Energie oder die Beschleunigung bestimmter Anwendungen, konfigurieren zu können (U4).

[6] https://cloud.google.com/appengine/

- U1 Einfache Nutzbarkeit: Das Design und die Implementierung einer Anwendung für das mobile Cloud Computing sollte möglichst einfach und intuitiv ablaufen. Domänenspezifische Kenntnisse sollten nicht notwendig sein und sich auf die allgemeinen Grundlagen der Entwicklung verteilter Systeme beschränken.
- U2 Angemessene Abstraktion: Eine kontextadaptive Anwendungsarchitektur sollte Transparenzeigenschaften anbieten, um eine angemessene Abstraktion von den spezifischen Problemstellungen des mobilen Cloud Computings zu liefern. Insbesondere sollten Details der Kommunikation durch eine geeignete Form der Verteilungstransparenz vor dem Entwickler verborgen werden.
- U3 Unterstützung von Entwicklungsstandards: Die Entwicklung mobiler Anwendungen auf Basis einer kontextadaptiven Anwendungsarchitektur sollte sich in die üblichen Verfahren und Werkzeuge für die Entwicklung mobiler Anwendungen integrieren.
- U4 Automatische Konfiguration: Die Anforderungen an die Konfiguration durch Endanwender sollten auf verständliche Optimierungsziele wie die Einsparung bestimmter Ressourcen, beispielsweise Energie oder Bandbreite, beschränkt sein.

4.2.6. Wartbarkeit

Ein zeitgemäßes Verständnis dessen, was unter einer erfolgreichen und nachhaltigen Softwareentwicklungsmethodik zu verstehen ist, bringt fast zwangsläufig Schlagwörter wie agile Methoden und einen evolutionären und iterativen Entwicklungsprozess mit sich. Um diesen Entwicklungsansatz angemessen zu unterstützen, ist es erforderlich, dass sich eine kontextadaptive Anwendungsarchitektur für mobiles Cloud Computing in diese Prozesse integriert und dabei die Komplexität der Wartung, Erweiterung, der Restrukturierung (refactoring) oder des Testens einer mobilen Anwendung nicht übermäßig erhöht. Beispielsweise sollte die Auslagerung einer Funktionalität stets zu denselben Ergebnissen wie die lokale Ausführung führen und sich lediglich in der Ausführungszeit von der lokalen Ausführung unterscheiden. Dies macht es erforderlich, dass der fachliche Kontrollfluss einer mobilen Anwendung nicht durch die Adaptionsstrategie einer kontextadaptiven Anwendungsarchitektur beeinflusst wird, unabhängig davon ob und welche Surrogates Verwendung finden (M1).

Hiervon abweichend ist festzuhalten, dass es kontextabhängig in bestimmten Fällen auch sinnvoll sein kann, dasselbe Ergebnis mithilfe eines alternativen Algorithmus zu ermitteln; beispielsweise kann ein rechenintensiver gegen einen speicherintensiven Algorithmus ausgetauscht werden, um die Ressourcen eines mobilen Gerätes oder eines Surrogates hierdurch insgesamt weniger zu belasten, oder weil dies aufgrund der aktuellen Auslastung der Ressourcen vorteilhaft ist. Zusätzlich kann auch kontextabhängig ein bewusster

Wechsel des Kontrollflusses dann sinnvoll sein, wenn es gilt, die Verfügbarkeit eines Dienstes bei reduzierter Dienstqualität aufrechtzuerhalten. Um die vorgenannten Szenarien zu unterstützen und dabei die breite Masse teils unerfahrener Anwendungsentwickler angemessen zu unterstützen, ist es erforderlich, dass das Debugging und die Wartung einer mobilen Anwendung auf Basis einer kontextadaptiven Anwendungsarchitektur nicht komplexer als bei konventionellen mobilen Anwendungen ist und dies durch eine entsprechende Unterstützung und Integration der üblichen Entwicklerwerkzeuge gewährleistet wird. Insbesondere sollte der Ausführungszustand mithilfe der üblichen Entwicklerwerkzeuge analysierbar bleiben (M2).

Ausgehend von der Anforderung zur spontanen Interaktion mit unterschiedlichen Surrogates verschiedener Dienstanbieter (P6) kann es im Lebenszyklus einer mobilen Anwendung zusätzlich erforderlich sein, diese im Laufe der Zeit tiefergehend an unterschiedliche technische Plattformen und Dienstanbieter anzupassen. Entsprechend sollte eine kontextadaptive Anwendungsarchitektur in der Lage sein, mit anderen Plattformen und anderen Dienstanbietern zu interagieren, um hierdurch die Abhängigkeit zu einem bestimmten Dienstanbieter möglichst gering zu halten (M3).

- M1 Determiniertheit: Generell sollte der Kontrollfluss einer mobilen Anwendung, unabhängig von den verwendeten Surrogates oder der Adaptionsstrategie, deterministisch sein. Es sei denn, die Adaptionsstrategie beinhaltet explizit eine Anpassung des Kontrollflusses.
- M2 Debugging: Debugging und Wartung sollten nicht komplexer als bei konventionellen mobilen Anwendungen sein. Der fachliche Kontrollfluss der mobilen Anwendung sollte dabei durch übliche Entwicklerwerkzeuge weiterhin einfach zu analysieren und zu warten bleiben.
- M3 Offene Standards: Die Nutzung offener Standards für die Kommunikation, zum Beispiel offene Protokolle und Datenformate, sollte die einfache Erweiterbarkeit einer Architektur für mobiles Cloud Computing auf neue Plattformen und Dienstanbieter gestatten, um Abhängigkeiten zu bestimmten Dienstanbietern möglichst gering zu halten.

4.2.7. Sicherheit

Die Speicherung, Übertragung und Verarbeitung von vertraulichen Daten in einer gemeinsam genutzten Infrastruktur stellt ein generelles Problem im mobilen Cloud Computing dar [ZSG+09, HZKL10]. Insbesondere wenn vertrauliche Daten auf nicht vertrauenswürdige Surrogates ausgelagert werden, ist die Kontrolle über die Daten und damit ihre Vertraulichkeit nicht länger gewährleistet. Entsprechend ist es für die Akzeptanz einer kontextadaptiven Anwendungsarchitektur für mobiles Cloud Computing entscheidend, dass Mechanismen bereitgestellt werden, die es erlauben, festzulegen, welche Informationen ein mobiles Gerät verlassen dürfen, welche nur auf vertrauenswürdige Surrogates ausgelagert werden dürfen und welche generell nur auf dem mobilen

Gerät verbleiben sollen (SE1). Mit Blick auf die gemeinsam genutzte Infrastruktur ist es zusätzlich erforderlich, dass sowohl zwischen der ausgelagerten Funktionalität unterschiedlicher Anwendungen desselben Anwenders als auch zwischen unterschiedlichen Anwendern stets eine strikte Isolation dieser Funktionalität voneinander erfolgt, um die Beeinflussung dieser untereinander zu unterbinden. Insbesondere wenn kein vertrauenswürdiger Code ausgeführt wird, ist eine entsprechende Isolation vorzusehen (SE2).

- SE1 Schutz der Vertraulichkeit: Vertrauliche Informationen sollten nicht automatisch in eine gemeinsam genutzte Infrastruktur übertragen, dort gespeichert und verarbeitet werden. Hierfür sind entsprechende Kontrollmechanismen durch eine kontextadaptive Anwendungsarchitektur bereitzustellen.

- SE2 Isolation: Die Ausführung von nicht vertrauenswürdigen Codes auf Surrogates sollte nicht zu Seiteneffekten führen, die die ausgelagerte Funktionalität anderer Anwendungen oder anderer Anwender beeinflussen oder deren Daten kompromittieren.

4.2.8. Zusammenfassung

Im Rahmen der Anforderungsanalyse wurden in Abschnitt 4.2.1 die ISO/IEC 25010 [Int11] Kriterien für Softwarequalität herangezogen, um einen angemessenen softwaretechnischen Entwurf sicherzustellen. Diese Anforderungen wurden anschließend um die spezifischen Problemstellungen des mobilen Cloud Computings erweitert. Tabelle 4.1 fasst die im Rahmen dieser Anforderungsanalyse ermittelten Kriterien zusammen, die sowohl die Problemstellungen im Bereich der Entwicklung verteilter Systeme und des Mobile Computings berücksichtigen als auch die spezifischen Problemstellungen des mobilen Cloud Computings beleuchten. Auf dieser Basis sollen im folgenden Abschnitt eine Reihe repräsentativer Lösungsansätze ausgewählt, bewertet und vorgestellt werden.

4.3. Existierende Lösungsansätze

Im Anschluss an die Ermittlung der Anforderungen an eine kontextadaptive Anwendungsarchitektur soll in diesem Abschnitt ein Abgleich mit existierenden Lösungsansätzen erfolgen, die im selben oder in angrenzenden Forschungsgebieten und Anwendungsbereichen entstanden sind. Innerhalb dieses Abgleichs werden eine Reihe relevanter verwandter Arbeiten identifiziert und gegen den entwickelten Kriterienkatalog evaluiert. Im Anschluss werden besonders relevante Lösungsansätze näher analysiert, bevor eine Zusammenfassung der Ergebnisse erfolgt.

Verfügbarkeit		Skalierbarkeit		Wartbarkeit	
A1	Fehlerbehandlung	S1	Overhead	M1	Determiniertheit
A2	Erweitertes Fehlermanagement	S2	Dienstsuche und Integration	M2	Debugging
A3	Prognose der Konnektivität	S3	Parallelisierung	M3	Offene Standards
A4	Effizienz	S4	Effizienz		

Portierbarkeit		Nutzbarkeit		Sicherheit	
P1	Generelle Auslagerungsfähigkeit	U1	Einfache Nutzbarkeit	SE1	Schutz der Vertraulichkeit
P2	Grundlegende Adaptionsfähigkeit	U2	Angemessene Abstraktion	SE2	Isolation
P3	Erweiterte Adaptionsfähigkeit	U3	Unterstützung von Standards		
P4	Koexistenz	U4	Automatische Konfiguration		
P5	Deployment				
P6	Offenheit				

Tabelle 4.1.: Übersicht der Kriterien der Anforderungsanalyse

4.3.1. Methodik der Auswahl

In den vergangenen Jahren sind eine Vielzahl von Forschungsarbeiten im Bereich der vorgestellten Klassen von Anwendungsfällen entstanden, die sich damit auch den in dieser Arbeit betrachteten Problemstellungen zurechnen lassen. Der Abgleich dieser Arbeiten stützt sich methodisch auf eine strukturierte Literaturrecherche, bei der zunächst 77 relevante verwandte Arbeiten identifiziert wurden, welche im Anschluss anhand ihrer Originalität und Relevanz auf 40 verwandte Arbeiten reduziert werden, welche im Folgenden gegen den entwickelten Anforderungskatalog evaluiert werden. Die Bewertung von Originalität und Relevanz stützt sich einerseits auf die Anzahl der Referenzen auf die entsprechende Veröffentlichung. Bei neueren Arbeiten, aufgrund der hier naturgemäß geringeren Anzahl von Referenzen auf die entsprechende Veröffentlichung, wird abweichend die Angemessenheit und der Innovationsgrad der Lösung in Bezug auf das mobile Cloud Computing berücksichtigt. Die Reduktion stützt sich ebenfalls auf das Weglassen weitgehend ähnlicher Lösungsansätze und fokussiert sich zusätzlich auf die Auswahl verwandter Arbeiten auf Lösungen, die sich angemessen in den Entwicklungsprozess mobiler Anwendungen integrieren lassen. Lediglich ergänzend werden weitere prominente Arbeiten aus Gründen der Vollständigkeit erwähnt.

Die Evaluation existierender Ansätze für kontextadaptive Anwendungsarchitekturen gegen den hierfür entwickelten Kriterienkatalog schließt entsprechend keine Arbeiten zur Entwicklung kontextadaptiver Systemunterstützungen und Arbeiten zur Prognose von Kontextattributen ein, da diese einen Großteil der Kriterien nicht abdecken. Diese Arbeiten werden entsprechend im Zusammenhang mit der Entwicklung geeigneter Mechanismen zur Kontextadaption in den folgenden Kapiteln noch separat betrachtet.

4.3.2. Quantitative Betrachtung

Bei Betrachtung der im Rahmen der Literaturrecherche ermittelten Arbeiten zeichnen sich mehrere Erkenntnisse ab. Einerseits zeigt sich, dass das Themengebiet bereits seit Längerem untersucht wird, da erste Arbeiten wie das Emerald-System [JLHB88] bereits im Jahr 1988 entwickelt wurde. Gleichzeitig ist wie in Abbildung 4.7 gezeigt eine starke Zunahme der Veröffentlichungen ab dem Jahr 2009 zu erkennen, was die anhaltende Relevanz des Forschungsgebiets unterstreicht.

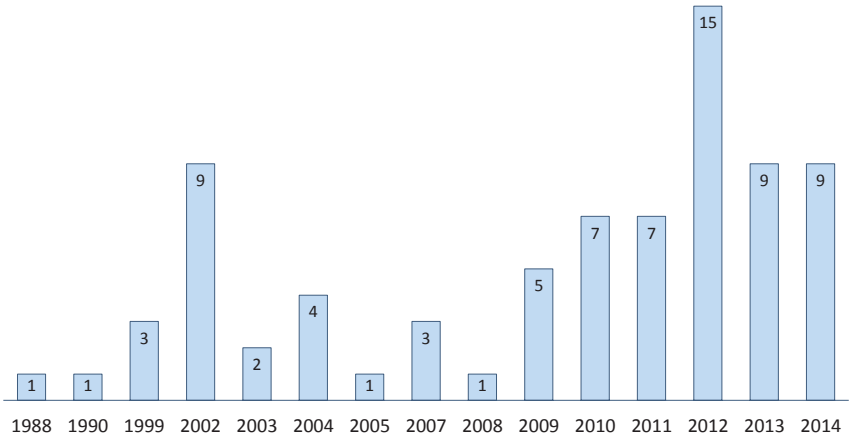

Abbildung 4.7.: Anzahl veröffentlichter Arbeiten im Zeitverlauf

Bei näherer Betrachtung zeigt sich zudem, dass sich die vergleichsweise hohe Anzahl von Veröffentlichungen im Jahr 2002 auf primär für das Cloud Computing entwickelte Ansätze konzentriert, die auf die Anwendung im Bereich mobiler Geräte hin erweitert wurden. Die Veröffentlichungen ab dem Jahr 2012 hingegen berücksichtigen vornehmlich explizit für das Teilgebiet des mobilen Cloud Computings entwickelte Lösungen.

4.3.3. Kategorisierung

Die wie zuvor beschrieben ermittelten Lösungsansätze sollen im Folgenden in sechs verschiedene Kategorien gruppiert werden, um eine Abschätzung der generellen Eignung einer jeweiligen Kategorie von Lösungen im Hinblick auf die Realisierung einer kontextadaptiven Anwendungsarchitektur zu erhalten.

Die Kategorisierung stützt sich dabei in erster Linie auf das ursprüngliche Themengebiet der Arbeit oder auf die Verwendung ähnlicher Basistechnologien. Entsprechend werden die existierenden Lösungsansätze zunächst in Bezug darauf klassifiziert, ob sie spezifisch für das mobile Cloud Computing konzipiert wurden. Die so ermittelten sechs Kategorien werden nun im Folgenden jeweils beschrieben und zueinander in Beziehung gesetzt, bevor die Evaluation der jeweiligen Lösungsansätze gegen den zuvor entwickelten Kriterienkatalog erfolgt.

Spezialisierte Sprachen Eine der wesentlichen Herausforderungen bei der Entwicklung einer kontextadaptiven Anwendungsarchitektur für das mobile Cloud Computing betrifft die Berücksichtigung eines ständig und schnell wechselnden Kontextes. Ein Lösungsansatz besteht darin, den Anpassungsbedarf der im Rahmen eines Kontextwechsels entsteht, direkt in den Kontrollfluss einer mobilen Anwendung zu integrieren. Hierfür wird ein spezifischer Anpassungsbedarf direkt durch entsprechende Kontrollstrukturen abgebildet, die als domänenspezifische Sprache (DSL) realisiert sein können, wie es beispielsweise im Konzept der kontextorientierten Programmierung (context-oriented programming) Anwendung findet. Dieses von Hirschfeld et al. in [HCN08] vorgeschlagene Konzept existiert aktuell als Erweiterung einer Reihe prominenter Programmiersprachen und wurde ebenso bereits für ein prominentes Betriebssystem für mobile Geräte angepasst [SAH11]. Der Ansatz der kontextorientierten Programmierung ermöglicht es entsprechend zur Laufzeit einer mobilen Anwendung, das Verhalten dieser kontextabhängig anzupassen.

Eine weitere domänenspezifische Sprache ist AmbientTalk [VCMB+07], eine objektorientierte Sprache, die für den Einsatz in mobilen Ad-hoc-Netzwerken und dementsprechend häufig wechselnden Konnektivitätseigenschaften konzipiert wurde. Listing 4.1 zeigt eine verteilte Version des Hello-World-Beispiels.

```
/* Define types that could be discovered on the network */
deftype Greeter;

def makeGreeter(myName) {
   /* Spawn an actor */
   actor: {
      /* Actors have a separate namespace, include the language
          futures in it */
      import /.at.lang.futures;
      /* A method that could be called by other greeters */
      def getName(){myName};
      /* Export this actor on the network */
      export: self as: Greeter;
      /* Main logic: if we discover another Greeter ... */
      whenever: Greeter discovered: {|other|
         /* Asynchronously get his name, and greet him */
         when: other<-getName()@FutureMessage becomes: {|name|
            system.println("Hello " + name + " from " + myName);
         };
      };
   };
};
/* Spawn 2 actors that will greet each other */
makeGreeter("Alice");
makeGreeter("Bob");
```

Listing 4.1: Verteilte Version des Hello-World-Beispiels in AmbientTalk (VCMB+07)

Aus dem Code-Beispiel ist erkennbar, dass im Sprachumfang selbst eine Unterstützung für Funktionen wie die Suche nach Diensten (service discovery), den Aufruf entfernter Dienste durch asynchronen Nachrichtenaustausch (futures) und die transparente Behandlung von Verbindungsabbrüchen existiert. Weiterhin erlaubt die Sprache durch einen vollständig ereignisgesteuerten Kontrollfluss und die Verwendung des Aktormodells eine nebenläufige und asynchrone Verarbeitung. Die Sprache ist auf Basis der Programmiersprache Java implementiert, wodurch die Lauffähigkeit auf mobilen Systemplattformen wie Android ermöglicht wird. Eine entsprechende Bibliothek[7] bietet zusätzlich eine Integration der beiden Sprachen.

Die gezeigten Eigenschaften erlauben somit die Entwicklung kontextadaptiver Anwendungen, die auf die Problemstellung des mobilen Cloud Computings hin angepasst werden können. Spezialisierte Sprachen erfordern jedoch entsprechende Kenntnisse, die bei der breiten Masse an Anwendungsentwicklern nicht vorausgesetzt werden können. Entwickler mit der entsprechenden Sprachsyntax und dem Programmierstil vertraut zu machen, ist dabei nicht das einzige Problem. Zusätzlich ist es ebenso erforderlich, durch das mobile Betriebssysteme eine entsprechende Unterstützung anzubieten, was momentan nicht der Fall ist.

Frameworks und Middleware-Plattformen Traditionelle Middleware wurde mit dem Ziel entwickelt, zwischen verschiedenen Anwendungen zu vermitteln und dabei die Komplexität dieser Vermittlung und insbesondere die Kommunikation innerhalb eines verteilten Systems vor diesen Anwendungen zu verbergen. Hierzu stellt eine Middleware den Anwendungen üblicherweise eine Reihe von High-Level-Mechanismen für den Nachrichtenaustausch über entsprechende Schnittstellen zur Verfügung. Dies umfasst typischerweise den entfernten Aufruf von Methoden und wird dabei klassisch um Funktionen wie um einen Transaktionsmonitor ergänzt, der es erlaubt, eine transaktionssichere Verarbeitung in einem verteilten System sicherzustellen.

Middleware für klassische, stationäre, verteilte Systeme setzt hierfür eine entsprechend stabile Verbindung zwischen den einzelnen Systemen voraus. Diese stabilen Verbindungen mit einer hohen Bandbreite und gleichbleibenden Verfügbarkeit sind in mobilen Umgebungen jedoch nicht vorhanden, was die Anforderungen an eine Middleware für mobile Systeme entsprechend erhöht.

So lassen sich die Aufgaben einer Middleware für klassische verteilte Systeme mit den entsprechenden Anforderungen an Kommunikation, Koordination, Zuverlässigkeit, Skalierbarkeit und Sicherheit beschreiben [BC16]. Werden diese Anforderungen auf den Bereich mobiler Middleware übertragen, so ergeben sich aus den in Abschnitt 2.2.4 aufgezeigten Restriktionen und Eigenschaften mobiler Geräte nach [BC16] drei wesentliche Problemfelder, die bei der Konstruktion einer mobilen Middleware Berücksichtigung finden soll-

[7] https://soft.vub.ac.be/amop/at/tutorial/symbiosis

ten und entsprechend gleichzeitig als die wesentlichen Anforderungen für eine mobile Middleware genannt werden:

- Die Mobilität der Nutzer selbst, welche die wechselnde Konnektivität und Bandbreite drahtloser Datenverbindungen weiter einschränkt und eines der zentralen Probleme mobiler Systeme widerspiegelt.
- Die Ressourcenbeschränkung mobiler Geräte, im Wesentlichen die begrenzte Rechen- und Speicherressourcen sowie der begrenzte Energievorrat, welche ebenso Einfluss auf die Konnektivität eines mobilen Gerätes haben.
- Die Berücksichtigung des wechselnden Ausführungskontextes[8].

Entsprechend ist eine angemessene mobile Middleware dafür konzipiert, mit der häufig wechselnden Konnektivität mobiler Geräte umzugehen, und sollte hierfür beispielsweise Mechanismen zum asynchronen Nachrichtenaustausch anbieten, welche die wechselnde Konnektivität mobiler Geräte berücksichtigen und zusätzlich entsprechende Mechanismen zur Fehlerbehandlung oder zum Zwischenspeichern von Nachrichten bereitstellen. Hierfür ist es erforderlich, dass eine mobile Middleware den Kontext eines Nutzers, seines Gerätes und der umgebenden Infrastruktur berücksichtigt und Änderungen im Kontext möglichst gut antizipiert, um zur Laufzeit einer mobilen Anwendung den Bedarf zur Adaption zu erkennen und eine entsprechende Anpassung auszulösen.

In diesem Zusammenhang wurde schon früh erkannt, dass das bewusste Reduzieren der Transparenzeigenschaften einer klassischen Middleware in diesem Zusammenhang sinnvoll ist, damit eine mobile Middleware einen bestimmten Teil des aktuellen Ausführungskontextes an die mobilen Anwendungen weitergeben kann, um bestimmte Adaptionsbedarfe innerhalb der mobilen Anwendung zu ermöglichen [SDA99]. Ebenso wurde festgestellt, dass hierzu die oberen Schichten einer Middleware einen bestimmten Grad an Transparenz in Form von Nachrichten über Änderungen des aktuellen Ausführungskontext an die darüberliegenden Schichten oder Anwendungen weiterreichen sollten, um es diesen zu ermöglichen, die Adaption auszulösen [SBCD09].

Für das Problemfeld des mobilen Cloud Computings wurden entsprechend oft existierende Middleware-Lösungen um die Berücksichtigung der speziellen Eigenschaften des mobilen Cloud Computings erweitert. Diese Form der Erweiterung wird oft in Form eines Frameworks realisiert. Das Framework stellt für den Entwickler ein softwaretechnisches Gerüst bereit, welches für sich selbst noch keine fertige Anwendung darstellt, sondern im Sinne einer Umkehrung des Kontrollflusses (inversion of control) den vom Entwickler zu ergänzenden Code zum entsprechenden Zeitpunkt zur Ausführung bringt. Das Framework gibt hierdurch die Architektur der Anwendung vor und sorgt durch

[8] Abweichend zur Definition in Abschnitt 3.3.2 entspricht der Begriff des Ausführungskontextes an dieser Stelle der Definition aus [BC16].

die Umkehrung des Kontrollflusses dafür, dass sich der Entwickler auf die Implementierung der Geschäftslogik konzentriert, welche im Anschluss durch das Framework benutzt wird.

Obwohl Frameworks in Kombination mit Middleware damit etablierte Konzepte für die Entwicklung von Anwendungen in heterogenen und verteilten Umgebungen darstellen, kann es für die Entwickler mobiler Anwendungen eine Hürde darstellen, sich zunächst mit den Aspekten dieser Basistechnologien auseinanderzusetzen.

Verteilte virtuelle Maschinen Im Gegensatz zur vorgenannten Kategorie der Frameworks und Middleware-Plattform, die eine explizite Anpassung (refactoring) einer klassischen mobilen Anwendung erfordern, um diese für das mobile Cloud Computing nutzbar zu machen, versucht diese Kategorie von Lösungen, vollständig auf die Erfordernis der Anpassung durch den Entwickler zu verzichten. Stattdessen wird in dieser Kategorie von Lösungen versucht, sämtliche Details des Ausführungskontexts vor der mobilen Anwendung zu verbergen und so durch einen hohen Grad an Verteilungstransparenz die Entscheidung über die Verteilung der Funktionalität einer mobilen Anwendung durch die darunterliegende Ausführungsschicht zu treffen. Verteilte virtuelle Maschinen wie *Jessica2* [ZWL02] und *exCloud* [MLW11], die eine Virtualisierung auf Anwendungsebene (application-layer virtual machine) bereitstellen, versuchen hierdurch, die Verteilung der Funktionalität einer mobilen Anwendung im Hinblick auf ein Adaptionsziel zu koordinieren.

Dieser Ansatz, in dem die mobile Anwendung keine Kenntnis davon besitzt, dass sie verteilt ausgeführt wird, führt gleichzeitig dazu, dass die darunterliegende Virtualisierungsschicht ebenso nur in stark begrenztem Umfang Kenntnis davon hat, welcher Teil des globalen Ausführungszustands der mobilen Anwendung (Heap, Stack und Variablenbelegung) von einer Funktionalität der mobilen Anwendung benötigt wird. Dies wiederum macht es erforderlich, den globalen Ausführungszustand einer mobilen Anwendung zwischen den an der Ausführung beteiligten Surrogates laufend zu synchronisieren.

Obwohl mit diesem Ansatz die Entwicklung mobiler Anwendungen stark vereinfacht wird, erfordert es aufseiten der Virtualisierungsschicht entsprechend gute Heuristiken, um die verteilte Ausführung der mobilen Anwendung hinreichend effizient durchführen zu können. Ebenso sind diese Lösungen oft nicht auf die wechselnde Konnektivität mobiler Umgebungen vorbereitet.

Pervasive und Ubiquitous Computing Ansätze, aus dieser Kategorie von Lösungsansätzen entstammen oft den Anwendungsbereichen der Smart Homes oder allgemeiner dem Forschungsgebiet der Ambient Intelligence. Sie wurden entwickelt, um (mobile) Anwendungen kontextadaptiv in Abhängigkeit vorhandener Ressourcen auszuführen, und bieten entsprechend fortgeschrittene Mechanismen für das Auffinden von Ressourcen [Sch07, vSvHW+06, PRK09] und für die Verteilung von Daten über verteilte Dateisysteme [SKK+90, HBC13]. Prominente Verwandte für einen entsprechend ganzheitlichen Ansatz

auf diesem Gebiet sind *Vivendi* [BSPO03] und *Gaia* [RHC+02]. Technisch sind diese Lösungen auf einer betriebssystemnahen Ebene implementiert.

Obwohl sie nicht explizit für den Einsatz im mobilen Cloud Computing konzipiert wurden, erlauben sie es, Anwendungen bzw. deren Betriebssystemprozesse zwischen verschiedenen Knoten innerhalb eines Netzwerks zu verschieben. Der Fokus dieser Lösungen liegt dabei in der Fähigkeit zur Anpassung an den aktuellen Ausführungs- und Nutzungskontext. Diese Anpassungslogik wird in der Entwicklungsphase einer mobilen Anwendung mehrheitlich durch domänenspezifische Sprachen oder High-level-Schnittstellen für bestehende Programmiersprachen realisiert, die Entwickler zur Implementierung einer entsprechenden Adaptionslogik nutzen können.

Dies erfordert es im Umkehrschluss, dass Entwickler sich mit der Spezifikation der jeweiligen Lösung vertraut machen müssen, und setzt zusätzlich eine entsprechende Lauffähigkeit auf allen für die Ausführung benötigten Geräten voraus. Insbesondere bei Lösungen, die auf betriebssystemnaher Ebene realisiert sind, setzt dies eine entsprechende Unterstützung des jeweiligen Betriebssystems voraus, wodurch die Nutzbarkeit der Lösung für eine Vielzahl heterogener mobiler Geräte eingeschränkt sein kann.

Systemunterstützungen für mobiles Cloud Computing Ausgelöst durch das aufgezeigte Wachstum mobiler Geräte und mobiler Anwendungen haben sich in den vergangenen Jahren eine Reihe von Forschungsarbeiten mit den Ressourcenbeschränkungen mobiler Geräte beschäftigt. Hierfür wurden Lösungsansätze entwickelt, von denen sich ein Großteil explizit mit der Auslagerung von Berechnungen (computation offloading) befasst, und entsprechend auch die spezifischen Problemstellungen im Bereich des mobilen Cloud versucht, direkt zu adressieren. Die Forschungsarbeiten lassen sich zunächst grob dahingehend weiter unterteilen, inwieweit sie Transparenzeigenschaften und insbesondere Aspekte der Verteilungstransparenz den mobilen Anwendungen gegenüber explizit machen (Anwendungsarchitekturen für mobiles Cloud Computing) oder sie verbergen (Systemunterstützungen für mobiles Cloud Computing).

Lösungsansätze, die den wechselnden Kontext mobiler Umgebungen und die Verteilung der Funktionalität einer mobilen Anwendung auf Surrogates gegenüber der mobilen Anwendung verbergen, setzen hierzu oft auf das Konzept der Virtualisierung – entweder auf Anwendungs- oder auf Systemebene. Der Vorteil diese Ansätzes besteht darin, dass keine oder nur geringe Anpassungen an einer mobilen Anwendung vorgenommen werden müssen und hierdurch ein Großteil der bereits existierenden mobilen Anwendungen für mobiles Cloud Computing nutzbar gemacht werden können.

Eine prominente Lösung wie *CloneCloud* [CIM+11] nutzt hierfür eine statische Codeanalyse im Zusammenhang mit einem Monitoring zur Laufzeit, um festzustellen, wie eine mobile Anwendung sinnvoll auf ein Surrogate verteilt werden kann, was im Verlauf der Arbeit noch näher vorgestellt wird.

Diese Kategorie von Lösungen berücksichtigt somit einzelne Aspekte des Kontexts mobiler Geräte; da die mobile Anwendung selbst nicht für eine verteilte Ausführung entwickelt wurde, ergeben sich ähnliche Herausforderungen wie im Bereich der verteilten virtuellen Maschinen, die es erforderlich machen, eine sinnvolle Verteilung der Funktionalität einer mobilen Anwendung mithilfe guter Heuristiken abzuschätzen. Dabei unterstützt diese Kategorie von Lösungen lediglich die Verteilung von Funktionalität auf ein einzelnes Surrogate, zu welchem eine zuverlässige Verbindung vorausgesetzt wird.

Anwendungsarchitekturen für mobiles Cloud Computing Neben der zuvor genannten Kategorie von Lösungsansätzen, die einen möglichst hohen Grad an Verteilungstransparenz gegenüber mobilen Anwendungen herstellen, um auch nicht für das mobile Cloud Computing modifizierte mobile Anwendungen in die Infrastruktur verteilen zu können, existiert eine weitere Kategorie von Lösungsansätzen, die sich dadurch abgrenzt, dass sie eine explizite Anpassung mobiler Anwendungen an die Problemstellungen des mobilen Cloud Computings durch die Entwickler voraussetzen und hierfür eine entsprechende Unterstützung bereitstellen.

Eine typische Lösung aus dieser Kategorie stellt *AlfredO* [GRJ+09] dar – ein komponentenbasierter Ansatz, bei dem das Wissen des Entwicklers genutzt wird, um die Geschäftslogik einer mobilen Anwendung in abgeschlossene Einheiten zu kapseln, die eine hohe Kohäsion und möglichst geringe Kopplung zu übrigen Einheiten besitzen, um hierdurch die Funktionalität einer mobilen Anwendung auf Basis der so definierten Einheiten in die Infrastruktur zu verlagern.

Im Rahmen der Auslagerungsentscheidung werden sowohl im Bereich der Systemunterstützungen als auch in den hier beschriebenen Anwendungsarchitekturen für mobiles Cloud Computing Heuristiken verwendet, die die Größe des zu synchronisierenden globalen Ausführungszustands, die erwartete Übertragungszeit und die erwartete Zeit oder Energieeinsparung berücksichtigen, um eine sinnvolle Verteilung der Funktionalität einer mobilen Anwendung im Hinblick auf ein gewähltes Adaptionsziel zu bewerten.

4.4. Qualitative Bewertung existierender Lösungsansätze

Nachdem die verwandten Arbeiten zunächst klassifiziert wurden, soll im Folgenden die Methodik der Bewertung und im Anschluss die Bewertung dieser Arbeiten anhand des entwickelten Anforderungskatalogs selbst vorgestellt werden.

Die Bewertung betreffend soll, um eine weitgehende Vergleichbarkeit der unterschiedlichen Kategorien von Lösungsansätzen zu ermöglichen, lediglich beurteilt werden, ob ein bestimmtes Kriterium vollständig erfüllt wurde (++), teilweise erfüllt wurde (+), nicht erfüllt wurde (-) oder nicht adressiert wurde (0). Um die Methodik der Bewertung weitergehend zu verdeutlichen, sollen

im Folgenden vier Lösungen unterschiedlicher Kategorien weitergehend untersucht, vorgestellt und jeweils die Erfüllung der im Anforderungskatalog genannten Kriterien beschrieben werden. Diese vier Lösungen wurden aufgrund ihrer Prominenz und als stellvertretend für eine bestimmte Kategorie von Optimierungszielen im mobilen Cloud Computing ausgewählt.

4.4.1. Optimierung der Ausführungszeit

CloneCloud [CIM+11] stellt einen der prominentesten Lösungsansätze aus dem Forschungsgebiet des mobilen Cloud Computings dar und soll als Vertreter der Kategorie der Systemunterstützungen für mobiles Cloud Computing vorgestellt werden.

Funktionsweise Das Konzept wurde für die Android-Systemplattform entwickelt und nutzt eine Virtualisierung auf Systemebene, bei der ein Abbild (Image) des kompletten nichtflüchtigen Speichers des mobilen Gerätes, welches alle Partitionen umfasst, die das Android-Betriebssystem dort angelegt hat, zunächst auf ein in der Infrastruktur befindliches leistungsstarkes Surrogate repliziert. Im Anschluss wird der so erzeugte „Clone“ des mobilen Gerätes auf dem Surrogate gestartet. Diese Replikation dient dazu, die rechenintensive Funktionalität einer mobilen Anwendung in die Infrastruktur auszulagern, um hierdurch eine Einsparung von Energie oder eine Beschleunigung der mobilen Anwendung zu erreichen. Das gezeigte Konzept fokussiert allerdings primär die Beschleunigung rechenintensiver Aufgaben, wie es am Beispiel eines Virenscanners oder der Suche nach Bildern gezeigt wird.

Diesen Zielen folgend wird bei CloneCloud ein Thread, der davor steht, einen rechenintensiven Teil des Programmcodes einer mobilen Anwendung zu durchlaufen, auf dem mobilen Gerät pausiert und sein aktueller Ausführungszustand (virtual state: program counter, registers, and stack) auf das Surrogate übertragen und somit der Ausführungskontext mit dem Surrogate synchronisiert, um im Anschluss auf dem Surrogate die Ausführung des Threads wieder aufzunehmen. Sobald der als rechenintensiv und für die Auslagerung identifizierte Teil des Programmcodes durchlaufen ist, erfolgt eine Rückführung des Kontrollflusses auf das mobile Gerät. Hierzu wird nach einem erneuten Anhalten des Threads der neue Ausführungskontext des Surrogates wieder mit dem mobilen Gerät synchronisiert und der entsprechende Thread auf dem mobilen Gerät fortgesetzt, wie in Abbildung 4.8 veranschaulicht wird.

Die Heterogenität zwischen dem mobilen Gerät, welches ein Prozessor mit ARM-Architektur verwendet, und dem Surrogate, das ein Prozessor mit x86-Architektur verwendet, wird von Chun et al. in [CIM+11] dadurch für die mobile Anwendung verborgen, dass auch auf dem Surrogate eine entsprechende Version[9] des Android-Betriebssystems zum Einsatz kommt. Der übrige Anpassungsbedarf fällt laut den Autoren so gering raus, sodass die entsprechenden Anpassungen auf dem mobilen Gerät selbst durchgeführt werden können. So

[9] http://www.android-x86.org/

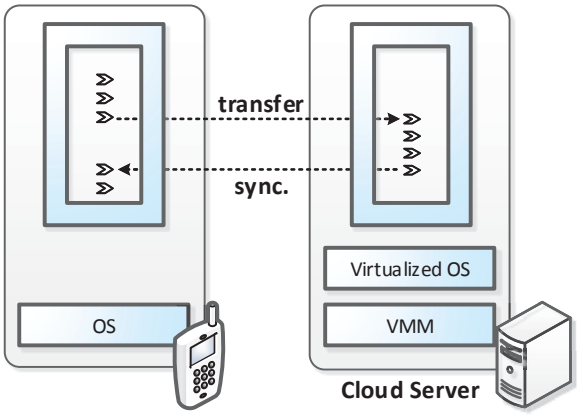

Abbildung 4.8.: CloneCloud: Auslagerung eines Threads

werden direkte Abhängigkeiten zwischen den verschiedenen Prozessorarchitekturen dadurch aufgelöst, dass Referenzen auf Speicheradressen wie Pointer durch Referenzen auf Klassen und Methodennamen ersetzt werden.

Bei dieser Art der Anpassung einer mobilen Anwendung zur Laufzeit wird der Quelltext der Anwendung nicht benötigt, wodurch es prinzipiell möglich ist, die Funktionalität beliebiger Anwendungen, die bereits auf dem mobilen Gerät vorhanden sind, in die Infrastruktur zu verlagern. Ein wesentlicher Nachteil dieses Ansatzes ist es allerdings, dass aufgrund der erwähnten Verbindungseigenschaften mobiler Geräte ein sogenanntes Distributed Shared Memory (DSM) nur schwierig umzusetzen ist. Dies wiederum führt dazu, dass die nebenläufige Ausführung von Threads einer mobilen Anwendung nicht möglich ist, sobald ein Thread auf ein Surrogate ausgelagert wird. Entsprechend bleiben die Mehrkernprozessoren mobiler Geräte weitestgehend ungenutzt, sobald eine Auslagerung durchgeführt wird, und es ist keine gleichzeitige Nutzung der Rechenleistung des mobilen Gerätes und des Surrogates möglich. Von den Autoren wird entsprechend hervorgehoben, dass Anwendungen, die nur geringe Abhängigkeiten zwischen ihren Threads haben, deutlich stärker von einer Unterstützung durch CloneCloud profitieren können.

Im Fall der konkreten Realisierung von CloneCloud bedeutet dies, dass alle weiteren Threads einer mobilen Anwendung während einer Auslagerung warten und dies für den Anwender während der Auslagerung den Anschein erweckt, dass die mobile Anwendung nicht auf Benutzereingaben reagiert. Um die Anzahl dieser Situation zu reduzieren, wird üblicherweise kein Programmcode der mobilen Anwendung ausgelagert, der eine direkte Abhängigkeit zur GUI der mobilen Anwendung besitzt. Ebenso wird keine Funktionalität ausgelagert, die Abhängigkeiten zu nativ implementierter API-Funktionalität, beispielsweise dem Zugriff auf die Kamera des mobilen Geräts, besitzt. Ein suboptimaler softwaretechnischer Entwurf des Entwicklers kann es somit verhindern, dass überhaupt für die Auslagerung zur Verfügung stehende Funktionsblöcke identifiziert werden können.

Um die für die Auslagerung relevanten Teile des Programmcodes zu identifizieren, nutzt CloneCloud eine statische Analyse des Programmcodes, aus der zunächst möglichst sinnvolle Migrationspunkte abgeleitet werden, die sich dadurch charakterisieren, dass der zu übertragende Ausführungszustand möglichst wenig Speicher und entsprechend wenig Übertragungskapazität beansprucht. Auf Basis dieser Migrationspunkte werden in einem folgenden Schritt eine Reihe möglicher Partitionen gebildet, die zusätzlich berücksichtigen, ob Teile des Programmcodes eine GUI-Interaktion oder einen API-Zugriff auf native Funktionen des mobilen Geräts enthalten. Letztere werden als nur für die Ausführung auf dem mobilen Gerät qualifiziert markiert. In einem weiteren Schritt wird durch ein mehrfaches Ausführen der mobilen Anwendung und unter Nutzung eines Profilers ermittelt, inwieweit sich die lokale und die ausgelagerte Ausführung der einzelnen Partitionen im Hinblick auf die Optimierungsziele der minimierten Ausführungszeit und der Energieeinsparung für die jeweilige Partition als vorteilhaft erweisen. Das Profiling erfasst dabei ebenso weitere Parameter wie die tatsächliche Größe des zu übertragenden Ausführungszustandes.

Abbildung 4.8 zeigt vereinfacht die Übertragung eines Threads am zuvor ermittelten Migrationspunkt. Realisiert wird diese Übertragung mithilfe des sogenannten *Migrators*, einer von zwei zusätzlichen Komponenten, die im Zusammenhang mit den Auslagerungsprozesses benötigt werden, und der dafür verantwortlich ist, die auszulagernden Threads anzuhalten, zu übertragen, den Ausführungszustand zu synchronisieren und den Thread auf dem mobilen Gerät fortzusetzen. Die zweite Komponente ist der sogenannte *Node Manager*: Dieser ist dafür verantwortlich, das Surrogate auf die Verarbeitung von ausgelagerten Threads vorzubereiten, wozu es gilt, das Systemabbild des mobilen Geräts mit dem entsprechenden Abbild auf dem Surrogate zu synchronisieren.

Bewertung Obwohl CloneCloud den Kontext des mobilen Gerätes nur in stark begrenztem Umfang in Form der verfügbaren Bandbreite zu dem Surrogate berücksichtigt, kann dieser Lösungsansatz in Bezug auf die Portierbarkeit (P1-P6) insgesamt als gut betrachtet werden, da unter anderem ein einfaches Deployment des Systemabbildes des mobilen Geräts auf dem neuen Surrogate durch den Node Manager ermöglicht wird. Das Deployment umfasst allerdings die Übertragung des üblicherweise sehr großen Abbildes des nichtflüchtigen Speichers des mobilen Geräts.

Ein weiterer Vorteil des vorgestellten Ansatzes ist es, dass mobile Anwendungen, deren Quellcode nicht vorliegt, durch CloneCloud unterstützt werden, woraus die gute Nutzbarkeit und Wartbarkeit der Lösung (U1-U4, M1-M3) resultiert. Der hohe Automatisierungsgrad dieser Lösung erfordert allerdings entsprechend gute Heuristiken, um eine sinnvolle Aufteilung der Funktionalität einer mobilen Anwendung zu ermöglichen, die wiederum eine effiziente Nutzung der vorhandenen Ressourcen im Hinblick auf das Adaptionsziel erlaubt. Gleichzeitig wird die Skalierbarkeit der Lösung (S1-S4) dadurch begrenzt, dass keine Parallelisierung der ausgelagerten Threads möglich ist und

die Ausführung weiterer lokaler Threads blockiert wird. Ebenfalls beschränkt ist die Verfügbarkeit (A1-A4) dieser Lösung, da Verbindungsabbrüche nicht betrachtet werden und eine stabile Verbindung zu den Surrogates unterstellt wird. Die Sicherheit (SE1-SE2) dieses Ansatzes ist ebenso nur begrenzt für die Nutzung im mobilen Cloud Computing geeignet, da nicht festgelegt werden kann, welche Informationen nicht in die Infrastruktur übertragen werden dürfen.

4.4.2. Optimierung des Energiebedarfs

ThinkAir [KAH+12] bietet ähnlich wie das zuvor vorgestellte CloneCloud eine, in Bezug auf die Größe der Einheiten von Programmcode, feingranulare Auslagerung zwischen einem mobilen Gerät und einem Surrogate in der Infrastruktur an. Der wesentliche Unterschied im Gegensatz zu CloneCloud besteht darin, dass bei dieser Lösung der Entwickler die für die Auslagerung vorgesehenen Methoden explizit durch eine entsprechende Annotation im Entwicklungsprozess markiert.

Funktionsweise Zur Laufzeit der Anwendung werden auf Basis der Annotationen zunächst verschiedene Profiler genutzt, um die Ausführungsmetriken einzelner Methoden zu ermitteln. Erfolgt ein weiterer Aufruf derselben Methode, so entscheidet der *Execution Controller*, eine der Komponenten des ThinkAir-Frameworks, über den Ausführungsort dieser Methode. Dieser steuert die verteilte Ausführung unter Berücksichtigung der erwarteten Ausführungszeit, dem erwarteten Energieverbrauch und der durch den Profiler erstellten Ausführungsstatistik. Hierzu wird das jeweilige Adaptionsziel der Zeit- oder Energieeinsparung oder einer Kombination aus beiden berücksichtigt. Soll die Methode auf einem Surrogate zur Ausführung gebracht werden, so wird das aufrufende Objekt auf das Surrogate übertragen, der Methodenaufruf durchgeführt und das entsprechende Ergebnis mit dem Ausführungszustand des mobilen Geräts synchronisiert. Um dies zu ermöglichen, ist die vorhergehende Übertragung des zugehörigen Codes in die Infrastruktur erforderlich. Hierzu existiert neben dem Execution Controller und dem Profiler auf dem mobilen Gerät eine weitere Komponente, nämlich der Client Handler, der für die Kommunikation zwischen mobilem Gerät und Surrogate verantwortlich ist. Aufseiten des Surrogates existieren die entsprechenden Gegenstücke der genannten Komponenten, wie Abbildung 4.9 veranschaulicht. Weitere Details zur Funktionsweise dieser Komponenten werden in [KAH+12] nicht genannt.

Bewertung Der Execution Controller bietet eine Reihe von Mechanismen, um mit Verbindungsabbrüchen umzugehen. In Bezug auf eine angemessene Kontextadaption wird beispielsweise im Fall eines Verbindungsabbruchs eine lokale Ausführung der Methode initiiert, und es wird versucht, die Verbindung zum Surrogate wiederherzustellen. Entsprechend kann diese Lösung im Hinblick auf die Verfügbarkeit (A1-A4) als sehr gut beurteilt werden.

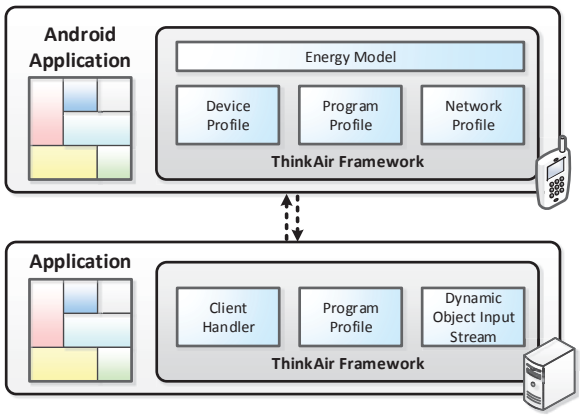

Abbildung 4.9.: Komponenten des ThinkAir-Frameworks

Die Möglichkeit, neue Surrogates in die verteilte Ausführung einbinden zu können und mehrere von ihnen gleichzeitig für die verteilte Ausführung einzelner Methoden verwenden zu können, führt entsprechend zu einer ebenfalls guten Portierbarkeit (P1-P6) und Skalierbarkeit (S1-S4) der Lösung. Ebenso kann die Nutzbarkeit (U1-U4) der Lösung als gut bewertet werden, da es für den Entwickler vergleichsweise einfach möglich ist, die entsprechenden Adaptionsziele der Zeit- oder Energieeinsparung flexibel miteinander zu kombinieren. Die zur Entwicklung bereitgestellte Erweiterung der Entwicklungsumgebung *Eclipse*[10] führt zu einer guten Wartbarkeit dieses Lösungsansatzes (M1-M3); jedoch bleibt der Entwickler in der Verantwortung, die für die verteilte Ausführung bestimmten Methoden zu identifizieren und entsprechend zu annotieren. Ähnlich wie bei CloneCloud werden Sicherheitsaspekte (SE1-SE2) nicht direkt adressiert, die Möglichkeit zur Annotation bietet allerdings indirekt die Möglichkeit, zu steuern, ob eine bestimmte Funktionalität in die Infrastruktur ausgelagert werden soll.

4.4.3. Verteilte Ausführung von Hintergrundprozessen

Cuckoo [KPKB10] basiert auf der Ibis[11]-Middleware-Plattform und unterstützt die Auslagerung bestimmter Funktionen einer mobilen Anwendung in die Infrastruktur auf der Ebene einzelner Methoden.

Funktionsweise Der Lösungsansatz setzt voraus, dass der Quellcode, der auszulagernden Anwendung durch den Anwendungsentwickler so angepasst wird, dass für jede auszulagernde Methode jeweils eine Implementierung für die lokale Ausführung und eine Implementierung für die Ausführung auf einem Surrogate durch den Entwickler zur Verfügung gestellt wird. Die Entscheidung über die Adaption in Form der Auslagerungsentscheidung stützt sich auf die Erreichbarkeit eines Surrogates, die laufend geprüft wird, um der schnell

[10] https://eclipse.org/
[11] http://www.cs.vu.nl/ibis/

wechselnden Konnektivität mobiler Umgebungen Rechnung zu tragen. Zusätzlich werden Mechanismen zur Fehlerbehandlung bereitgestellt. Im Hinblick auf die Steuerung der Auslagerung können Adaptionsziele wie die Einsparung von Energie oder die Minimierung der Ausführungszeit gesetzt werden.

Cuckoo integriert sich in das von der Android-Systemplattform umgesetzte Konzept der Trennung der Präsentationsschicht, in Form der Android *Activities,* die für die Interaktion mit dem Nutzer verantwortlich sind, und der Ausführungsschicht in Form der Android-*Services*, den Aufgaben, die die Geschäftslogik implementieren. Letztere bieten sich im Rahmen der Nutzung von Cuckoo als Kandidaten für die Auslagerung an, wie im in Abbildung 4.10 veranschaulichten Auslagerungsprozess dargestellt wird.

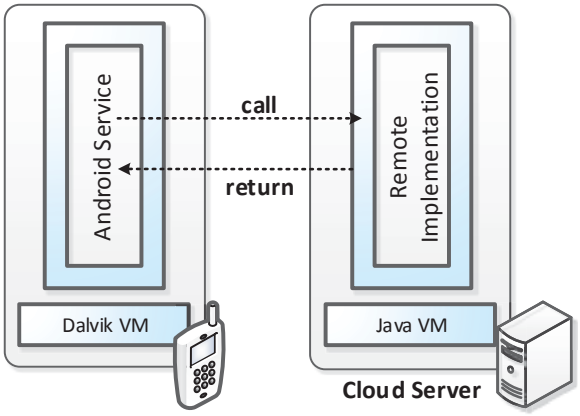

Abbildung 4.10.: Cuckoo: Prozess der Auslagerung

Da der Entwickler dazu aufgefordert ist, für jede entfernt auszuführende Methode eine zusätzliche Implementierung bereitzustellen, wird es möglich, diese bewusst von der lokalen Implementierung zu entkoppeln, um so eine für die Ausführung auf Surrogates speziell angepasste Implementierung zu erhalten, die beispielsweise von deren erweiterten Multithreading-Fähigkeiten dieser Geräte Gebrauch macht und so eine erweiterte Parallelverarbeitung innerhalb einzelner Teile einer mobilen Anwendung ermöglicht.

Bewertung Der vorgestellte Lösungsansatz fokussiert in erster Linie eine gute Integration in die Systemarchitektur eines weitverbreiteten Betriebssystems für mobile Geräte und integriert sich ebenfalls gut in den entsprechenden Entwicklungsprozess mobiler Anwendungen auf dieser Plattform, was in einer entsprechend guten Bewertung der Nutzbarkeit (U1-U4) und Wartbarkeit (M1-M3) dieser Lösung resultiert, da diese Schritte zusätzlich durch ein entsprechendes Plugin für die Entwicklungsumgebung Eclipse unterstützt werden.

Trotz der Anforderungen an die Entwickler, einen Teil der Methoden doppelt zu implementieren, ergeben sich durch diesen Ansatz eine Reihe von Vorteilen, aus denen insbesondere die gute Bewertung der Skalierbarkeit (S1-S4) dieser Lösung resultiert, da eine Parallelisierung der auszulagernden Funktionalität

erfolgt, für die automatisch eine entsprechende Anzahl von Surrogates verwendet werden.

Die Portierbarkeit (P1-P6) der Lösung kann hingegen nur als durchschnittlich bewertet werden, da ein beträchtlicher Anteil einer mobilen Anwendung doppelt implementiert werden muss. Zusätzlich besteht hierdurch die Gefahr, dass selbst bei einer identischen Implementierung der lokalen und der Remote-Implementierung nicht sichergestellt ist, dass auch dieselben Bibliotheken und dieselbe Version der Ausführungsumgebung in Form der Java-Virtual-Machine Verwendung finden, wodurch die lokale und die Remote-Ausführung zu unterschiedlichen Ergebnissen führen kann.

In Bezug auf die Verfügbarkeit (A1-A4) kann diese Lösung hingegen wiederum als gut bewertet werden, da sie entsprechende Mechanismen bereitstellt, um Verbindungsabbrüche und Fehler in der Ausführung zu erkennen und die entsprechenden Aufgaben anderen Surrogates zuzuweisen oder eine lokale Ausführung zu initiieren. Sicherheitsaspekte sind (SE1-SE2) von den Autoren nicht direkt adressiert, Entwickler können jedoch Methoden als nicht für die Auslagerung verfügbar deklarieren.

4.4.4. Feingranulare verteilte Ausführung

eXCloud [MLW11] unterstützt die verteilte Ausführung mobiler Anwendungen dadurch, dass der für die auszulagernde Funktionalität relevante Teil des Stacks dieser Anwendung zur Laufzeit auf ein Surrogate in der Infrastruktur übertragen wird.

Funktionsweise Für diese Art der Auslagerung stellt eXCloud eine Ausführungsschicht (execution layer) zwischen der mobilen Anwendung und der darunterliegenden Ausführungsumgebung bereit. Diese zusätzliche Ausführungsschicht wird von Ma et al. in [MLW11] als *Stack-on-Demand Execution Engine (SODEE)* bezeichnet und erlaubt dem in Abbildung 4.11 dargestellten Auslagerungsprozess auf Basis kleinster Einheiten des Programmcodes einer mobilen Anwendung.

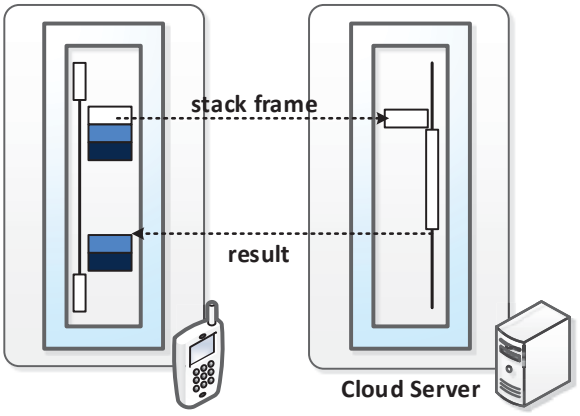

Abbildung 4.11.: eXCloud: Abbildung einer Auslagerung

Weitere Komponenten, die für diese Art der Auslagerung relevant sind, sind der *Migration Manager*, der auf den Surrogates für die Auslagerung einer bestimmten Einheit des Programmcodes verantwortlich ist, der *Worker Manager*, verantwortlich für den Empfang dieses auszulagernden Abschnitts, und der *Communication Manager*, verantwortlich für die allgemeine Koordination zwischen dem mobilen Gerät und den Surrogates.

Bewertung Obwohl diese Lösung auf offenen Standards basiert, kann die Wartbarkeit (M1-M3) jedoch nur als moderat charakterisiert werden, da das Debugging durch die vollständige Verteilungstransparenz, die der SODEE-Layer herstellt, eine Reihe von Herausforderungen mit sich bringt. Die Nutzbarkeit (U1-U3) dieses vollkommen transparenten Lösungsansatzes kann hingegen als sehr hoch bewertet werden. Diesem Ansatz folgend kann die Portierbarkeit (P1-P6) dieser Lösung insgesamt als durchschnittlich bewertet werden, da eine entsprechende Ausführungsumgebung inklusive aller Bibliotheken auf den Surrogates vorausgesetzt wird. Weichen diese von den auf dem mobilen Gerät verwendeten Bibliotheken ab, kann dies dazu führen, dass die Ausführung desselben Bytecodes zu unterschiedlichen Ergebnissen führt. Weiterhin existiert bei dieser Lösung keine spezielle Funktionalität zur Abbildung einer konkreten Entscheidungslogik im Hinblick auf das Ziel einer bestimmten Auslagerung; zudem erfordert dieser Ansatz, den Stack des Surrogates entweder laufend oder jeweils vor der Ausführung eines auszulagernden Teils des Programmcodes mit dem mobilen Gerät zu synchronisieren, was unter Umständen zu hohen Synchronisierungsaufwänden führen kann. Die Skalierbarkeit (S1-S4) der Lösung kann entsprechend als durchschnittlich bewertet werden. Die Verfügbarkeit (A1-A4) hingegen kann nur als gering bezeichnet werden, da in [MLW11] keine explizite Berücksichtigung des Kontextes oder Routinen zur Fehlerbehandlung erwähnt werden. Ebenso finden die Sicherheitsaspekte (SE1-SE2) der Lösung keine entsprechende Erwähnung.

4.4.5. Bewertung weiterer Lösungsansätze

Entsprechend der vorgestellten Bewertungsmethodik wurden die übrigen der 40 ausgewählten Lösungsansätze gegen den vorgestellten Kriterienkatalog evaluiert. Tabelle 4.2 fasst die Ergebnisse dieser Bewertung zusammen.

Nutzbarkeit und Wartbarkeit Die vorhergehende Analyse zusammenfassend soll zunächst festgehalten werden, dass die Nutzbarkeit und Wartbarkeit der sechs untersuchten Kategorien von Lösungsansätzen als vergleichbar gut bewertet werden können. Dies kann dadurch begründet werden, dass eine Vielzahl der Lösungen im Hinblick darauf entwickelt wurde, sich gut in den üblichen Entwicklungsprozess mobiler Anwendungen und die hierfür verwendeten Werkzeuge zu integrieren. In Bezug auf die einfache Benutzbarkeit (U1) muss jedoch festgehalten werden, dass diese insbesondere bei Lösungen, die nicht explizit für mobiles Cloud Computing entworfen wurden, entsprechend eingeschränkt ist.

Portierbarkeit und Skalierbarkeit Die Portierbarkeit der mit den entsprechenden Lösungen erstellten Anwendungen kann über alle Kategorien hinweg als gleichermaßen gut angesehen werden, was darin begründet sein kann, dass eine Vielzahl dieser Ansätze auf bewährte Paradigmen und Technologien aus dem Bereich der verteilten Systeme aufsetzen. Insbesondere die Kriterien der Koexistenz (P4) und des Deployments (P5) können hier als überdurchschnittlich gut bewertet werden, wohingegen sowohl die Anforderungen an einfache Adaptionsmechanismen (P2) als auch an erweiterte Adaptionsszenarien (P3) noch als verbesserungswürdig angesehen werden müssen.

Mit Bezug auf die Fähigkeit zur Kontextadaption soll an dieser Stelle festgehalten werden, dass nur wenige Lösungen, insbesondere auch nur wenige der explizit für mobiles Cloud Computing entwickelten Lösungen, entsprechende Mechanismen vorsehen. Unter Berücksichtigung der in Abschnitt 4.1 aufgeführten Anwendungsszenarien wird jedoch ersichtlich, dass diese von einer Unterstützung entsprechend stark profitieren würden.

Die Skalierbarkeit der untersuchten Lösungen kann insgesamt als durchschnittlich bewertet werden. Insbesondere die Dienstsuche (S2) ist dabei oft nicht auf die spontane Integration neuer Ressourcen vorbereitet, wodurch eine opportunistische Ressourcennutzung nur eingeschränkt möglich ist.

Verfügbarkeit und Sicherheit In Bezug auf die vorgenannten Einschränkungen der Portabilität und der Skalierbarkeit ist ein weiteres entscheidendes Kriterium für die Nutzung mobiler Anwendungen im mobilen Cloud Computing ihrer Verfügbarkeit (A1-A4). Um die Nutzbarkeit einer Anwendung sicherzustellen, sind insbesondere Funktionalitäten zur Fehlerbehandlung (A2) und zur Vorhersage zukünftiger Kontextzustände (A3), zum Beispiel die Konnektivität und mögliche Verbindungsabbrüche zu Surrogates, erforderlich, die allerdings nur ansatzweise und von einer geringen Zahl der untersuchten Lösungsansätze berücksichtigt werden.

Im Hinblick auf Sicherheitsaspekte zeigt sich in allen untersuchten Kategorien von Lösungen nur eine sehr begrenzte Berücksichtigung der in der Anforderungsanalyse ermittelten Kriterien.

Zusammenfassung der Bewertung Zusammenfassend kann damit festgehalten werden, dass sich existierende Arbeiten aus dem Themengebiet des mobilen Cloud Computings sowie aus verwandten Themengebieten bisher auf eine Verbesserung der Nutzbarkeit konzentriert haben. Unter Berücksichtigung des Adaptionszieles der Reduktion des Energieverbrauchs und der Ausführungszeit setzen insbesondere die für das mobile Cloud Computing entwickelten Lösungsansätze als Adaptionsform auf eine Auslagerung der Berechnungen. Zusätzliche Aspekte der Portierbarkeit und Skalierbarkeit mobiler Anwendungen sind dabei allerdings noch weitgehend unberücksichtigt.

Ebenso sind weitergehende Adaptionskonzepte, die eine Berücksichtigung des aktuellen sowie auch des zukünftigen Kontexts einer mobilen Anwendung ermöglichen, bisher ebenso nur ansatzweise von den existierenden Lösungsan-

		A1	A2	A3	A4	P1	P2	P3	P4	P5	P6	S1	S2	S3	S4	U1	U2	U3	U4	M1	M2	M3	SE1	SE2
Spezialisierte Sprachen	AmbientTalk [VCMB+07]	-	++	-	++	-	+	-	++	++	0	0	+	++	++	-	++	0	0	+	+	0	0	0
	Kairos [GGG05]	-	-	-	0	-	-	-	-	-	+	-	0	0	0	+	++	0	+	-	++	0	0	0
	Pleiades [KGMG07]	-	0	-	-	-	+	-	-	-	-	-	0	0	-	+	+	0	+	-	-	-	0	0
Frameworks und Middleware	Agilla [FRL05]	-	-	0	0	-	-	-	0	0	0	++	-	-	-	-	-	-	0	0	-	-	0	0
	ASM [SG14]	0	0	-	0	++	-	-	0	++	-	++	0	0	0	++	++	++	++	++	++	-	-	-
	CoDAMoS [CoD03, PPRB06]	-	-	+	-	+	++	++	++	0	++	0	++	0	0	++	0	++	++	++	-	0	0	0
	Hyrax [Mar09]	+	-	-	++	++	++	+	0	0	++	-	-	++	++	+	+	0	++	0	0	++	++	0
Verteilte VMs	COMET [GJM+12]	++	++	-	-	++	+	-	++	++	++	++	-	++	-	++	++	++	++	+	++	+	-	+
	Jessica2 [ZWL02]	0	-	-	++	++	-	-	++	++	++	+	0	++	+	++	++	++	++	++	++	++	0	0
Pervasive und Ubiquitous Computing	Chroma/Vivendi [BGSH07]	++	++	0	0	++	++	0	++	++	++	++	++	++	0	+	++	+	++	++	++	-	-	-
	CADeComp [ATBB08]	-	-	-	-	+	-	-	++	++	++	0	0	0	++	-	+	++	0	++	++	++	0	0
	CAwbWeb [BGHM10]	-	0	-	-	-	-	-	0	++	++	-	0	-	-	-	+	++	++	0	0	++	-	-
	GAIA [RHC+02, CAMCM05]	-	+	-	++	-	+	-	++	++	++	++	++	0	++	++	++	0	++	++	0	-	++	++
	Ghiani [GPP+12]	-	-	+	++	-	++	++	++	++	-	-	++	-	0	+	++	++	+	++	+	++	+	++
	MDAgent [YCWL06]	-	-	-	-	++	-	-	0	++	++	0	+	0	0	-	++	++	0	++	++	++	++	0
	MOB-Aware [LLY+10]	++	-	+	-	++	+	+	0	0	0	++	-	-	-	++	+	++	-	++	++	0	0	0
Anwendungsarchitekturen für mobiles Cloud Computing	AIOLOS [VSDTD12]	++	++	0	0	++	++	0	++	++	++	++	0	0	0	++	++	++	0	++	++	++	++	0
	AlfredO [RRA08]	-	-	-	-	++	-	-	++	-	-	++	-	-	-	+	+	++	++	++	++	++	++	0
	CMH Application [MRST10]	0	0	0	0	++	-	-	++	++	++	-	0	0	0	++	++	+	0	0	++	++	0	0
	Cuckoo [KPKB10]	++	++	+	++	++	++	+	++	+	++	++	+	++	++	++	++	++	++	++	++	++	+	-
	exCloud [MLW11]	0	++	0	0	++	0	0	++	++	++	++	0	++	0	++	++	++	0	-	-	0	0	0
	Giurgiu et al. [GRA12]	-	-	+	0	++	++	++	++	++	++	++	++	-	++	++	++	++	++	++	++	++	0	0
	Gu et al. [GNM+03]	-	-	-	++	++	-	-	++	++	++	++	++	-	+	++	++	++	+	++	++	0	0	0
	Huerta-Can. et al. [HCL10]	-	++	+	++	++	++	+	+	++	+	++	++	-	-	++	++	++	0	++	+	++	++	++
	IC CLOUD [SPN+13]	-	0	++	-	++	++	++	0	0	0	++	-	++	-	++	++	++	0	++	++	0	++	0
	MOCHA [SMF+12]	-	-	-	-	+	-	-	+	-	++	0	0	+	-	0	0	0	++	0	0	0	0	0
	NAM4J [GPZA13]	0	0	-	++	++	+	-	++	++	++	++	++	-	0	++	++	++	-	++	++	++	-	-
	Odessa [RSM+11]	++	++	++	0	++	++	++	++	-	-	++	++	++	0	++	++	0	++	++	++	0	0	0
	Scavenger [Kri10]	0	0	-	0	++	++	-	++	++	++	++	++	++	0	-	++	++	++	++	++	++	++	+
	Zhang [ZKJG11]	0	-	-	0	++	++	-	++	++	++	++	-	++	++	-	-	++	++	++	++	++	-	-
	müCloud [MGL+11]	-	-	-	0	++	-	-	++	++	-	0	0	-	0	++	++	++	+	++	++	-	-	-
Systemunterstützungen für mobiles Cloud Computing	AgentISR [LCTB+06]	++	-	++	-	++	+	-	++	++	0	-	0	++	0	++	++	++	++	++	++	++	++	++
	Chen et al. [COH12]	-	-	-	+	++	-	-	-	++	++	++	-	0	+	++	++	++	0	++	++	++	-	-
	Cirrus Clouds [SLAZ12]	-	++	++	0	++	++	++	++	++	++	++	++	++	0	++	++	++	++	+	-	-	0	0
	CloneCloud [CIM+11]	0	0	-	0	++	-	-	++	++	++	+	+	-	0	++	++	++	++	+	-	++	-	0
	Jupiter [GZK+11]	+	-	-	0	+	0	0	++	-	++	++	0	0	0	++	++	0	++	++	++	++	0	0
	MAUI [CBC+10]	++	-	+	++	++	++	+	++	++	++	-	+	-	++	++	++	++	++	++	++	-	++	0
	Slingshot [SF05]	++	+	-	++	++	-	-	++	++	++	++	+	-	+	+	+	++	-	++	++	++	-	-
	ThinkAir [KAH+12]	++	++	+	++	++	++	+	++	+	++	++	+	++	++	+	++	++	++	++	++	++	+	-
	VEE Hung et al. [HCS13]	0	-	-	++	++	++	0	++	++	++	0	0	++	0	-	-	++	++	++	-	0	++	++

++ vollständig erfüllt + teilweise erfüllt - nicht erfüllt 0 nicht adressiert

Tabelle 4.2.: Bewertung existierender Lösungsansätze

sätzen abgedeckt. Abschließend ist zusätzlich festzuhalten, dass es sich größtenteils um Forschungsarbeiten handelt und keine der untersuchten Lösungen eine marktreife Software darstellt.

4.5. Zusammenfassung

In diesem Kapitel wurden zunächst auf Basis charakteristischer Anwendungsfälle für mobiles Cloud Computing die Anforderungen an eine entsprechende kontextadaptive Anwendungsarchitektur erhoben und in Form eines Anforderungskataloges festgehalten, wobei jeweils die existierenden Problemstellungen aufgezeigt und entsprechende Lösungsalternativen diskutiert wurden. Gegen den entwickelten Anforderungskatalog wurden anschließend existierende Lösungsansätze aus verschiedenen angrenzenden Forschungsgebieten bewertet und im Hinblick auf die Erweiterbarkeit zur Unterstützung kontextadaptiver Adaptionsszenarien hin untersucht. Im Zusammenhang mit dieser Analyse wurden anschließend vielversprechende existierende Lösungsansätze für das mobile Cloud Computing näher untersucht. Hierbei wurde festgestellt, dass insbesondere die Fähigkeit zur Kontextadaption von bestehenden Lösungen nur in Ansätzen berücksichtigt wird.

Aufbauend auf den in diesem Kapitel entwickelten Anforderungen soll im nun folgenden Kapitel 5 die Entwicklung eines eigenen Lösungsansatzes fokussiert werden.

5. Entwicklung einer kontextadaptiven Anwendungsarchitektur

Nachdem im vorhergehenden Kapitel 4 anhand typischer Klassen von Anwendungsfällen die Anforderungen an eine kontextadaptive Anwendungsarchitektur für das mobile Cloud Computing erhoben wurden, wird in diesem Kapitel der konzeptionelle Entwurf einer entsprechenden Architektur erarbeitet. Hierzu erfolgt zunächst die Ableitung einer Basisarchitektur, die die in Kapitel 4 ermittelten generellen Anforderungen abdeckt.

Anschließend werden aus den in Abschnitt 4.1 aufgeführten Gruppen von Anwendungsbeispielen hieraus resultierende Adaptionsbedarfe mobiler Anwendungen ermittelt. Diese werden zunächst abstrakt anhand verschiedener Formen der Adaption beschrieben, bevor die jeweiligen Dimensionen dieser Adaption weiter detailliert und daraus die hiermit verbundenen Adaptionsziele abgeleitet werden. Daraufhin werden in Abschnitt 5.3 relevante Teilprobleme und entsprechende Lösungsansätze näher untersucht, die sich bei der Entwicklung einer entsprechenden Architektur ergeben und im Hinblick auf die Entwicklung einer kontextadaptiven Anwendungsarchitektur von besonderer Relevanz sind.

In Abschnitt 5.4 wird anschließend die in Abschnitt 5.1 entwickelte Basisarchitektur für mobiles Cloud Computing als Grundlage für die Entwicklung einer eigenen kontextadaptiven Anwendungsarchitektur und einer entsprechenden Systemunterstützung verwendet, die sich an den im Rahmen der Anforderungsanalyse erhobenen qualitativen Kriterien orientiert. In diesem Zusammenhang werden die wesentlichen funktionalen sowie nicht-funktionalen Eigenschaften der entwickelten Infrastrukturkomponenten einer Systemunterstützung vorgestellt.

In einem weiteren Schritt werden, die entwickelte Architektur ergänzend, bestehende Adaptionsmechanismen für kontextadaptive Systeme im Hinblick auf ihre Passfähigkeit in Bezug auf die in Abschnitt 5.2 definierten Adaptionsformen und Adaptionsziele bewertet. Passend zur entwickelten Anwendungsarchitektur wird im Anschluss in Abschnitt 5.5 ein generischer Prozess zur Kontextadaption entwickelt, welcher vorhandene Kontextinformationen verarbeitet und im Hinblick darauf aufbereitet, dass sie von der entwickelten Anwendungsarchitektur zur Erstellung von Adaptionsstrategien genutzt werden können. Hierzu werden sowohl etablierte Prozesse für die Kontextverarbeitung genutzt als auch existierende Lösungsansätze und Forschungsarbeiten im Bereich der Kontextprognose integriert, um eine Proaktivität der Adaption zu ermöglichen. Der entwickelte Prozess wird anschließend auf den konkreten Anwendungsfall der Kontextadaption innerhalb des mobilen Cloud Computings angewendet.

5.1. Basisarchitektur für mobiles Cloud Computing

Aus der Untersuchung existierender Lösungsansätze in Abschnitt 4.1 lassen sich einige Gemeinsamkeiten ableiten, aus denen im nun folgenden Abschnitt eine Basisarchitektur für mobiles Cloud Computing abgeleitet werden soll. Die Basisarchitektur konzentriert sich zunächst, analog den untersuchten Arbeiten, auf Auslagerungsprozesse als Adaptionsform. Die Basisarchitektur soll in diesem Zusammenhang gleichzeitig die im Rahmen der Anforderungsanalyse ermittelten Kriterien für eine kontextadaptive Anwendungsarchitektur abbilden.

5.1.1. Adaptionsprozess

Die Betrachtung der in den existierenden Lösungsansätzen umgesetzten typischen Adaptionsprozesse zeigt eine Reihe von Gemeinsamkeiten auf. Die von den existierenden Lösungen umgesetzten Auslagerungsprozesse dienen der Entlastung der begrenzten Ressourcen eines mobilen Gerätes und verschieben hierzu einen Teil der Funktionalität einer mobilen Anwendung von einem mobilen Gerät auf ein Surrogate in der Infrastruktur. Ein typischer Auslagerungsprozess, dargestellt in Abbildung 5.1, umfasst dabei die folgenden Schritte:

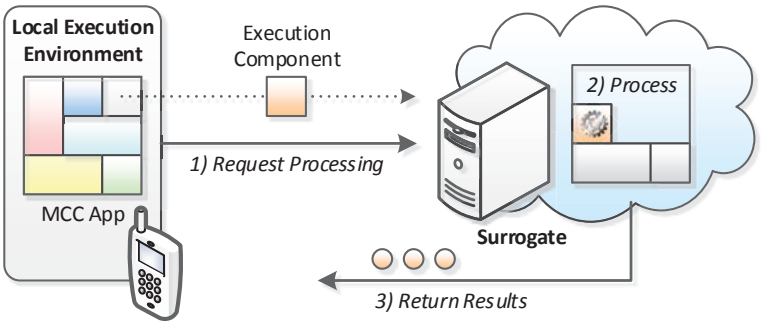

Abbildung 5.1.: Typischer Auslagerungsprozess im mobilen Cloud Computing

Profiling und Partitionierung Im ersten Schritt wird eine mobile Anwendung entweder zur Entwurfszeit durch eine entsprechende Analyse des Programmcodes oder zur Laufzeit durch ein Profiling der Anwendung im Zusammenhang mit typischen Nutzungsmuster dahingehend untersucht, ob sich ein Teil oder konkreter eine bestimmte Funktionalität dieser Anwendung für eine Auslagerung in die Infrastruktur eignet.

Das Ergebnis dieser Untersuchung liefert eine Partitionierung, die die jeweils für die Auslagerung geeigneten Teile des Programmcodes voneinander abgrenzt. Diese Abgrenzungen sind dabei typischerweise so angelegt, dass die im Rahmen der späteren Auslagerung der Funktionalität zu übertragende Datenmengen, der hierfür relevante Teil des Ausführungszustands, möglichst gering ausfällt. Entsprechend besteht eine mobile Anwendung für den

Einsatz im mobilen Cloud Computing aus mehreren Partitionen, die üblicherweise durch Aufteilung einer existierenden mobilen Anwendung definiert werden [CIM+11, CBC+10] oder die, falls die mobile Anwendung bereits entsprechend modular entwickelt wurde, durch die entsprechenden Funktionsbausteine (building blocks), beispielsweise Komponenten [GRJ+09, GRA12], oder einer Zusammenfassung dieser Komponenten definiert sind.

Integration der Ressourcen Beim Start einer mobilen Anwendung gilt es zunächst, die für eine Auslagerung nutzbaren Surrogates innerhalb der Infrastruktur zu erkennen und diese gegebenenfalls für die Ausführung einer bestimmten Funktionalität einer mobilen Anwendung vorzubereiten. Diese Vorbereitung umfasst üblicherweise den Transfer des für die Ausführung der Funktionalität notwendigen Programmcodes sowie gegebenenfalls weitere Daten. Insbesondere bei einer verteilungstransparenten Ausführung dieser Funktionalität umfasst dies mitunter die initiale Synchronisation des gesamten Ausführungskontextes der mobilen Anwendung [SGKB13].

Abhängig davon, ob der Verbindungsstatus des mobilen Gerätes wechselt, kann es erforderlich sein, diesen Schritt periodisch oder in Abhängigkeit von der Konnektivität der einzelnen Schnittstellen für die Mobilkommunikation zu wiederholen.

Auslagerungsentscheidung Wird im Laufe der Ausführung der mobilen Anwendung eine Funktionalität erkannt, die im Hinblick auf ein gewähltes Adaptionsziel, zum Beispiel die Einsparung von Energie, nicht auf dem mobilen Gerät ausgeführt werden soll und sich die entsprechende Partition für eine Auslagerung in die Infrastruktur eignet, wird die sogenannte Auslagerungsentscheidung (offloading decision) getroffen.

Hierzu wird einerseits die im Rahmen des Profiling ermittelte Ausführungsstatistik herangezogen und andererseits der aktuelle Kontext des mobilen Gerätes berücksichtigt, um abzuschätzen, ob die beabsichtigte Auslagerung vorteilhaft erscheint. Dies umfasst beispielsweise die verfügbare Bandbreite der Schnittstellen für die Mobilkommunikation und die Größe des für die Ausführung der Funktionalität zu übertragenden Ausführungszustands der mobilen Anwendung.

Auslagerung Lässt die Auslagerungsentscheidung eine Ausführung in der Infrastruktur sinnvoll erscheinen, wird der Kontrollfluss zusammen mit dem relevanten Teil des Ausführungszustands für die auszulagernde Funktionalität auf dem mobilen Gerät angehalten, erfasst, auf das für die Auslagerung bestimmte Surrogate übertragen und die Ausführung des zuvor gehaltenen Kontrollflusses fortgesetzt. Nach Durchlaufen dieses für die Auslagerung bestimmten Teils der Geschäftslogik wird das Ergebnis der Auslagerung, in Form eines veränderten Ausführungszustands, zurück auf das mobile Gerät transferiert und die Ausführung des Kontrollflusses dort wieder aufgenommen.

In bereits modular entwickelten mobilen Anwendungen, bei denen die einzelnen Partitionen auch bei lokaler Ausführung über Dienstaufrufe (service calls) miteinander kommunizieren, ist diese Art der Synchronisation entsprechend einfacher und kann mithilfe eines entfernten Dienstaufrufs abgebildet werden.

5.1.2. Komponenten einer Systemunterstützung

Um den zuvor beschriebenen Adaptionsprozess umzusetzen, verwenden die untersuchten Lösungsansätze eine Reihe zusätzlicher Komponenten, die zunächst unter dem Begriff der *Systemunterstützung* zusammengefasst werden sollen. Diese Infrastrukturkomponenten[1] weichen je nach Lösung voneinander ab, es lassen sich bei entsprechender Abstraktion der konkreten Funktionalität jedoch die folgenden Komponenten identifizieren, die üblicherweise im Rahmen eines Auslagerungsprozesses benötigt werden und in Abbildung 5.2 veranschaulicht sind:

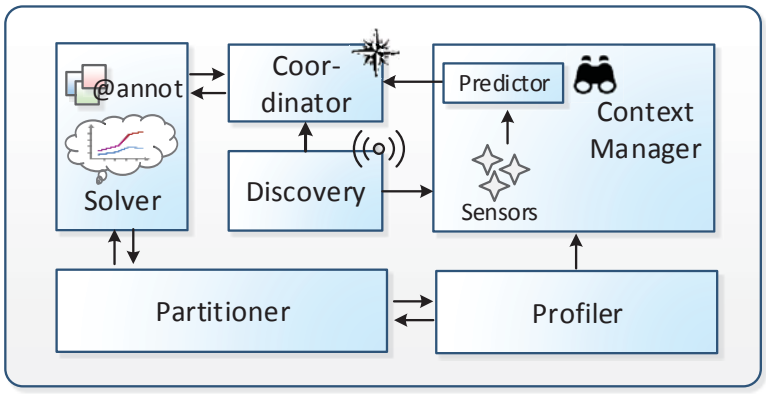

Abbildung 5.2.: Typische Infrastrukturkomponenten einer Systemunterstützung

- Ein *Profiler*, verantwortlich für die Analyse der mobilen Anwendung im Hinblick auf die für die Auslagerung einer bestimmten Funktionalität relevanten Parameter.
- Ein *Partitioner*, der die für eine Auslagerung in die Infrastruktur geeigneten Funktionalitäten von angrenzenden Teilen des Kontrollflusses einer mobilen Anwendung abgrenzt und hierdurch Partitionen bildet, die sich prinzipiell für eine Auslagerung qualifizieren.
- Ein *Discovery Service*, der für die Suche und Integration neuer Ressourcen verantwortlich ist und diesen Prozess in Abhängigkeit von der Konnektivität des mobilen Geräts regelmäßig wiederholt und hierdurch aktuelle Informationen zu verfügbaren Surrogates bereitstellt.

[1] In Abweichung zu der in Abschnitt 2.2.6 zitierten Definition von Roth in [Rot02] sollen in dieser Arbeit Infrastrukturkomponenten als die Komponenten der Systemunterstützung der entwickelten kontextadaptiven Anwendungsarchitektur verstanden werden. Eine dieser Komponenten ist beispielsweise der Kontextmanager.

- Ein *Kontextmanager*, der alle relevanten Parameter des Kontextes eines mobilen Gerätes erfasst, sammelt und gegebenenfalls prognostiziert, um die für die Auslagerungsentscheidung relevanten Kontextinformationen bereitzustellen.
- Ein *Solver*, der die Informationen des Kontextmanagers und des Discovery Service nutzt, um zur Laufzeit zu entscheiden, ob und, wenn ja, zu welchem Surrogate die für die Auslagerung ermittelten Funktionen ausgelagert werden können.
- Einen *Koordinator*, der übergreifend die Steuerung und Abwicklung (Synchronisation, senden, empfangen, erneute Synchronisation) der Auslagerung übernimmt.

5.2. Adaptionsbedarfe kontextadaptiver mobiler Anwendungen

Neben einer Reihe spezieller Adaptionsstrategien bestimmter mobiler Anwendungen existieren einige allgemeine Adaptionsbedarfe, von denen angenommen wird, dass eine große Zahl mobiler Anwendungen von ihrer Berücksichtigung profitieren. Hierzu wurden in Abschnitt 4.1 eine Reihe repräsentativer Klassen von Anwendungsbeispielen und entsprechende existierende mobilen Anwendungen vorgestellt. Die konkreten Adaptionsbedarfe dieser Anwendungsbeispiele sollen im Folgenden näher untersucht werden. Hierzu werden zunächst die konkreten Formen der Adaption und im Anschluss die daraus abgeleiteten Dimensionen und Ziele der Adaption näher vorgestellt.

5.2.1. Formen der Adaption

Insbesondere wenn es um die Bearbeitung ressourcenintensiver Aufgaben geht, kann ein Großteil mobiler Anwendungen nur mit Einschränkungen oder gar nicht auf mobilen Geräten ausgeführt werden, ohne dass diese eine Unterstützung durch eine Infrastruktur erfahren. Obwohl die Anforderungen in Bezug auf die Adaption einzelner mobiler Anwendungen für das mobile Cloud Computing teilweise stark variieren, kann festgehalten werden, dass die Mehrheit dieser mobilen Anwendungen von einer Kooperation mit der umgebenden Infrastruktur profitiert. Dabei können mit Blick auf die Anwendungsszenarien die vier folgenden Varianten unterschieden werden. Diesen Varianten ist gemeinsam, dass sie die Geschäftslogik einer Anwendung nicht in Abhängigkeit vom Kontext verändern, sondern nur der Ort und die Zeit der Ausführung kontextabhängig angepasst werden:

- Auslagern von Geschäftslogik: Für eine Auslagerung einer bestimmten Funktionalität ist es erforderlich, einen Teil der Geschäftslogik einer mobilen Anwendung zu entkoppeln und ihn in die Infrastruktur zu verlagern. Dieser Anwendungsfall stellt den wohl prominentesten im Bereich

des mobilen Cloud Computings dar und wird als *computation offloading* beschrieben (vergleiche Abschnitt 2.4.2). Offloading dient dabei in erster Linie dazu, die Ressourcen eines mobilen Geräts zu entlasten, indem es eine üblicherweise rechen-oder speicherintensive Aufgabe vom mobilen Gerät in die Infrastruktur auslagert und nach dessen Beendigung das Ergebnis zurück auf das mobile Gerät transferiert wird (vergleiche erste Problemstellung des mobilen Dilemmas in Abschnitt 2.2.9).

- Einlagern von Geschäftslogik: Neben der Auslagerung existiert jedoch ein weiterer, in der Literatur jedoch deutlich seltener erwähnter Fall, der den umgekehrten Fall der Auslagerung beschreibt und in dem die Daten oder die Geschäftslogik aus der Infrastruktur auf das mobile Gerät gezogen werden. Diese Adaptionsform ist insbesondere dann anzutreffen, wenn die Verfügbarkeit oder die Dienstqualität einer mobilen Anwendung erhöht oder aufrechterhalten werden soll, obwohl die Konnektivität des mobilen Geräts zur Infrastruktur eingeschränkt ist. Hierbei können einerseits Daten zwischengespeichert, also vorher auf das mobile Gerät transferiert (cachen) oder aber Teile der Geschäftslogik, die üblicherweise in der Infrastruktur liegen, auf das mobile Gerät transferiert werden (vergleiche: zweite Problemstellung des mobilen Dilemmas in Abschnitt 2.2.9). Dieses vorherige Abrufen (prefetching) von Daten kann entsprechend in Situationen erfolgen, in denen das mobile Gerät über eine gute Konnektivität zur Infrastruktur verfügt. Letzterer Anwendungsfall wurde im prominenten Beispiel des verteilten Dateisystems *Coda* [SKK+90] gezeigt.

- Verzögern von Geschäftslogik: Eine weitere mögliche Form der Adaption im mobilen Cloud Computing ist die Verzögerung eines bestimmten Teils der Geschäftslogik, was es je nach Art des Adaptionsmechanismus erforderlich machen kann, dass mobile Geräte insgesamt die betroffene mobile Anwendung oder Teile dieser anhalten. Diese Art der Adaption ist insbesondere dann anzutreffen, wenn der aktuelle Ausführungskontext hierfür nicht geeignet ist, beispielsweise weil geeignete Ressourcen oder benötigte Dienste in der Infrastruktur oder auf einem mobilen Gerät aktuell nicht verfügbar sind. Ein weiterer Fall für die Adaptionsform der Verzögerung sind Ressourcenbeschränkungen, beispielsweise eine zu geringe verbleibende Menge an Energie und Akkulaufzeit, die eine Ausführung verhindern.

- Vorziehen von Geschäftslogik: Das Vorziehen der Ausführung eines Teils der Geschäftslogik stellt im Vergleich zur Adaptionsform der Verzögerung weitaus höhere Anforderungen an die Proaktivität der Adaption. Diese Form der Adaption kann dabei einerseits in Form von einer vor Berechnung eines Ergebnisses (precomputation) oder das vorherige Abrufen von Daten oder einer Kombination aus beidem geschehen. Die Vorberechnung von Inhalten kann dabei als üblicherweise rechenintensiver Prozess bei-

spielsweise in Zeiträumen erfolgen, in denen die Energieressourcen des mobilen Geräts nicht beschränkt sind, weil das Gerät aktuell an eine externe Energiequelle angeschlossen ist.

Eine weitere Gruppe von Adaptionsformen konzentriert sich auf die kontextabhängige Veränderung der Geschäftslogik. Im Gegensatz zu den vier zuvor vorgestellten Adaptionsformen, die lediglich den Ausführungsort und die Ausführungszeit eines Teils der Geschäftslogik beeinflussen, bewirkt sie eine Anapassung des Kontrollflusses. Um diese Form der Anpassung sinnvoll durchführen zu können, ist es erforderlich, optionale oder austauschbare Teile der Geschäftslogik durch den Entwickler entsprechend markieren zu lassen. Abgesehen von dieser Erfordernis erlaubt dieser Ansatz ebenfalls eine transparente Ausführung, da keine weitergehende Adaptionslogik vorgegeben werden muss. Diese Gruppe weist somit die größte Ähnlichkeit zu den klassischen Adaptionsformen der kompositionellen Adaption auf (vergleiche Abschnitt 3.3.2) und kann entsprechend in die folgenden drei konkreten Adaptionsformen unterschieden werden:

- Hinzufügen von Geschäftslogik: Bei dieser Adaptionsform wird die Geschäftslogik einer mobilen Anwendung kontextabhängig ergänzt. Hierdurch kann die Dienstqualität einer mobilen Anwendung verbessert werden, wenn beispielsweise bei entsprechender Ressourcenverfügbarkeit Teile der Geschäftslogik ergänzt werden, um eine erweiterte Funktionalität bereitzustellen. Hierbei kann zum Beispiel die Suchfunktion innerhalb eines Text-Betrachters dahingehend erweitert werden, dass die ressourcenschonend ausgelegte Suchfunktion um eine Volltextsuche ergänzt wird, zusätzlich Wortähnlichkeiten berücksichtigt werden oder eine erweiterte Suchanfrage-Syntax unterstützt wird. Realisiert wird diese Adaptionsform oft über die Integration von als optional deklarierten Komponenten in die Geschäftslogik einer mobilen Anwendung, wie es unter anderem in [MMAM+14, PB05] gezeigt wird.

- Entfernen von Geschäftslogik: So wie im vorhergehenden Beispiel die Geschäftslogik einer mobilen Anwendung ergänzt werden kann, kann sie ebenfalls um optionale Komponenten reduziert werden, wenn diese beispielsweise mithilfe der aktuell verfügbaren Ressourcen nicht sinnvoll ausgeführt werden können. So kann ein optionaler Schritt im Rahmen der Verarbeitung eines Datenstroms gegebenenfalls weggelassen werden, wodurch die Dienstqualität einer mobilen Anwendung möglicherweise reduziert wird, jedoch die Dienstverfügbarkeit aufrechterhalten wird. Wie im Zusammenhang mit der Untersuchung sinnvoller Adaptionsziele in Abschnitt 5.2.3 gezeigt wird, ist die Entscheidung über das Entfernen von Geschäftslogik jedoch nur selten trivial. So gilt es häufig die Dienstqualität aufrecht zu erhalten, was es erforderlich macht, abzuschätzen, in welchem Fall das Entfernen eine andernfalls scheiternde Taskausführung zu einem erfolgreichen Abschluss führt.

- Austauschen von Geschäftslogik: Eine weitere Form der Adaption, die kontextabhängig den Kontrollfluss einer mobilen Anwendung beeinflussen kann, ist die des Austauschens der Implementierung eines Teils der Geschäftslogik einer mobilen Anwendung. Dies kann beispielsweise dadurch realisiert werden, dass in einer komponentenbasierten Anwendung eine Komponente durch eine andere ersetzt wird. Einerseits können so zwei unterschiedliche Implementierungen derselben Geschäftslogik gegeneinander ausgetauscht werden. Beispielsweise lässt sich eine speicher- gegen eine rechenintensive Variante austauschen. Andererseits kann hierdurch die Geschäftslogik selbst verändert werden und beispielsweise die Dienstqualität einer mobilen Anwendung bewusst herabgesetzt werden, um die Dienstverfügbarkeit aufrechtzuerhalten. Hierdurch kann beispielsweise die Auflösung eines Videodatenstroms zwischen zwei Geräten herabgesetzt werden, wenn die für die Übertragung benötigte Bandbreite nicht verfügbar ist. Diese Form der Anpassung wurde in Bezug auf das mobile Cloud Computing ursprünglich von Brian Noble in [NSN+97] als sogenannte Fidelity-Adaptation vorgeschlagen und anschließend in einer Reihe von Forschungsarbeiten erfolgreich eingesetzt, um den begrenzten Ressourcen mobiler Geräte Rechnung zu tragen [BGSH07].

In welcher Situation die jeweilige Form der Adaption oder eine Kombination von Adaptionsformen sinnvoll ist, wird im Zusammenhang mit der Erarbeitung der Ziele der Adaption (vergleiche Abschnitt 5.2.3) wieder aufgegriffen. Zunächst sollen hierfür die jeweiligen Adaptionsformen auf entsprechende Dimensionen der Adaption hin abstrahiert werden, um ein konzeptionelles Modell zu erarbeiten, welches auch Kombinationen einzelner Adaptionsformen sinnvoll unterstützt.

5.2.2. Dimensionen der Adaption

Nachdem zunächst eine Reihe von Adaptionsformen für den Anwendungsbereich des mobilen Cloud Computings als relevant identifiziert wurden, sollen diese in diesem Abschnitt zunächst weiter auf unterschiedliche Dimensionen der Adaption hin abstrahiert werden, bevor sie im Hinblick auf die in Abschnitt 4.1 betrachteten Gruppen von Anwendungsszenarien hin bewertet werden.

Mit Verweis auf die Adaptionsformen der *Auslagerung* und der *Einlagerung* von Funktionalität lässt sich diese ebenso als eine Verlagerung innerhalb der Dimension des Ausführungsortes beschreiben. Entsprechend lassen sich die *Verzögerung* und das *Vorziehen* einer Ausführung als eine Verschiebung entlang der zeitlichen Dimension beschreiben. Ebenso können die Adaptionsformen des *Hinzufügens*, *Entfernens* und *Austauschens* der Geschäftslogik einer Anwendung einer Verschiebung entlang der Dimension der Implementierung beschrieben werden.

Mit Bezug auf die im vorhergehenden Abschnitt vorgestellten sieben unterschiedlichen Adaptionsformen lassen sich diese entsprechend als Verschiebung

entlang der Dimensionen Zeit, Ort und Implementierung beschreiben, wie sie in Abbildung 5.3 veranschaulicht wird. Auf Basis dieser drei Dimensionen sollen nun im Folgenden die Anpassungsbedarfe innerhalb dieser Dimensionen im Zusammenhang mit den in Kapitel 4.1 eingeführten Gruppen von Anwendungsszenarien untersucht werden.

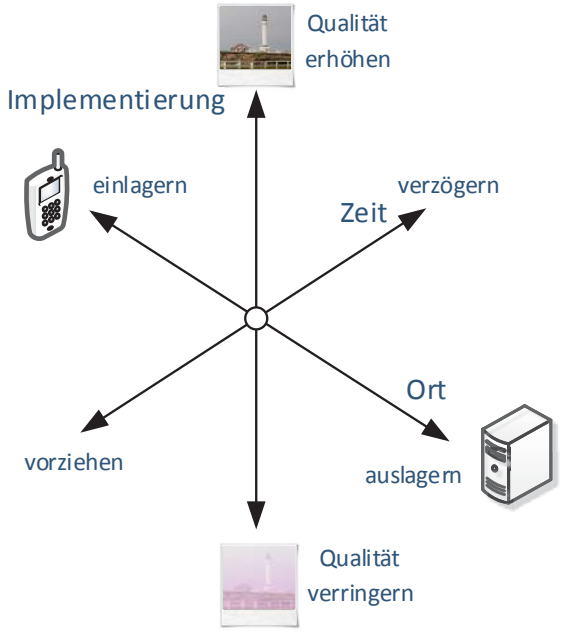

Abbildung 5.3.: Dimensionen der Adaption

Verarbeitung von Sensordaten Durch die aufgezeigten Leistungsbeschränkungen einzelner Sensorknoten profitiert diese Gruppe von Anwendungsszenarien in erster Linie davon, dass Teile der Speicherung und Verarbeitung von Sensordaten in die Infrastruktur verlagert werden können, um die begrenzten Speicher- und Rechenressourcen eines Sensorknotens bei einer längeren oder ständigen Nutzung nicht zu überlasten.

Dadurch, dass auf einem Sensorknoten selbst oft nur eine begrenzte Historie von Messwerten vorgehalten werden kann, kann diese Historie beim Bestehen einer entsprechenden Verbindung zur Infrastruktur in diese zur Speicherung übertragen werden, was eine Auslagerung der Daten im Sinne einer Veränderung des Speicherorts (Anpassung der Dimension Ort) entspricht. Hierdurch wird es möglich, innerhalb der Infrastruktur eine prinzipiell beliebig lange Historie von Messwerten vorzuhalten. Da diese Messwerte in einigen Fällen direkt nach ihrer Erfassung auch verarbeitet werden sollen, bietet es sich aufgrund der ebenfalls begrenzten Rechenressourcen des Sensorknotens an, ebenso die Verarbeitung der Sensordaten in die Infrastruktur zu verlagern und sie damit ebenso in der Dimension des Ausführungsortes zu verschieben.

Die genannten Leistungsbeschränkungen können es zusätzlich erforderlich machen, in Situationen mit eingeschränkter Konnektivität zur Infrastruk-

tur oder einer geringen verbleibenden Energiemenge eine Fidelity-Adaptation durchzuführen, um die Dienstverfügbarkeit bei gleichzeitiger Einschränkung der Dienstqualität aufrechtzuerhalten.

Von einer Anpassung innerhalb der zeitlichen Dimension profitieren diese Anwendungsfälle jedoch nur bedingt, da die Erfassung, Verarbeitung und Weiterleitung von Messwerten häufig zeitnah zu erfolgen hat. Ist dies nicht der Fall, kann diese Gruppe von Anwendungsfällen dadurch von einer Anpassung innerhalb der zeitlichen Dimension profitieren, dass bestimmte Aufgaben verzögert werden, bis Kontextattribute wie der verbleibende Energievorrat oder die Verbindungsqualität zur Infrastruktur eine erfolgreiche Ausführung der Geschäftslogik erwarten lassen.

Analyse von Audiodaten und Sprachassistenzsysteme Der Großteil der in der Gruppe von diesen Anwendungsszenarien charakterisierten Anwendungsfälle stützt sich auf eine direkte Verarbeitung der aufgezeichneten Audiodaten, wie beispielsweise bei der Simultanübersetzung gesprochener Sprache. Entsprechend profitiert diese Gruppe von Anwendungsfällen nur bedingt von einer Verschiebung innerhalb der zeitlichen Dimension.

In Bezug auf die Verschiebung des Ausführungsortes hingegen profitiert die oft rechenintensive Geschäftslogik der Spracherkennung häufig von einer Verschiebung auf leistungsstarke Surrogates, wodurch die ressourcenbeschränkten mobilen Geräte entlastet werden. Sind jedoch keine entsprechenden Surrogates für solch eine Auslagerung verfügbar, bietet sich alternativ oder zusätzlich eine Anpassung innerhalb der Dimension der Implementierung an. In diesem Fall kann ähnlich wie im vorgenannten Beispiel die Anpassung der Dienstqualität zur Aufrechterhaltung der Dienstverfügbarkeit genutzt werden.

Video und Bildverarbeitung Der Anwendungsbereich der Video- und Bildverarbeitung ist charakterisiert durch rechenintensive Operationen auf größeren Datenmengen. Entsprechend kann je nach Anforderung der Nutzer einerseits eine Verschiebung innerhalb der Dimension der Implementierung, im Sinne einer Fidelity-Adaptation, sinnvoll sein, um die Ressourcen eines mobilen Geräts zu entlasten. Andererseits kann für diese angestrebte Entlastung ebenso eine Verschiebung innerhalb der Dimension des Ausführungsortes angestrebt werden, um so in beiden Fällen die Dienstverfügbarkeit aufrechtzuerhalten und im Fall der Verlagerung der rechenintensiven Operationen in die Infrastruktur zusätzlich die Dienstqualität zu erhöhen. Mit Berücksichtigung der genannten Anwendungsszenarien zeigt sich zusätzlich, dass diese nur zum Teil zeitkritisch sind, womit sich ebenfalls eine Verschiebung innerhalb der zeitlichen Dimension anbietet.

Mobiles Cloud Gaming Im Bereich des mobilen Cloud Gamings kann für die überwiegende Zahl von Anwendungsfällen davon ausgegangen werden, dass

eine Verschiebung innerhalb der Dimension Zeit nur für wenige Anwendungsfälle von Nutzen ist. Allerdings kann es vor dem Hintergrund der in diesem Anwendungsfall genannten Anforderungen mit Blick auf die Latenz von Vorteil sein, Teile der Geschäftslogik in die direkte Infrastruktur zu verlagern oder die Dienstqualität anzupassen. Entsprechend bieten sich für diese Gruppe von Anwendungsfällen eine Anpassung der Geschäftslogik innerhalb der Dimensionen des Ausführungsortes oder der Implementierung an.

Augmented Reality Im Anwendungsbereich der Augmented Reality zeigt sich bei Betrachtung der angeführten Beispiele, dass eine Anpassung innerhalb der Dimension der Implementierung den beschränkten Ressourcen eines mobilen Geräts am deutlichsten entgegenkommt. Darüber hinaus kann ebenfalls eine Verschiebung eines Teils der Geschäftslogik in die Infrastruktur von Vorteil sein, wodurch die Dimension des Ausführungsortes als weitere für die Adaption vorteilhafte Dimension betrachtet werden kann. Ebenso wie im vorhergehenden Beispiel des mobilen Cloud Gamings gilt es jedoch, die Anforderungen im Hinblick auf die Latenz zu berücksichtigen.

Zusammenfassung der Analyse Die Analyse der einzelnen Gruppen von Anwendungsszenarien im Hinblick auf die zuvor vorgestellten Adaptionsformen ist in Tabelle 5.1 zusammengefasst. Die Auswertung zeigt, dass die vorgestellten Anwendungsfälle in erster Linie von einer Anpassung des Ausführungsortes und der Implementierung profitieren. Nachgelagert profitieren einige der Anwendungsfälle ebenso von einer zeitlichen Anpassung, im Sinne einer Verzögerung der Ausführung.

	Zeit der Ausführung	Ort der Ausführung	Implementierung
Verarbeitung von Sensordaten	+	++	++
Analyse von Audiodaten	+	++	++
Video- und Bildverarbeitung	++	++	++
Mobiles (Cloud) Gaming	-	++	++
Augmented Reality	-	++	++

++ vollständig erfüllt + teilweise erfüllt - nicht erfüllt 0 nicht adressiert

Tabelle 5.1.: Bewertung der Anwendungsfälle in Bezug auf die Adaptionsformen

Diese durch die Adaptionsformen ermöglichten generellen Potenziale sollen im Folgenden nun im Hinblick auf die konkreten Ziele einer Adaption näher untersucht werden.

5.2.3. Ziele der Adaption

Die Ziele einer Kooperation innerhalb des mobilen Cloud Computings wurden bereits grob in Abschnitt 2.3 umrissen und sollen im Folgenden anhand konkreter Adaptionsziele näher charakterisiert werden, um sie im Anschluss im Hinblick auf die Umsetzung mithilfe der im vorhergehenden Abschnitt vorgestellten Adaptionsformen hin abzugleichen.

Die Ziele einer Adaption ergeben sich entsprechend aus den in Abschnitt 2.2.4 aufgezeigten Restriktionen mobiler Geräte, wie beispielsweise der wechselnden und schwankenden Verbindungsqualität zwischen mobilen Geräten und ihrer Infrastruktur, welche zu einer generell eingeschränkten Ressourcenverfügbarkeit gegenüber stationären Geräten führt. Die entsprechenden Adaptionsziele spiegeln die Kompensation dieser eingeschränkten Ressourcenverfügbarkeit mobiler Geräte wider und sollen im Folgenden vorgestellt werden.

Verbesserung der Performance Das im mobilen Cloud Computing prominenteste Optimierungsziel versucht, der eingeschränkten Rechenleistung mobiler Geräte Rechnung zu tragen, und betrifft die Optimierung der Ausführungszeit. Entsprechend häufig wurde versucht, dieses Optimierungsziel in existierenden Forschungsarbeiten zu erreichen [CIM+11, CBC+10, BDR14a, KYK+16, FSB14, SHP+14, YOC08, WLHL13, KNP15]. Wie dieses Adaptionsziel mithilfe der im vorhergehenden Abschnitt vorgestellten Adaptionsformen erreicht werden kann, soll hierzu im Folgenden beschrieben werden:

- Auslagern von Geschäftslogik: Mithilfe der Adaptionsform der Auslagerung kann rechenintensive Geschäftslogik auf ein entsprechend leistungsstarkes Surrogate ausgelagert werden, um sie dort entsprechend schneller auszuführen. Eine Auslagerung zur Verbesserung der Performance bedarf allerdings einer entsprechenden Kommunikation mit der Infrastruktur. Diese wiederum belastet die Netzwerkressourcen eines mobilen Geräts, deren Benutzung gleichzeitig den üblicherweise begrenzten Energievorrat eines mobilen Gerätes reduziert. Die Adaptionsform der Auslagerung steht damit oft in direktem, oft konkurrierendem Zusammenhang mit weiteren beschränkten Ressourcen und gegebenenfalls damit verbundenen weiteren Adaptionszielen.

  Ein weiterer Aspekt den es bei der Auslagerung zur Verbesserung der Performance zu beachten gilt, ist die Berücksichtigung aller im Rahmen eines typischen Auslagerungsprozesses notwendigen Schritte (vergleiche Abschnitt 5.1.1). Dies beinhaltet unter anderem die Suche entsprechender Surrogates, die Initiierung der Auslagerung, den Datentransfer zur Synchronisation des Zustands mit dem mobilen Gerät vor und nach der Ausführung der Geschäftslogik in der Infrastruktur und auch die eigentliche für die Ausführung der Geschäftslogik benötigte reine Ausführungszeit. Eine Verbesserung der Performance durch eine Auslagerung von Geschäftslogik ist daher stets vor dem Hintergrund weiterer und möglicherweise konkurrierender Optimierungsziele zu betrachten.

- Vorziehen von Geschäftslogik: Eine Verbesserung der Performance kann in bestimmten Fällen auch dadurch erreicht werden, dass bestimmte Berechnungsschritte im Voraus ausgeführt werden. Dieses Vorziehen ist oft dann vorteilhaft, solange das mobile Gerät gut mit der Infrastruktur verbunden ist und über erweiterte Ressourcen verfügt. Die Anwendungsfälle, für die diese Form der Adaption eine Verbesserung der Performance ermöglicht, stellen jedoch nicht den Regelfall dar.

- Entfernen von Geschäftslogik: Die Verbesserung der Performance kann ebenso über das Entfernen optionaler Teile der Geschäftslogik erfolgen, wenn diese einen relevanten Teil der gesamten Ausführungszeit einnehmen.

- Austauschen von Geschäftslogik: Wenn der vorgenannte Fall des Entfernens der Geschäftslogik nicht möglich oder im Sinne der Verbesserung der Performance einer mobilen Anwendung nicht vorteilhaft erscheint, bietet es sich alternativ an, die Implementierung auszutauschen. Im Rahmen einer Fidelity-Adaptation ist es somit möglich, einen zeitintensiven Teil der Geschäftslogik durch eine beschleunigte Variante mit reduzierter Qualität zu ersetzen, um somit eine verkürzte Ausführungszeit zu erreichen.

Einsparung von Energie Neben der Verbesserung der Performance stellt die Einsparung von Energie ein weiteres prominentes Ziel innerhalb des mobilen Cloud Computings dar, das ebenso häufig als Optimierungsziel existierender Forschungsarbeiten untersucht wurde, unter anderem in: [CBC+10, SSX+12, ESM14, TLH+16, MYHZ14, OKA+15, LT11, NMSH14, XDL+14]. Das Adaptionsziel der Einsparung von Energie bezieht sich entsprechend auf den beschränkten Energievorrat mobiler Geräte und die daraus resultierende Notwendigkeit eines sparsamen Umgangs mit dieser Ressource, da sich die Energiedichte der Akkus mobiler Geräte pro Jahr nur um fünf bis zehn Prozent erhöht (vergleiche Abschnitt 2.2.4).

Die Einsparung von Energie hat damit primär zum Ziel, die Laufzeit eines mobilen Geräts im Batteriebetrieb so weit wie möglich zu verlängern, und kann entweder ständig als Optimierungsziel festgelegt werden oder lediglich dann, wenn abzusehen ist, dass die verbleibende Betriebszeit des mobilen Geräts nicht bis zur nächsten Lademöglichkeit ausreicht, wie in [Mey16] und [CTSC11] gezeigt wird. Dieses Adaptionsziel kann mithilfe der folgenden Adaptionsformen erreicht werden:

- Auslagern von Geschäftslogik: Wie bereits zuvor im Zusammenhang mit der Optimierung der Performance im Hinblick auf die Adaptionsform der Auslagerung beschrieben kann diese Adaptionsform ebenfalls zur Einsparung von Energie verwendet werden.

  Allerdings steht auch in diesem Fall das Optimierungsziel der Einsparung von Energie oft in Konkurrenz mit weiteren Ressourcenbeschrän-

kungen oder Adaptionszielen. Entsprechend gilt es beispielsweise, die für die Durchführung der Auslagerung benötigte Energie zu berücksichtigen, die beim Senden und Empfangen der auszulagernden Geschäftslogik benötigt wird.

- Verzögern von Geschäftslogik: Eine Einsparung von Energie kann ähnlich wie eine Verbesserung der Performance ebenso dadurch realisiert werden, dass gewisse energieintensive Berechnungsschritte so lange verzögert werden, bis eine entsprechende Lademöglichkeit oder eine Verbindung zu einer externen Energiequelle verfügbar sind oder in Aussicht stehen, um so das Leerlaufen des Akkus zu verhindern, wie es in [Mey16] gezeigt wurde.

- Vorziehen von Geschäftslogik: Eine Einsparung von Energie kann ähnlich wie eine Verbesserung der Performance darüber realisiert werden, dass gewisse energieintensive Berechnungsschritte vorab ausgeführt werden oder dass zu einem späteren Zeitpunkt benötigte Daten vorab eingelagert werden.

  Dieses Vorziehen bietet sich an, solange das Gerät energetisch günstig mit der Infrastruktur verbunden ist und kommunizieren kann, beispielsweise kabelgebunden statt drahtlos, oder solange eine Verbindung zu einer externen Energiequelle besteht und der verbleibende Energievorrat des mobilen Geräts hierdurch nicht beeinträchtigt wird. Da die Nutzung einer mobilen Anwendung in einem zukünftigen Zeitintervall jedoch nur unter Unsicherheit vorhergesagt werden kann, wird diese Adaptionsform nur in bestimmten Fällen als hilfreich angesehen [SKK+14, SHD12, KMYI09, DGP10].

- Entfernen von Geschäftslogik: Ebenso wie bei einer Verbesserung der Performance kann ein Entfernen energieintensiver Berechnungsschritte der Geschäftslogik auch zu einer Reduktion der benötigten Energie genutzt werden.

- Austauschen von Geschäftslogik: Können energieintensive Berechnungsschritte nicht entfernt werden, so bietet es sich alternativ an, diese im Hinblick auf die Verbesserung der Energieeffizienz gegen eine alternative Implementierung auszutauschen. Hierdurch kann die Dienstverfügbarkeit aufrechterhalten und gleichzeitig eine Energieeinsparung umgesetzt werden.

Einsparung von Bandbreite Die Bandbreite eines mobilen Gerätes stellt unter mehreren Gesichtspunkten eine beschränkte und gleichzeitig teure Ressource eines mobilen Geräts dar. Einerseits belastet die Verwendung der Netzwerkschnittstellen, insbesondere der drahtlosen, das energetische Budget eines mobilen Geräts. Andererseits gilt es in Bezug auf die Nutzung der verschiedenen Standards zur Mobilkommunikation zu berücksichtigen, dass deren Nutzung im Bereich der WWANs oft volumenabhängig abgerechnet wird,

hingegen im Bereich der WLANs üblicherweise ein kostengünstiger oder kostenloser Zugang zum Internet bereitgestellt wird. Entsprechend haben sich eine Reihe existierender Arbeiten mit der Einsparung von Bandbreite und der Verschiebung von Datentransfers sowohl zeitlich als auch im Hinblick auf die verwendeten Schnittstellen für die Mobilkommunikation beschäftigt [LBLX13, GJM+12, LLY+10, SA13, MS14, NNH+14, ZTC14, XLC15]. Hinsichtlich der vorgestellten Adaptionsformen ergeben sich hieraus die folgenden Varianten zur Einsparung von Bandbreite:

- Einlagerung von Geschäftslogik: Welche Herausforderungen der Umgang mit einer drahtlosen Verbindung zur Infrastruktur, die einer wechselnden Konnektivität und Bandbreite unterworfen ist, mit sich bringen, wurde bereits im Zusammenhang mit dem mobilen Dilemma (vergleiche Abschnitt 2.2.9) beschrieben. Die dort beschriebene Einlagerung von Geschäftslogik zur Reduktion der Abhängigkeit von der Infrastruktur und einer damit einhergehenden Erhöhung der Dienstverfügbarkeit kann ebenso dazu benutzt werden, das zu übertragende Datenvolumen zwischen einem mobilen Gerät und seiner Infrastruktur zu reduzieren.

  Wie ebenfalls im mobilen Dilemma erwähnt wird, steht diese Einlagerung von Geschäftslogik jedoch oft im direkten Konflikt zu den begrenzten Ressourcen eines mobilen Gerätes, wodurch es sich bei diesem Adaptionszielen ebenfalls oft um ein mit anderen Optimierungszielen konkurrierendes handelt.

- Auslagerung von Geschäftslogik: Im Zusammenhang mit dem mobilen Dilemma wurde ebenso diskutiert, inwiefern die Auslagerung von Geschäftslogik vorteilhaft sein kann, insbesondere mit Blick auf die effiziente Nutzung der beschränkten Ressourcen eines mobilen Gerätes.

  Beispielsweise kann eine mobile Anwendung zum Durchsuchen und Filtern von Nachrichtenportalen so aufgeteilt werden, dass der für den Download und das Filtern verantwortliche Teil der Geschäftslogik in die Infrastruktur verlagert werden kann. Hierdurch müssen nicht länger alle Informationen auf das mobile Gerät transferiert werden, sondern lediglich die im Rahmen der Suchkriterien als relevant identifizierten. Eine Form der Adaption, die dazu dienen kann, dass zu übertragende Datenvolumen zwischen einem mobilen Gerät und seiner Infrastruktur zu reduzieren und damit das Adaptionsziel der Einsparung von Bandbreite zu realisieren.

- Verzögern von Geschäftslogik: Das Verzögern von bandbreitenintensiven Teilen der Geschäftslogik kann ebenso dazu genutzt werden, den Bedarf an Bandbreite auf bestimmten drahtlosen Schnittstellen wie der WWAN-Schnittstelle zu reduzieren. In diesem Zusammenhang lässt sich prinzipiell durch ein sehr langes Verzögern stets eine Optimierung erreichen. Vor dem Hintergrund der durch den Nutzer wahrgenommenen eingeschränkten Dienstqualität gilt es jedoch maximal akzeptable Verzögerungszeiten

zu berücksichtigen, die je nach Anwendungsszenario individuell festzulegen sind.

- Vorziehen von Geschäftslogik: Als Gegenstück zur Verzögerung von Geschäftslogik können ebenso Teile der Geschäftslogik vorgezogen ausgeführt werden, solange eine kostengünstige und zuverlässige Verbindung zur Infrastruktur besteht.

  Exemplarisch für diesen Anwendungsfall ist insbesondere das verteilte Dateisystem *Coda* [SKK+90] zu erwähnen, welches die zuvor genannten Phasen zuverlässiger Konnektivität nutzt, um zukünftig benötigte Daten einzulagern. Es bedarf allerdings einer guten Prognose der zukünftigen Zugriffe auf diese Daten, um dieses Adaptionsziel sinnvoll zu realisieren.

- Entfernen von Geschäftslogik: Das Entfernen von Geschäftslogik kann nicht nur der Optimierung der Ausführungszeit oder der Einsparung der benötigten Energie dienen, sondern, wenn es sich um optionale bandbreitenintensive Anwendungen handelt, auch dem Optimierungsziel der Einsparung von Bandbreite.

- Austauschen von Geschäftslogik: Sollte der vorgenannte Fall des Entfernens der Geschäftslogik nicht möglich oder nicht vorteilhaft im Sinne der Reduktion des Bandbreitenbedarfs einer mobilen Anwendung erscheinen, so kann im Rahmen einer Fidelity-Adaptation ein Austausch eines bandbreitenintensiven Teils der Geschäftslogik hin zu einer Variante erfolgen, deren Bedarf an Bandbreite geringer ist.

Erhöhung der Dienstverfügbarkeit Neben den zuvor vorgestellten drei Varianten zur Entlastung einzelner Ressourcen eines mobilen Gerätes existiert ein weiteres Optimierungsziel, welches sich auf die Verfügbarkeit eines Dienstes bezieht. In diesem Fall ist davon auszugehen, dass auf eine Reduktion der Dienstqualität vom Nutzer einer mobilen Anwendung akzeptiert wird, um im Gegenzug die Verfügbarkeit dieses Dienstes aufrechtzuerhalten. Auch dieses Adaptionsziel wurde bereits in einer Reihe existierender Arbeiten untersucht [NSN+97, SKK+90, GCGBDRD16, SVHVV15]; es stellt jedoch im Hinblick auf das mobile Cloud Computing einen weniger prominenten Anwendungsfall dar. Realisiert werden kann dieser durch die folgenden Adaptionsformen:

- Einlagerung von Geschäftslogik: Wie im Zusammenhang mit der Optimierung der Bandbreite beschrieben kann die Einlagerung von Daten wie im Coda-Dateisystem gezeigt oder von Geschäftslogik nicht nur die Beanspruchung mobiler Datenverbindung verringern, sondern führt gleichzeitig zu einer ständigen Verfügbarkeit der entsprechenden Daten oder der entsprechenden Funktionalität auf dem mobilen Gerät. Die Einlagerung steht jedoch häufig im Konflikt mit den begrenzten Speicher- und Rechenressourcen mobiler Geräte.

- Vorziehen von Geschäftslogik: Eine Verbesserung der Performance kann in bestimmten Fällen ebenso dadurch erreicht werden, dass gewisse Berechnungsschritte ausgeführt werden, solange das mobile Gerät mit der Infrastruktur verbunden ist und auf die dortigen Ressourcen zugreifen kann. Auf diesem Weg kann ebenso die Einlagerung bestimmter vorbereiteter Ergebnisse ermöglicht werden.

- Entfernen von Geschäftslogik: Das Weglassen optionaler Teile der Geschäftslogik kann in bestimmten Fällen dazu genutzt werden, die Verfügbarkeit eines Dienstes zu erhöhen. Insbesondere ressourcenintensive Teile der Geschäftslogik bieten sich hierfür an, um im Gegenzug die Verfügbarkeit eines Dienstes aufrechtzuerhalten oder zu verlängern.

- Austauschen von Geschäftslogik: Analog dem vorgenannten Fall kann statt des Weglassens auch ein Austausch der Geschäftslogik im Sinne einer Fidelity-Adaptation durchgeführt werden, um die Dienstverfügbarkeit aufrechtzuerhalten.

Erhöhung der Dienstqualität Im Hinblick auf die in Abschnitt 2.2.4 vorgestellten Restriktionen mobiler Geräte und die gleichzeitig steigenden Anforderungen der Nutzer dieser Geräte bietet eine Kooperation mobiler Geräte mit der Infrastruktur jedoch nicht nur die Möglichkeit, bestimmte Teile der für ein mobiles Gerät konzipierten Geschäftslogik zu beschleunigen oder den verbleibenden Energievorrat zu schonen. Eine Kooperation kann ebenso dazu genutzt werden, die von stationären Geräten bereitgestellte Dienstqualität zu realisieren.

- Austauschen von Geschäftslogik: Ein Beispiel findet sich in den Dockingstationen von Laptops oder Spielekonsolen, die eine (Hardware)-Erweiterung der Ressourcen mobiler Geräte bereitstellen, mithilfe derer das mobile Gerät in die Lage versetzt wird, eine mit stationären Geräten vergleichbare Dienstqualität bereitzustellen. Ein Teil der Geschäftslogik kann in diesem Fall gegen eine Variante ausgetauscht werden, die von den zusätzlichen Ressourcen gebrauch macht. So kann beispielsweise durch zusätzliche Hardware zur Beschleunigung der Grafikausgabe eine erhöhte Auflösung oder Bildwiederholrate bereitgestellt werden. Im Sinne der hier vorgestellten Adaptionsformen entspricht dies weitestgehend einer Fidelity-Adaptation, die mithilfe des Austauschens der Geschäftslogik realisiert wird.

Zusammenfassung der Adaptionsziele Wie in Tabelle 5.2 zusammenfassend gezeigt wird, ergibt sich für die sieben betrachteten Adaptionsformen eine ungefähr gleich hohe Relevanz und damit Anwendbarkeit im Hinblick auf die identifizierten Optimierungsziele.

Auch wenn es nach dieser Auswertung so scheint, als ob die Hinzunahme optionaler Komponenten in keinem Fall von Vorteil ist, so ist jedoch anzumer-

	Einsparung von Zeit	Einsparung von Energie	Einsparung von Bandbreite	Erhöhung der Dienstverfügbarkeit	Erhöhung der Dienstqualität
Auslagerung	++	++	++	-	+
Einlagerung	-	-	++	++	-
Verzögerung	-	++	+	-	-
Vorziehen	+	+	+	+	+
Hinzufügen	-	-	-	-	+
Weglassen	++	++	+	++	-
Austauschen	++	++	++	++	++

++ vollständig erfüllt + teilweise erfüllt - nicht erfüllt 0 nicht adressiert

Tabelle 5.2.: Bewertung der Adaptionsformen im Hinblick auf die Adaptionsziele

ken, dass diese zwar nicht im Sinne der genannten Einsparungs- und Optimierungsziele vorteilhaft erscheint, jedoch insgesamt, alle diese Optimierungsziele auf eine erhöhte oder verbesserte Dienstqualität mobiler Anwendungen auf mobilen Geräten abzielen. In diesem Zusammenhang ist die optionale Hinzunahme weiterer Teile der Geschäftslogik dann ebenfalls als vorteilhaft im Sinne der Erhöhung der Servicequalität zu betrachten.

Wie die Ziele der Adaption mithilfe der verschiedenen Adaptionsformen softwaretechnisch realisiert werden können, soll im Folgenden untersucht werden. Die Untersuchung orientiert sich dabei an den von Geihs in [GRWK09] als wesentlich definierten Anforderungen an Adaptionsprozesse. Hiernach gilt es insbesondere, die Fähigkeit zur Rekonfiguration und damit die generelle Adaptionsfähigkeit, die dynamische Dienstsuche, den Umgang mit Heterogenität sowie die Integration von Kontextinformationen in die Planung der Adaption zu berücksichtigen. Entsprechend werden diese Aspekte in den nun folgenden Abschnitten dieses Kapitels näher vorgestellt.

5.2.4. Ebenen der Adaption

Die Adaptionsfähigkeit bedarf der generellen Fähigkeit zur Konfiguration, welche mithilfe der vorgestellten Adaptionsformen der parametrischen und kompositionellen Adaption realisiert werden kann. In diesem Zusammenhang lassen sich im mobilen Cloud Computing weitere Varianten unterscheiden, die weitgehend mit der kompositionellen Adaption in Verbindung gebracht werden können.

Hierzu werden einerseits von Fuchß in [Fuc09] die Adaption auf Infrastrukturebene und von Shiraz et al. in [SGKB13] drei weitere Migrationsformen (*migration patterns*) beschrieben. Dieses sind die Adaption auf Systemebene (*VM Migration Based Application Offloading*), die Adaption auf Anwendungsebene (*Entire Application Migration Based Application Offloading*) und die Partitionierung von Anwendungen (*Application Partitioning Based Application Offloading*). Hierbei sind die vorgenannten *Ebenen der Adaption*, die die Schicht innerhalb einer Systemarchitektur beschreiben, oberhalb welcher die

Adaption transparent abläuft, nicht mit der jeweiligen *Granularität der Adaption* gleichzusetzen, die beschreibt, bis zu welchem Grad die Geschäftslogik einer mobilen Anwendung für eine Adaption zerteilt wird. Beispielsweise nutzt die in Abschnitt 4.4.1 vorgestellte Lösung CloneCloud [CIM+11] als Ebene der Adaption die Systemebene und verwendet als Granularität der Adaption einzelne Threads einer mobilen Anwendung - ein Zusammenhang, der im Verlauf dieses Abschnitts noch näher erläutert wird. Entsprechend stellt die dritte von Shiraz et al. in [SGKB13] vorgeschlagene Kategorie keine weitere Ebene der Adaption, sondern die verschiedenen Granularitäten der Adaption dar.

Nach der Vorstellung der einzelnen Ebenen der Adaption sollen diese dahingehend bewertet werden, wie gut sie sich zur Umsetzung der in den vorhergehenden Abschnitten entwickelten Adaptionsformen und Adaptionsziele eignen. Hierzu wird zunächst die Umsetzbarkeit der verschiedenen für die Optimierungsziele benötigten Adaptionsbedarfe auf den verschiedenen Ebenen der Adaption bewertet und eine geeignete Ebene der Adaption ausgewählt. Anschließend werden die Granularitäten der Adaption dahingehend bewertet, wie sie sich zur Umsetzung der Adaptionsziele auf der zuvor ausgewählten Ebene der Adaption eignen. Hieraus wird abschließend eine Entscheidung abgeleitet, auf welcher Basis eine kontextadaptive Anwendungsarchitektur vor dem Hintergrund der zu realisierenden Adaptionsformen und Optimierungsziele sinnvoll zu realisieren ist.

Adaption auf Infrastrukturebene Neben den für das mobile Cloud Computing typischen drei Ebenen der Adaption existiert ebenso die Möglichkeit zur Adaption auf Infrastrukturebene, die der Vollständigkeit halber an dieser Stelle Erwähnung findet. Das in Zusammenhang mit dem Mobilfunk Standard UMTS entwickelte *Virtual Home Environment* (VHE) [ETS99] bietet hierzu laut Fuchß [Fuc09] einem Mobilfunkteilnehmer die Möglichkeit, eine einheitliche und vertraute Infrastruktur zur Nutzung bereitzustellen.

Der Ansatz des VHE versucht hierzu, eine umfassende Lösung zu bieten, bei der einerseits das Einbuchen mobiler Geräte in Fremdnetze (Roaming), aber auch die Vielfalt mobiler Geräte berücksichtigt wird, um einem Nutzer personalisierte Dienste, beispielsweise Adressbücher, Favoritenlisten, Profile und abonnierte Dienste wie Voice- und Mailboxen, sicher sowie netz- und geräteübergreifend bereitzustellen. Hierzu nutzt VHE die Möglichkeit, mithilfe der personenbezogenen SIM-Karte eines Nutzers individuelle Dienste unabhängig vom Endgerät bereitstellen zu können. VHE definiert hierzu eine Referenzarchitektur, die darauf abzielt, eine standardisierte Ausführungsumgebung bereitzustellen, die laut [Fuc09] wie folgt aufgebaut ist:

- *DAT*: ein definierter Speicher für die dienstspezifischen Daten, zum Beispiel Kontonummern, E-Mail-Adressen, Favoritenlisten.
- *PRG*: ein definierter Programmspeicher, in dem die diensterbringenden Applikationen verwaltet werden.

- *EXE*: eine definierte Ausführungsumgebung bzw. Laufzeitumgebung für die Dienste und Applikationen, vergleichbar mit einer Java-Sandbox. Die EXE verfügt ihrerseits über zwei Schnittstellen:
  - Die *Service-API*: eine offene Schnittstelle, die zur Anbindung der Dienste dient. Über diese Schnittstelle werden die Programme aus dem Programmspeicher in die EXE-Umgebung geladen, und es wird auf die Daten aus dem Datenspeicher zugegriffen.
  - Die *Network-API*: über diese Schnittstelle können Programme, die in der EXE zur Ausführung kommen, die Kommunikation steuern. Hierzu gehört die Möglichkeit zur Rufweiterleitung, genauso wie die Möglichkeit zur Annahme eines Anrufs.
- *COC*: eine definierte Einheit zur Kommunikationssteuerung. Dieses Modul liegt unterhalb der Network-API und realisiert die an der Network-API angebotene Funktionalität.

Diese Referenzarchitektur ist damit weder dienst- noch gerätespezifisch und bildet die grundlegende Architektur für alle an der Erbringung eines Dienstes beteiligten Komponenten. Ebenso berücksichtigt dieser Ansatz die Ausführung personalisierter Dienste innerhalb der Infrastruktur (CAMEL) sowie auch auf dem mobilen Gerät (MExE) oder auf der SIM-Karte selbst (SAT), wobei letzteres zusätzlich dem Aspekt des Schutzes persönlicher Daten in einer geteilten Infrastruktur Rechnung trägt.

Diese Lösung besitzt damit im Gegensatz zu vielen anderen Konzepten eine vergleichsweise hohe Marktreife, was unter anderem daran zu erkennen ist, dass offene Problemstellungen wie beispielsweise die Heterogenität der Infrastruktur und eine nutzungsbezogenen Abrechnung durch den Mobilfunkanbieter adressiert werden. Allerdings hat dieser seit über 15 Jahren bestehende Standard bis heute zu keiner praktischen Umsetzung geführt [Fuc09]. Dies kann darauf zurückzuführen sein, dass zum Zeitpunkt der Standardisierung nur wenige Anwendungen verfügbar waren, die von diesem Standard profitiert hätten, und dass das Konzept inzwischen von cloud-basierten Modellen abgelöst wurde, die nicht nur mobile, sondern auch stationäre Geräte unterstützen.

Entsprechend soll die Adaption auf Infrastrukturebene im Folgenden nicht weiter betrachtet werden, da es einerseits tiefe Eingriffe in die Infrastruktur der Mobilfunkanbieter erfordern würde und andererseits als technologisch überholt bewertet werden kann.

Adaption auf Systemebene Bei der Adaption auf Systemebene erfolgt auf der Ebene des Betriebssystems des mobilen Gerätes eine Anpassung seines Systemabbildes oder seines momentanen Ausführungszustands. Die Adaption besteht auf dieser Ebene darin, das Systemabbild und den momentanen Ausführungszustands des mobilen Geräts in eine leistungsstärkere Infrastruktur zu migrieren, wie es in Abschnitt 4.4.1 beschrieben wurde.

Entsprechende Lösungen nutzen hierzu häufig eine Virtualisierung, um eine Verteilungstransparenz oberhalb der unterschiedlichen Systemarchitekturen herzustellen [SGKB13]. Diese ermöglicht es, von der darunterliegenden Hardware zu abstrahieren und hierdurch den Ausführungszustand des mobilen Betriebssystems und der laufenden mobilen Anwendungen zwischen dem mobilen Gerät und seiner Infrastruktur zu migrieren. Frühe Lösungen im Bereich der Adaption auf Systemebene, die eine Migration durch die Übertragung eines Systemabbildes durchführen, weisen hierdurch einen entsprechend hohen Overhead in Bezug auf die übertragene Datenmenge auf. Zusätzlich erfordern sie zur erfolgreichen Erfassung und Übertragung des Ausführungszustands, die Ausführung auf dem mobilen Gerät zu pausieren oder zu verlangsamen, womit oft eine temporäre Einschränkung der Nutzbarkeit einhergeht [CFH+05, LQLL10].

Fortgeschrittene Lösungen, wie sie in [GJM+12, ZXC+10, CIM+11] vorgestellt werden, begegnen dieser Einschränkung dadurch, dass sie versuchen, nach einer initialen Synchronisation nur die Veränderungen gegenüber vorherigen Synchronisierungspunkten zu übertragen. Hierzu wird von diesen Lösungen üblicherweise der Heap und der Stack überwacht, um nach der Erkennung eines Adaptionsbedarfes innerhalb möglichst kurzer Zeit den Synchronisationsbedarf zu erkennen und die Synchronisation starten zu können. Auch hier ist es allerdings erforderlich, während der Übertragung des Ausführungszustands die Ausführung zu pausieren.

Diese Trennung des zu synchronisierenden Ausführungszustands in ein Basisabbild und ein Differenzabbild (diff) ermöglicht es zusätzlich zu den Vorteilen im Hinblick auf ein deutlich geringeres zu übertragendes Datenvolumen und eine beschleunigte Synchronisation auch den Speicherbedarf innerhalb der Infrastruktur zu reduzieren, wenn die Differenzabbilder mehrstufig gebildet werden. In diesem Zusammenhang kann ein geräteabhängiges Basis-Abbild für alle Nutzer dieses Gerätes um ein nutzerabhängiges Differenzabbild ergänzt werden, welches die Anwendungen und Daten des jeweiligen Nutzers enthält. Mit einem weiteren Differenzabbild kann dieses nutzerabhängige Differenzabbild hin zum aktuellen Ausführungszustand des mobilen Gerätes ergänzt werden.

Trotz dieser Optimierungen weist diese Gruppe von Lösungen durch die verbleibenden Synchronisierungsaufwände typischerweise eine so hohe Latenz auf, dass aus Nutzersicht eine Einschränkung der Dienstqualität entstehen kann [AGK+15]. Eine zusätzliche Möglichkeit, die sinnvollen Einsatzmöglichkeiten dieses Ansatzes zu verbessern, besteht darin, lediglich die Ein- und Ausgabe des mobilen Gerätes umzuleiten und dieses als Terminal oder den Client zu verwenden, wie es in [LLS11] gezeigt wurde. Der Nachteil dieser Variante der Adaption auf Systemebene besteht jedoch in einer zusätzlich erhöhten Abhängigkeit von der Infrastruktur.

Insgesamt ist die praktische Nutzbarkeit der Adaption auf Systemebene zusätzlich dadurch eingeschränkt, dass durch die beschriebenen Synchronisierungsaufwände und das Erfordernis der (ressourcenintensiven) Ausführung

des Systemabbildes in der Infrastruktur jeweils nur ein Surrogate zur Zeit zur Erweiterung der Ressourcen eines mobilen Gerätes herangezogen werden kann. Hierdurch muss die Möglichkeit zur spontanen Interaktion als eingeschränkt bewertet werden. In diesem Zusammenhang gilt es zusätzlich, zwischen den häufig unterschiedlichen CPU-Architekturen (*ARM*[2] auf dem mobilen Gerät, *x86* in der Infrastruktur) zu übersetzen, dies durch einen entsprechenden Hardware Abstraction Layer zu unterstützen oder, wie bei CloneCloud [CIM+11] realisiert, eine entsprechende Modifikation des mobilen Betriebssystems vorzunehmen, wodurch die Ressourcenbelastung zusätzlich erhöht und die Möglichkeit zur Ad-hoc-Interaktion weiter eingeschränkt wird.

Im Gegenzug bietet diese Ebene der Adaption allerdings den Vorteil, mobile Anwendungen ohne Anpassung und ohne Vorhandensein des Quellcodes in die Infrastruktur zu übertragen und ausführen zu können. Diese unmodifizierte Ausführung mobiler Anwendungen schränkt wiederum gleichzeitig die Möglichkeit zur nebenläufigen Ausführung stark ein, sodass eine mobile Anwendung nur bedingt von den Möglichkeiten der nebenläufigen Ausführung in der Infrastruktur profitieren kann [SGKB13]. So kann beispielsweise bei Einsatz der Lösung CloneCloud [CIM+11] nur ein Thread in der Infrastruktur ausgeführt werden, während die Ausführung auf dem mobilen Gerät pausiert, um eine Rückführung der Ergebnisse auf das mobile Gerät zu erlauben.

Adaption auf Anwendungsebene Im Gegensatz zur Auslagerung eines vollständigen Abbilds des mobilen Geräts hat sich eine Reihe weiterer existierender Ansätze wie [MDP+00, MLW11, HSC12] mit einer Migration auf der Ebene der einzelnen Anwendungen eines mobilen Geräts beschäftigt. Frühe Arbeiten, die eine Auslagerung auf Ebene der mobilen Anwendungen vollziehen, nutzen als Ebene der Adaption oft den Betriebssystemprozess als Granularität der Adaption. Diese wird von Fugetta in [FPV98] weiter in die starke und die schwache Migration differenziert. Bei der schwachen Migration werden lediglich Programmcode und Daten transferiert, wohingegen bei der starken Migration der Ausführungszustand einer mobilen Anwendung mit übertragen wird.

Dem Prinzip geschuldet, dass aktuell eingesetzte Betriebssysteme, insbesondere die für mobile Geräte, zur Ausführung von mobilen Anwendungen eine virtuelle Maschine auf Anwendungsebene nutzen (application-layer virtual machine), bedienen sich neuere Ansätze wie [GC04, BDR14a] dieser konkreten Ebene der Adaption. Ein wesentlicher Vorteil der Migration auf dieser Ebene entsteht durch die stärkere Abstraktion von der auf dem mobilen Gerät und in der Infrastruktur verwendeten Hardware.

Dies erfolgt durch die Übersetzung des Quellcodes einer mobilen Anwendung in einen sogenannten Bytecode, ein zur Ausführung auf einer virtuellen Maschine bestimmter und leicht portierbarer Code, der ohne Modifikation auf jedem Gerät ausführbar ist, das eine entsprechende virtuelle Maschine bereitstellt. Ein prominentes Beispiel ist die Nutzung der *Dalvik Virtual Machine*

[2] http://infocenter.arm.com

unter dem Betriebssystem Android, welche mithilfe der *Java Programming Language* entwickelte Software ausführen kann. Allerdings gilt es im Gegensatz zur Adaption auf Systemebene, sämtliche Abhängigkeiten, zum Beispiel gemeinsam genutzte Bibliotheken, zu berücksichtigen, wenn eine Migration einer mobilen Anwendung in die Infrastruktur erfolgt [HSC12].

Der Vorteil einer Migration auf Ebene der mobilen Anwendung besteht damit im Gegensatz zur Adaption auf Systemebene im Wesentlichen darin, dass eine deutlich feingranularere Adaption möglich wird, unterschiedliche mobile Anwendungen auf unterschiedliche Surrogates in der Infrastruktur migriert werden können und die für die Synchronisation des Zustands zu übertragende Datenmenge im Regelfall deutlich geringer ist. Hierdurch wiederum erhöht sich die Anzahl der Situationen, in denen eine solche Adaption auch für mobile Geräte mit schlechter Konnektivität zur Infrastruktur sinnvoll ermöglicht wird. Da üblicherweise während der Migration keine Interaktion mit der Anwendung möglich ist, profitiert zusätzlich auch die Dienstqualität einer mobilen Anwendung, da eine nahtlose und verzögerungsarme Migration begünstigt wird [AGK+15]. Werden diese mobilen Anwendungen zusätzlich modifiziert und auf eine nebenläufige Ausführung in der Infrastruktur explizit vorbereitet, lässt sich die Effizienz dieser Kooperation weiter steigern, wie es unter anderem in [KAH+12, JCY14] gezeigt wurde.

5.2.5. Granularität der Adaption

Neben den zuvor vorgestellten drei Ebenen der Adaption existiert ein weiteres Kriterium, welches zur Unterscheidung der verschiedenen Realisierungsmöglichkeiten einer Adaption im mobilen Cloud Computing herangezogen werden kann – die Granularität der Adaption. Die Granularität der Adaption beschreibt, auf welcher Ebene die Geschäftslogik einer mobilen Anwendung im Hinblick auf eine Adaption zerteilt wird, eine Aufgabe, die üblicherweise von der Komponente des Partitioners (vergleiche Abschnitt 5.1) übernommen wird.

Der Grund für die Partitionierung einer mobilen Anwendung besteht darin, die im Hinblick auf die Adaption zu erreichenden Ziele besser umsetzen zu können, als dies mit einer alleinigen Adaption auf System- oder Anwendungsebene möglich wäre. Existiert beispielsweise innerhalb einer mobilen Anwendung zur Bildverarbeitung lediglich ein bestimmter Teil von Geschäftslogik, der von einer Migration in die Infrastruktur profitiert, so bietet es sich an, nur diesen Teil der Geschäftslogik und den hierzu benötigten Teil des Ausführungszustands in die Infrastruktur zur Ausführung zu übertragen. Muss hingegen stets die gesamte mobile Anwendung übertragen werden, so steigt unter anderem der hierfür benötigte Bedarf an Übertragungskapazität auf den Schnittstellen des mobilen Geräts, und es erhöht sich gleichzeitig die für die Migration benötigte Zeit – eine Situation, durch die sich die Möglichkeiten zur spontanen Interaktion mit der Infrastruktur deutlich einschränken. Die Partitionierung mobiler Anwendungen ist im Hinblick auf das Ziel des mobilen Cloud Computings, die Ressourcenbeschränkung mobiler Geräte zu umgehen, stets als vor-

teilhaft anzusehen, sofern die für die Partitionierung und Entscheidung zur Auslagerung benötigten Ressourcen wie die Rechenleistung geringer sind als die zu erwartende Entlastung der Ressourcen, die durch die Kooperation mit der Infrastruktur entsteht [LAS+15].

Die Partitionierung von Anwendungen, wie sie im Zusammenhang mit der Basisarchitektur für mobiles Cloud Computing und im Rahmen der Charakterisierung eines typischen Auslagerungsprozesses bereits beschrieben wurde, kann mithilfe einer Reihe verschiedener Granularitäten ermöglicht werden, die im Folgenden vorgestellt und eingeordnet werden sollen. Grundlegend kann hierbei zwischen zwei Arten von Granularitäten unterschieden werden – zwischen denen, die sich anhand der Struktur einer Anwendung zum Zeitpunkt des Entwurfs definieren, und denen, die sich anhand der Struktur einer Anwendung zur Laufzeit definieren.

Die zum Entwurfszeitpunkt einer mobilen Anwendung herangezogenen strukturierenden Elemente sind beispielsweise die *Methoden* [CBC+10, YKC+13, SSX+12], die *Klassen* [GNM+03, OYH07] oder die teils explizit durch den Entwickler auf den Zweck der Adaption zugeschnittenen *Komponenten* [RSM+11, MGL+11, SSSL12, YCY+13, OKA+15, VSDTD12a, VSDTD13, SVHVV15] oder *Komponenten-Bundles* [GRJ+09, RRA08, BGSH07] einer mobilen Anwendung.

Die zur Laufzeit einer mobilen Anwendung herangezogenen strukturierenden Elemente orientieren sich hingegen am Kontrollfluss oder Zustand einer mobilen Anwendung und sind entsprechend die Threads [NTG+09, CIM+11], die Objekte [NSA14] oder die einzelnen Tasks [SHP+14, JCY14, MYHZ14, NMSH14, YYJZ13, WL04] einer solchen, wobei letztere ebenfalls oft vom Entwickler speziell auf den Zweck der Adaption hin ausgerichtet werden. Ein Task stellt hierbei eine, aus Sicht des Anwenders zusammenhängende, fachliche Aufgabe innerhalb einer Anwendung dar, wodurch diese auch als voneinander unabhängig betrachtet werden.

5.2.6. Bewertung in Bezug auf Adaptionsformen und Ziele

Nachdem die verschiedenen Ebenen und Granularitäten der Adaption vorgestellt wurden, soll im Folgenden ihre Eignung im Hinblick auf die Realisierung der angestrebten Adaptionsformen und Ziele und damit einer kontextadaptiven Anwendungsarchitektur für das mobile Cloud Computing bewertet werden.

Ebene der Adaption Mit Blick auf die vorgestellten Ebenen der Adaption wurde bereits festgehalten, dass die Adaption auf Infrastrukturebene im Rahmen der vorliegenden Arbeit nicht weiter betrachtet werden soll (vergleiche Abschnitt 5.2.4). Alternativ bietet sich hierzu die Adaption auf Systemebene an, deren wesentlicher Vorteil darin zu sehen ist, dass mobile Anwendungen oft ohne Anpassung adaptiert und in die Infrastruktur migriert werden können. Diese vollständige Verteilungstransparenz schränkt jedoch gleichzeitig die in

Abschnitt 4.2.3 ermittelte Anforderung der Fähigkeit einer mobilen Anwendung zur Wahrnehmung des aktuellen Ausführungskontexts und der Anpassung an diesen ein. Dieser prinzipielle Vorteil ist somit vor dem Hintergrund der Relevanz der Fähigkeit zur Kontextadaption differenziert zu betrachten. Ebenso ist die Anforderung des geringen Overheads einer entsprechenden Lösung mit der durch die Nutzung virtueller Maschinen einhergehende ressourcenintensive Speicherung, Übertragung und Ausführung dieser nicht zu vereinen.

Zur Realisierung der angestrebten Adaptionsziele bietet sich entsprechend die Adaption auf Anwendungsebene an, bei welcher keine prinzipiellen Nachteile gegenüber der Adaption auf Systemebene ermittelt werden konnten. Zusätzlich bietet diese Ebene der Adaption den Vorteil, dass der zu erfassende und synchronisierende Ausführungszustand kleiner ist und sich somit die Adaptionsziele der Verbesserung der Performance, der Einsparung von Energie und der Einsparung von Bandbreite deutlich einfacher realisieren lassen und aus diesem Grund die Situationen, in denen eine spontane Interaktion mit der Infrastruktur vorteilhaft ist, weitaus häufiger zu erwarten sind.

Granularität der Adaption In Bezug auf die Granularität der Adaption kann mit Blick auf die existierenden Lösungen festgehalten werden, dass bei den zur Entwurfszeit vorhandenen Strukturierungsmerkmalen am häufigsten Methoden und Komponenten Anwendung finden. Bei den zur Laufzeit vorhandenen Strukturierungsmerkmalen stellen einzelne Tasks innerhalb einer mobilen Anwendung die am häufigsten verwendete Granularität dar.

Komponenten bringen den Vorteil einer losen Kopplung zwischen den einzelnen Teilen der Geschäftslogik einer mobilen Anwendung, die insbesondere bei einer wechselnden Konnektivität zwischen den einzelnen Systemen einer mobilen Cloud von Vorteil ist [WHB13]. Zusätzlich sorgt diese, im Vergleich zu Methoden höhere, Granularität dafür, dass die im Rahmen der Adaption zu berücksichtigenden Abhängigkeiten deutlich einfacher und damit schneller zu ermitteln sind, was ebenso als vorteilhaft im Hinblick auf eine spontane Interaktion gesehen wird, da gleichzeitig die Synchronisierungsaufwände deutlich geringer sind [GRA12]. Damit soll festgehalten werden, dass sich die Granularität der Komponente am ehesten zur Realisierung der angestrebten Adaptionsziele anbietet.

Um jedoch gleichzeitig den in Abschnitt 5.2.1 entwickelten Adaptionsformen der Anpassung innerhalb der Dimensionen Zeit, Ort und Implementierung Rechnung zu tragen, bietet es sich an, die zur Entwurfszeit durch den Entwickler zu definierende komponentenbasierte Granularität mit der zur Laufzeit vorhandenen Granularität einzelner Tasks einer mobilen Anwendung zu kombinieren, um, wie noch gezeigt wird, gleichzeitig die geforderten Aspekte der Kontextadaptivität mobiler Anwendungen angemessen zu berücksichtigen.

Wie im Zusammenhang mit der Basisarchitektur für mobiles Cloud Computing in Abschnitt 5.1 gezeigt wurde, ist es hierfür erforderlich, eine mobile

Anwendung entsprechend zu partitionieren. Wie diese Partitionierung sinnvoll erfolgt und welche weiteren Teilprobleme sich bei der Realisierung einer kontextadaptiven Anwendungsarchitektur ergeben, soll im nun folgenden Abschnitt beschrieben werden.

5.3. Identifikation von Teilproblemen und Lösungsansätzen

Die wesentlichen Herausforderungen innerhalb des mobilen Cloud Computings wurden in Kapitel 2 initial erwähnt und im Rahmen der Anforderungsanalyse in Abschnitt 4.2 weiter konkretisiert. Diese Anforderungen werden im nun folgenden Abschnitt im Hinblick auf die sich aus den Anforderungen ergebenden konkreten Teilprobleme hin untersucht.

5.3.1. Heterogenität mobiler Clouds und fehlende Standards

Durch eine Vielzahl verschiedener Kategorien mobiler Geräte besteht eine hohe Heterogenität zwischen den vorhandenen Geräteklassen. Bei der Entwicklung von Anwendungen für mobile Geräte gilt es entsprechend, diese Vielfalt mobiler Geräte zu berücksichtigen.

Mit Bezug auf das mobile Cloud Computing erstreckt sich diese Heterogenität ebenso auf die hier genutzte Infrastruktur. Die Standardisierung der Integration der verschiedenen existierenden Geräte und Kommunikationsstandards im mobilen Cloud Computing ist bis auf erste frühe Standardisierungsansätze (vergleiche [ETS14, ETS15]) jedoch noch nicht zu erkennen [SAGB14]. Aktuell existieren weder Plattform-Dienstleister noch Infrastruktur-Dienstleister, die diese Art von Anwendungen oder entsprechende Surrogate-Infrastrukturen bereitstellen. Ausgenommen hiervon existieren lediglich etablierte Cloud Service Provider wie Amazon [Ama16], die ihre Dienstleistungen jedoch zentralisiert vorhalten und keine Infrastruktur an den Kanten oder Endpunkten der Netzwerke bereitstellen. Weiterhin existieren aktuell keine Geschäfts- oder Abrechnungsmodelle, die für die entsprechenden Anbieter wie Mobilfunkdienstleister oder Cloud Service Provider einen Anreiz bereitstellen würden, solche Infrastrukturen zu etablieren. Dies wiederum liegt auch darin begründet, dass aktuell noch kein für das notwendige Investment in die Infrastruktur rentabler Anwendungsfall identifiziert wurde, der solch einen Aufbau rechtfertigen würde.

Implikationen für diese Arbeit Ein etabliertes Konzept zum Umgang mit dieser Heterogenität stellt die Abstraktion von der spezifischen Hardware mithilfe einer zusätzlichen Virtualisierungsschicht dar [CDK12]. Entsprechend soll der Heterogenität im Wesentlichen dadurch Rechnung getragen werden, dass für die Entwicklung einer kontextadaptiven Anwendungsarchitektur eine Virtualisierung auf Anwendungsebene (vergleiche Abschnitt 5.2.4) genutzt wird, um der entsprechenden Anforderung P1 (vergleiche Abschnitt 4.2.8) in Bezug auf die Berücksichtigung der Heterogenität Rechnung zu tragen.

5.3.2. Suche und Integration neuer Ressourcen

Eine weitere wesentliche Anforderung an die Konstruktion einer kontextadaptiven Anwendungsarchitektur stellt die Fähigkeit zur Suche und Integration neuer Ressourcen dar (vergleiche Anforderung S2 aus Abschnitt 4.2.8).

Im Hinblick auf den schnell wechselnden Kontext mobiler Geräte gilt es hierbei, mobilen Anwendungen zur Laufzeit die spontane Interaktion mit der Infrastruktur zu ermöglichen. Diese Suche und Integration neuer Ressourcen ist ein bekanntes Problem innerhalb des Forschungsgebietes der verteilten Systeme. In diesem Zusammenhang hat sich das Architekturmuster der dienstorientierten Architektur (service oriented architecture) [Erl05] etabliert, in welchem die Ressourcen, sowohl Hardware als auch Software, über wohldefinierte Schnittstellen in Form von Diensten bereitgestellt werden. Ein bestimmter Dienst kann somit auch durch unterschiedliche Ressourcen bereitgestellt werden, ist aber jeweils über die gleiche Schnittstelle ansprechbar.

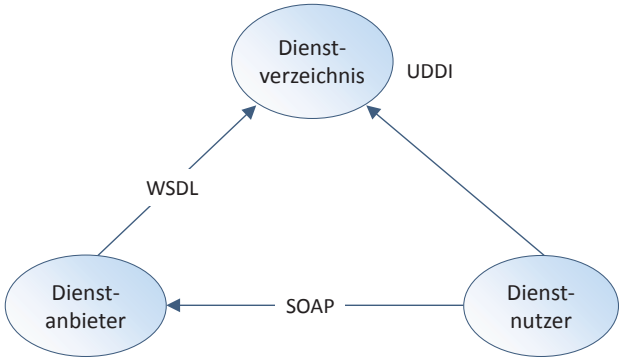

Abbildung 5.4.: Elemente einer dienstorientierten Architektur

Wie in Abbildung 5.4 gezeigt wird, existieren in diesem Zusammenhang Diensterbringer, die diese Dienste bereitstellen und sich hierzu bei einem entsprechenden Verzeichnisdienst registrieren. Dienstkonsumenten können diesen Verzeichnisdienst anschließend befragen, um eine Referenz auf die entsprechende Dienstinstanz zu erhalten und diese zu nutzen.

Anforderungen an die Dienstsuche Die wesentlichen Aufgaben sind nach [BR00] wie folgt charakterisiert:

- Die Erkennung verfügbarer Dienste
- Die Auswahl des richtigen Dienstes
- Die Nutzung des ausgewählten Dienstes

Die in diesem Zusammenhang entwickelten Verfahren wurden jedoch im Hinblick auf die Verwendung in infrastrukturbasierten Netzwerken entwickelt und lassen sich nur bedingt in die im mobilen Cloud Computing anzutreffenden mobilen Ad-hoc-Netzwerke integrieren [Fur12]. Die wesentlichen Herausforderungen der Dienstsuche in mobilen Umgebungen bestehen damit dar-

in, zwischen ressourcenbeschränkten Geräten, die über drahtlose Verbindungen zur Mobilkommunikation kommunizieren, zu vermitteln. Das erste konkrete Teilproblem in diesem Zusammenhang ist die beschränkte Bandbreite [MBB09] und wechselnde Konnektivität [Fur12]. Das zweite konkrete Teilproblem betrifft die zur Nutzung der Schnittstellen zur Mobilkommunikation nötige Energie im Zusammenhang mit dem begrenzten Energievorrat mobiler Geräte [ASG+14]. In diesem Zusammenhang ergeben sich die folgenden Anforderungen an die Dienstsuche in mobilen Umgebungen:

- Fähigkeit zur Kommunikation: Die Suche und Integration neuer Ressourcen erfordert es zudem, dass die beteiligten Geräte miteinander kommunizieren können. Diese Kommunikation kann dabei einerseits direkt oder unter Zuhilfenahme einer Infrastruktur geschehen. Im Fall der Adhoc-Kommunikation, wie sie beispielsweise im Bluetooth-Standard stattfindet, können die beteiligten Geräte üblicherweise direkt miteinander Nachrichten austauschen. Im Hinblick auf eine schnelle und dynamische Nutzung von Ressourcen wie bei dem mobilen Cloud Computing gilt es, sowohl die infrastrukturbasierte als auch die direkte Kommunikation zur Integration neuer Ressourcen zu unterstützen.

  Sind mobile Geräte jedoch über eine Infrastruktur wie das Internet mit anderen Geräten verbunden, können diese Geräte oft nicht direkt miteinander kommunizieren, da Verfahren wie die *Network Address Translation* oder *Firewalls* dazu eingesetzt werden, die (privaten) Subnetze, in denen sich die mobilen Geräte befinden, vom direkten Zugriff aus dem Internet zu trennen. Es ist davon auszugehen, dass diese Restriktionen nicht ohne Weiteres zu umgehen sind. Entsprechend sind Mechanismen zur Etablierung einer direkten Kommunikation vorzusehen, die hierzu die verschiedenen vorhandenen Schnittstellen zur Mobilkommunikation nutzen, um eine möglichst hohe Konnektivität des mobilen Geräts sicherzustellen. Hierfür scheint es erforderlich, eine logische Adressierung oberhalb der physischen Struktur der einzelnen Schnittstellen zur Mobilkommunikation zu etablieren, ein sogenanntes Overlay-Netz (vergleiche Anforderung A1).

- Mobilität und Robustheit: Ist durch die vorgenannte Anforderung die grundlegende Kommunikationsfähigkeit zwischen den beteiligten Geräten etabliert, so gilt es, zwischen den beteiligten Ressourcen zu erkennen, welche von ihnen einen benötigten Dienst bereitstellt. Eine Funktionalität, die es einer mobilen Anwendung erlaubt, nach verfügbaren Diensten zu suchen, sollte dabei möglichst so gestaltet sein, dass sie auch in fremden Netzwerken automatisch die benötigten Dienste zur Laufzeit der Anwendung sucht und für die Benutzung an die mobile Anwendung bindet.

  Ist ein Dienst an einer Anwendung gebunden, sollte hierdurch die Endgerätemobilität (vergleiche Abschnitt 2.2.2) nicht zusätzlich eingeschränkt werden. Entsprechend sollte sichergestellt sein, dass die Nutzung ei-

nes Dienstes über die Grenzen eines Netzwerks hinweg möglich ist und hierfür einerseits die (für den erwarteten Zeitraum der Dienstnutzung) nächstgelegene und bestmöglich erreichbare Instanz eines Dienstes ausgewählt wird [KH07]. Zusätzlich gilt es, sowohl in Bezug auf die Suche als auch die Nutzung eines Dienstes eine Fehlertransparenz (vergleiche Anforderung A1) sicherzustellen und dafür Sorge zu tragen, dass der Ausfall von hierfür benötigten Knoten eines verteilten Systems kompensiert wird, ohne dass es zu einem Ausfall des Dienstes oder der Suche kommt.

- Effizienz und Skalierbarkeit: Da sich die Topologie und die Größe eines mobilen Ad-hoc-Netzwerkes unter Umständen häufig und stark verändert, ist es erforderlich, dass die Kommunikation so effizient wie möglich stattfindet, um einerseits neue Dienste innerhalb eines Netzwerkes möglichst schnell zu erkennen und andererseits die energetisch teuren Schnittstellen zur Mobilkommunikation nur so häufig, wie es hierfür nötig ist, zu aktivieren [ASG+14]. Im Hinblick auf die Skalierbarkeit gilt es hierzu ebenso, dieses Verhalten sowohl in kleinen Netzwerken mit wenigen Knoten als auch in größeren mobilen Ad-hoc-Netzwerken zu unterstützen.

Architekturen und Protokolle für die Dienstsuche In Bezug auf die verschiedenen Architekturen der Dienstsuche lassen sich zunächst die *aktive Suche* und die *passive Suche* voneinander unterscheiden. Im ersten Fall wird die Anfrage nach einem bestimmten Dienst an alle erreichbaren Knoten versendet. Im Fall der passiven Suche wird nur lokal nach einem Dienst gesucht und hierzu die bereits empfangenen Nachrichten über vorhandene Dienste im Netzwerk verwendet [MBB09]. In diesem Zusammenhang können Architekturen für die Dienstsuche weiter dahingehend unterschieden werden, ob ihnen eine verzeichnisbasierte oder eine verzeichnislose Architektur zugrunde liegt [VP08].

In einer verzeichnisbasierten Architektur existiert ein oder eine Reihe von Knoten, die die Beschreibungen aller im Netzwerk vorhandenen Dienste sammeln (sogenannte *directory agents*) und diese den anfragenden Knoten zur Verfügung stellen. Beispiele für verzeichnisbasierte Protokolle sind das *Service Location Protocol* (SLP) [Gut99], *Jini* [Wal99] und Salutation [Pas99].

Exemplarisch soll an dieser Stelle das Service Location Protocol kurz vorgestellt werden. Wie in Abbildung 5.5 veranschaulicht werden hier Dienste durch einen den Dienst bereitstellenden *Service Agent* bekannt gemacht. Die Bekanntmachung erfolgt dabei gegenüber dem sogenannten *Directory Agent*, dessen Aufgabe es ist, ein Verzeichnis aller im Netzwerk erreichbaren Dienste vorzuhalten. Diese Dienste können dann durch Knoten der *User Agents* gefunden und genutzt werden, indem der Directory Agent auf die Suchanfrage des User Agents mit einer entsprechenden Referenz auf den Dienst antwortet.

Bei Betrachtung dieser Architektur wird deutlich, dass von einer weitgehend gleichen Konnektivität der Knoten untereinander ausgegangen wird. Insbe-

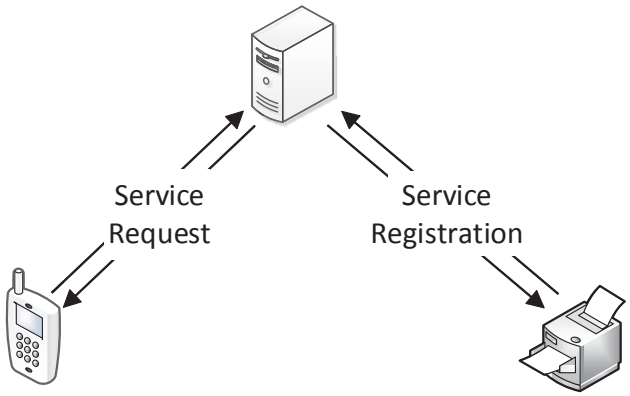

Abbildung 5.5.: Ablauf der Dienstsuche im Service Location Protocol

sondere die Erreichbarkeit des Verzeichnisdienstes entscheidet darüber, ob eine verzeichnisbasierte Architektur nutzbar ist.

Um dieser Einschränkung zu begegnen, existieren verzeichnislose Architekturen, in welchen kein Verzeichnisdienst existiert und Dienstanbieter und Dienstkonsumenten direkt miteinander kommunizieren. Beispiele hierfür sind das *Universal Plug and Play* (UPnP) [JW03] und das SLP im Modus der „*active discovery*“, in welchem User Agent und Service Agent ihre Anfragen bzw. Registrierungen mithilfe des IP-Protokolls in eine Multicast-Gruppe senden. Solch eine verzeichnislose Architektur erfordert es allerdings, dass alle Knoten einerseits laufend empfangsbereit sein müssen, um die Bekanntmachung neuer Dienste zu erkennen und dass sie andererseits auf Dienstanfragen anderer Knoten reagieren müssen. Im Hinblick auf die beschränkten Energie- und Bandbreitenressourcen mobiler Geräte sind diese aktiven Mechanismen zur Dienstsuche somit nur bedingt oder nur für relativ kleine Netzwerke geeignet, da in größeren Netzwerken sonst relativ schnell ein signifikanter Teil der verfügbaren Bandbreite von der Suche und Ankündigung der Dienste beansprucht wird [MBB09].

Ein weiteres Problem bei der Nutzung von Protokollen für die Dienstsuche, die für die Nutzung in stationären Netzwerken entwickelt wurden, liegt in der Annahme, dass sich die Struktur eines Netzwerks nicht oder nur langsam verändert. Der laufend wechselnde Kontext eines mobilen Gerätes führt allerdings dazu, dass sich auch die Konnektivität laufend ändert und zum Zeitpunkt der Suche noch verfügbare Dienste möglicherweise kurz darauf nicht mehr verfügbar sind. Entsprechend gilt es, bei der Dienstsuche in mobilen Umgebungen einen Kompromiss zwischen dem Vorhalten eines möglichst aktuellen Verzeichnisses verfügbarer Dienste und den begrenzten Energie- und Bandbreitenressourcen mobiler Geräte einzugehen.

Implikationen für diese Arbeit Im Hinblick auf die zuvor formulierten Anforderungen der Mobilität, Robustheit, Effizienz und Skalierbarkeit soll an dieser Stelle festgehalten werden, dass im Zusammenhang mit der Realisierung einer kontextadaptiven Anwendungsarchitektur bestehende Ansätze zur Dienstsu-

che nur bedingt genutzt werden können. Da die zu unterstützende Mobilität als hoch und die Größe des Netzwerks als klein bis mittel angesehen werden kann, bietet sich nach [MBB09] die Implementierung einer verzeichnislosen Architektur mithilfe eines Overlay-Netzwerks an. In diesem Zusammenhang soll es einerseits möglich sein, aktiv nach Diensten zu suchen sowie auch bestehende Referenzen auf Dienste für eine spätere Verwendung lokal zu speichern.

5.3.3. Analyse des Laufzeitverhaltens und Profiling

Ob und wie eine mobile Anwendung im Hinblick auf die in Abschnitt 5.2.3 vorgestellten Ziele der Adaption angepasst werden kann, kann nur beurteilt werden, wenn die Ergebnisse der Anpassung auch tatsächlich gemessen werden. Hierzu gilt es, das Laufzeitverhalten einer mobilen Anwendung zu untersuchen und die Auswirkung der Ausführung dieser mobilen Anwendung auf das darunterliegende System zu berücksichtigen. Werkzeuge, die das Laufzeitverhalten anderer Anwendungen analysieren, werden auch als *Profiler* bezeichnet. Diese können nach [LAS+15] in die folgenden drei Kategorien unterteilt werden:

- Hardware-Profiler: Sie sammeln Informationen zum Zustand des mobilen Gerätes selbst, dies umfasst beispielsweise die aktuelle Auslastung der CPU sowie des Arbeitsspeichers und den verbleibenden Energievorrat des mobilen Gerätes.

- Software-Profiler: Sie analysieren das Laufzeitverhalten von Anwendungen und sammeln zur Laufzeit einer Anwendung Kriterien wie die benötigte CPU-Zeit und den belegten Arbeitsspeicher. Sie liefern zudem Informationen darüber, welche Daten die Anwendung liest, schreibt oder mit anderen Geräten austauscht.

  Zusätzlich können Software-Profiler aber insbesondere auch die Abhängigkeiten zwischen einzelnen Teilen der Geschäftslogik durch die Analyse von Abhängigkeiten zwischen strukturgebenden Merkmalen des Programmcodes wie Methoden oder Komponenten aufzeigen. In diesem Zusammenhang werden insbesondere Parameter wie die Ausführungszeit einzelner Methoden, die Speichernutzung und der Energieverbrauch, der hierbei auftritt, gemessen. Sogenannte *Call-graph Profiler* sind explizit auf diese Messungen und die damit einhergehende Ermittlung der Abhängigkeiten und entsprechenden Aufrufsequenzen der einzelnen Einheiten der Geschäftslogik untereinander ausgerichtet.

- Netzwerk-Profiler: Sie erfassen Informationen, die im Zusammenhang mit der Nutzung der Schnittstellen zur Mobilkommunikation stehen. Diese umfassen die Konnektivität, erreichbare Übertragungsrate und tatsächlich genutzte Bandbreite eines mobilen Geräts auf den jeweiligen Schnittstellen.

Obwohl für die drei genannten Kategorien von Profilern jeweils eine Reihe von Lösungen existiert, sollen im Folgenden die speziellen Herausforderungen bei der Integration dieser Profiler in mobile Anwendungen und Geräte betrachtet werden. Im Hinblick auf die in Abschnitt 4.4 vorgestellten Lösungen (vergleiche [CIM+11, CBC+10, KAH+12, MLW11] und im Hinblick auf die in Abschnitt 5.2.3 vorgestellten Adaptionsziele sollen dabei die jeweils für die Erfassung durch den jeweiligen Profiler als relevant identifizierten Informationen herausgestellt werden.

- Hardware-Profiler: Hardware-Profiler erfassen üblicherweise systemweite Zustände, die nicht oder nur bedingt auf einzelne Ereignisse oder Anwendungen zurückgeführt werden können. In diesem Zusammenhang werden die folgenden Informationen als relevant angesehen:
  - Auslastung von CPU und Speicher
  - Energievorrat und Ladezustand
  - Status und Nutzung der Schnittstellen zur Mobilkommunikation
  - Status wesentlicher Energieverbraucher

  Als wesentliche Energieverbraucher werden in diesem Zusammenhang das Display und die Schnittstellen zur Mobilkommunikation angesehen [MN10].
- Software-Profiler: Im Gegensatz zur vorgenannten Kategorie ermitteln Software-Profiler üblicherweise anwendungsbezogene Informationen und Zustände. Insbesondere für diese Kategorie eines Profilers existieren eine Reihe vorhandender Lösungen, die sich üblicherweise in die verwendete Entwicklungsumgebung integrieren lassen. Für diese Messung ergänzt der Profiler üblicherweise den Quellcode einer Anwendung oder nutzt eine vorhandene Schnittstelle zu einer Laufzeitumgebung wie der Java Virtual Machine, die diese Anwendung ausführt. Dieses Vorgehen kann prinzipiell auf mobile Geräte und die dort verwendeten mobilen Betriebssysteme übertragen werden, jedoch gilt es dabei, die üblicherweise vorhandene Abgrenzung mobiler Anwendungen gegeneinander (sandboxing) zu berücksichtigen. In diesem Zusammenhang werden die folgenden Informationen für den zu realisierenden Software-Profiler als relevant betrachtet:
  - Ausgeführte Anwendung
  - Gesamter Speicherbedarf
  - Ausführungszeit und CPU-Zeit einzelner Teile der Geschäftslogik
  - Speicherbedarf einzelner Teile der Geschäftslogik
  - Energiebedarf einzelner Teile der Geschäftslogik
  - Interaktion zwischen verschiedenen Teilen der Geschäftslogik
  - Parameter der Aufrufe von Methoden/Diensten und die Größe übergebener Objekte

Hierbei kann allerdings eine der benötigten Informationen nur indirekt von einem Software-Profiler gemessen werden: die benötigte Energie.

Um den Energiebedarf zu ermitteln, kann ein Software-Profiler zwar prinzipiell auf einen Hardware-Profiler zurückgreifen und den Energievorrat vor und nach der Ausführung eines bestimmten Teils der Geschäftslogik zur Abschätzung der hierfür benötigten Energie heranziehen, diese Messung ist allerdings nur bedingt genau. Unter anderem deswegen, weil bei aktuellen mobilen Betriebssystemen stets eine Reihe von Basisdiensten und Hintergrunddiensten, zum Beispiel die Standortermittlung via GPS), in unregelmäßigen Abständen aktiv ist und hierdurch nur näherungsweise ermittelt werden kann, welcher Teil des Energieverbrauchs der mobilen Anwendung zuzuordnen ist. Des Weiteren kann der wechselnde Kontext eines mobilen Gerätes zu unterschiedlichen Energieverbräuchen führen, wenn die Ausführung der Geschäftslogik die Nutzung der Schnittstellen zur Mobilkommunikation auslöst und die verwendete Schnittstelle und die verfügbare Bandbreite nicht identisch sind [MN10]. Selbst unter gleichen Bedingungen sind hier Schwankungen des Energiebedarfs nicht untypisch [HQG+13]. Zusätzlich gilt es zu berücksichtigen, dass mobile Geräte eine Reihe von Energiesparfunktionen nutzen und den Energieverbrauch zusätzlich reduzieren, wenn das thermische Budget der Geräte ausgeschöpft ist [Fru16, BC12].

Entsprechend gilt es zur Messung des Energieverbrauchs einerseits, alle nicht benötigten Dienste des mobilen Betriebssystems zu deaktivieren, um die Grundlast möglichst konstant zu halten. Andererseits gilt es, ein Messverfahren zu verwenden, welches in der Lage ist, mit einem möglichst geringen Fehler den Energiebedarf der einzelnen Teile der Geschäftslogik zu ermitteln. Wie in Abschnitt 5.2.6 festgehalten wurde, stellen diese Teile der Geschäftslogik die jeweiligen Dienste der durch den Entwickler einer mobilen Anwendung definierten Komponenten und die von ihnen bereitgestellten Dienste dar. Ein Vergleich verwandter Arbeiten mit gleicher Problemstellung [CIM+11, CBC+10, KAH+12, MLW11] und die Analyse existierender Energie-Profiler in [Bak14] zeigt, dass hierzu eine ergänzende hardwarebasierte Messung notwendig ist.

So wurde in einer untersuchten Lösung [CBC+10] zur Optimierung des Energievorrats mobiler Geräte eine hardwarebasierte Messung verwendet, um ein Energie-Profil für das jeweilige mobile Gerät zu ermitteln. Auf Basis dieses Energie-Profils, welches detaillierte Informationen zum Energieverbrauch der einzelnen Komponenten des mobilen Geräts enthält, wird anschließend durch mehrfache Ausführung des zu messenden Teils der Geschäftslogik die tatsächlich für die Ausführung benötigte Energie ermittelt. Dieses Vorgehen entspricht damit dem von Kansal et al. in [KZ08] vorgestellten Verfahren zur feingranularen Messung des Energieverbrauchs von Anwendungen.

- Netzwerk-Profiler: Der dritte zu realisierende Profiler bringt ebenso eine Reihe von Herausforderungen mit sich. Mit Blick auf die verwandten Arbeiten und die zu realisierenden Adaptionsziele werden für diesen die folgenden Informationen als relevant für die Erfassung angesehen:
  - Genutzte Schnittstelle(n) zur Mobilkommunikation
  - Genutztes/Eingebuchtes Netzwerk
  - Genutzte Bandbreite von und zum mobilen Gerät
  - Theoretisch nutzbare Bandbreite von und zum mobilen Gerät
  - Tatsächlich realisierbare Bandbreite von und zum mobilen Gerät
  - Paketumlaufzeit (RTT)

  Zur Ermittlung der genutzten Schnittstellen und Bandbreiten lassen sich passive Messverfahren nutzen, die durch Beobachtung der empfangenen und versendeten Pakete die Nutzung erfassen. Die Ermittlung der tatsächlich realisierbaren Bandbreite über die einzelnen Schnittstellen zur Mobilkommunikation ist hingegen deutlich schwieriger durch Beobachtung zu realisieren. Existierende Ansätze wie [GPSV10, MBF+15] erfordern hierzu die häufige oder laufende Aktivierung der Schnittstellen zur Mobilkommunikation und beanspruchen den Energievorrat mobiler Geräte oft stark. Ein zusätzliches Problem steht durch die im TCP-Protokoll implementierte Flusskontrolle. Dies erfordert zunächst, eine bestimmte Menge von Daten zu übertragen, um die tatsächlich realisierbare Bandbreite messen zu können [WSW13], eine Problemstellung, die sich bei der Nutzung des Mobilfunkstandards LTE noch weiter verstärkt [HQG+13], da hier die Zuteilung der verfügbaren Kapazität auch auf Basis des jeweiligen Bedarfs an Bandbreite erfolgt.

  Um den genannten Problemen zu begegnen, nutzen existierende Lösungen oft eine aktive Messung der Bandbreite. In diesem Zusammenhang wird von Cuervo et al. in [CBC+10] einerseits der Effekt der Energiesparmodi der Schnittstellen zur Mobilkommunikation untersucht und zur Abschätzung der realisierbaren Bandbreite in einem WLAN das Senden von 10-KB-Daten auf einer bereits offenen TCP-Verbindung als praktikabel angesehen.

Ein generelles Problem, insbesondere von Software-Profilern, ist somit, dass die beobachtete Anwendung durch das Profiling ein anderes Verhalten aufweisen kann und das Profiling in der Regel auf einem mobilen Gerät ausgeführt wird, was zu einem höheren Energieverbrauch führen kann. Ähnlich hierzu verhält es sich mit Netzwerk-Profilern, deren Messungen erst bei der vollen Auslastung der Kapazität der Schnittstelle zur Mobilkommunikation Rückschlüsse auf die erzielbaren Übertragungsraten zulassen, wie unter anderem in [HQG+13] gezeigt wird.

Implikationen für diese Arbeit Zusammenfassend gilt es dementsprechend bei der Implementierung von Profiling-Mechanismen zu berücksichtigen, dass der begrenzte Energie- und Bandbreitenbedarf mobiler Geräte durch das Profiling nicht zu stark beansprucht wird und dass dieser Einfluss möglichst dem Profiling selbst zugeordnet werden kann. Im Hinblick auf die in dieser Arbeit untersuchte Fragestellung der kontextadaptiven Ausführung existieren eine Reihe weiterer Kontextattribute, die auch den das Gerät umgebenden Kontext beschreiben und die relevant für eine entsprechende Adaption sein können. Diese werden jedoch erst im Zusammenhang mit der Entwicklung kontextadaptiven Verhaltens untersucht werden, da diese Informationen eher periodisch und nicht wie im Profiling nutzungsbezogen im Zusammenhang mit einer bestimmten Anwendung ermittelt werden.

5.3.4. Partitionierung mobiler Anwendungen

Wie im Zusammenhang mit der Vorstellung der Basisarchitektur in Abschnitt 5.1 gezeigt wurde, ist es im mobilen Cloud Computing oft erforderlich, die Geschäftslogik einer mobilen Anwendung zu partitionieren. Ob eine gute Partitionierung einer mobilen Anwendung erreicht werden kann, ist von einer Reihe verschiedener Parameter abhängig und wird als eine der zentralen Problemstellungen im Bereich des mobilen Cloud Computings angesehen [PRK09]. Dies ist unter anderem darin begründet, dass die optimale Partitionierung einer mobilen Anwendung entscheidend für eine gute Auslagerungsentscheidung ist [Fli12], also dem Schritt, der sich an die Partitionierung anschließt und in welchem entschieden wird, ob eine bestimmte Partition der Geschäftslogik auf ein Surrogate ausgelagert wird.

5.3.4.1. Ziele der Partitionierung

Die Partitionierung zielt darauf ab, eine sinnvolle Kooperation mit der Infrastruktur zu realisieren, und bildet die Grundlage dafür, eine Verteilung der Geschäftslogik auf ein Surrogate vorzubereiten und damit die Ausführung innerhalb der Dimension des Ausführungsortes zu flexibilisieren - eine Problemstellung, die es nicht nur im Bereich des mobilen Cloud Computings zu untersuchen gilt. So untersucht [Zap12] ebenfalls die Partitionierung, allerdings mit dem abweichenden Ziel der Flexibilisierung der Prozessausführung. Den Untersuchungsgegenstand dieser Arbeit stellten entsprechend Geschäftsprozesse und deren Ausführung innerhalb von Prozessmanagementsystemen dar. In der vorliegenden Arbeit wird hingegen die Entwicklung kontextadaptiver Anwendungen für das mobile Cloud Computing fokussiert. Den Anwendungsbereich stellen dementsprechend mobile Geräte dar, auf denen das Ziel der Partitionierung die Ressourceneinsparung ist und die auf den schnell wechselnden Kontext des mobilen Cloud Computings ausgerichtet sind. Aus diesem Grund konzentriert sich die folgende Analyse von Lösungsansätzen auf diesen Anwendungsbereich.

Nach [LAS+15] gilt es, bei der Partitionierung von Anwendungen im mobilen Cloud Computing, die folgenden Anforderungen zu unterstützen:

- Die Umsetzung unterschiedlicher Adaptionsziele: wie die Beschleunigung der Ausführung oder die Erweiterung der Ressourcen (zum Beispiel Speicher oder Energie) eines mobilen Gerätes.
- Die Unterstützung mehrerer Surrogates (multi-site offloading).
- Benutzbarkeit des Partitionierungsverfahrens: Der zusätzliche Aufwand für den Entwickler sollte möglichst gering sein.
- Adaption zur Laufzeit: Die Anpassung der Partitionierung sollte zur Laufzeit einer mobilen Anwendung erfolgen können.

5.3.4.2. Durchführung der Partitionierung

Zur Durchführung einer Partitionierung ist es zunächst erforderlich, eine entsprechende Granularität der Adaption (vergleiche Abschnitt 5.2.6) für die einzelnen Einheiten einer mobilen Anwendung auszuwählen. Anschließend kann unter Berücksichtigung des oder der gewünschten Ziele der Adaption (vergleiche Abschnitt 5.2.3) eine Partitionierung erfolgen.

Dabei ist es erforderlich, diese Einheiten um weitere Informationen zu ergänzen, die beschreiben, auf welchen Surrogates diese Teile der Geschäftslogik unter welchen Umständen ausgeführt werden können. Beispielsweise gilt es, zu entscheiden, ob bestimmte Teile der Geschäftslogik überhaupt in die Infrastruktur verlagert werden können, weil diese spezifische Funktionen oder Hardware des mobilen Gerätes nutzen. Ein einfaches Verfahren findet sich beispielsweise in [SSSL12], bei welchem auf Basis der von den Entwicklern definierten Java-basierten Annotationen festgelegt wird, welche Teile einer mobilen Anwendung überhaupt für eine Migration in die Infrastruktur infrage kommen, bevor die verbleibende Geschäftslogik partitioniert wird.

In diesem Zusammenhang ist anzumerken, dass ein Großteil der verwandten Arbeiten sich in diesem Aspekt auf eine binäre Partitionierung beschränkt und entsprechend lediglich ein (spezifisches) Surrogate bei der Auslagerung von Funktionalität unterstützt wird. Eine Restriktion, die im Verlauf dieser Arbeit noch gesondert und die entsprechend an dieser Stelle ebenso nicht näher betrachtet wird.

Das Ziel der Partitionierung ist es entsprechend, die Geschäftslogik so zwischen einem mobilen Gerät und einem Surrogate aufzuteilen, dass ein oder mehrere Adaptionsziele möglichst gut erreicht werden können. Um beispielsweise die Ausführungszeit zu minimieren, gilt es häufig, den Teil des Kontrollflusses zu identifizieren und als eine Partition zu markieren, der effizient auf ein Surrogate übertragen und dort ausgeführt werden kann. Die Effizienz wird dabei anhand entstehender Kosten, wie beispielsweise der nötigen Bandbreite, beurteilt, die für die Übertragung des Zustands zwischen dem mobilen Gerät und einem Surrogate entstehen.

Bezüglich der Durchführung dieser Partitionierung lassen sich nach [LAS+15] drei Gruppen von Verfahren unterscheiden, nach denen eine Partitionierung durchgeführt werden kann: die Gruppe der graph-basierten Verfahren, die Gruppe der Verfahren, die auf linearer Programmierung basieren, und hybride Verfahren, die eine Kombination der beiden zuvorgenannten Ansätze verwenden. Im Folgenden sollen die unterschiedlichen Verfahren jeweils kurz beschrieben werden:

Graph-basierte Verfahren Häufig erfolgt die Ermittlung einzelner Partitionen einer mobilen Anwendung auf Basis eines Graphen. Die Knoten des Graphen repräsentieren dabei die einzelnen Einheiten einer mobilen Anwendung und seine Kanten die Abhängigkeiten, die im Hinblick auf das jeweilige Ziel der Adaption relevant sind. Hierbei handelt es sich um Informationen wie die Menge der durchschnittlich ausgetauschten Daten zwischen zwei Partitionen, die Frequenz der jeweiligen wechselseitigen Aufrufe, den Daten- oder Kontrollfluss oder zu die zu erwartenden Wartezeiten, die sich beim Aufruf zwischen Komponenten ergeben.

Zur Nutzung dieser Verfahren gilt es zunächst, die Einheiten der Geschäftslogik, beispielsweise Komponenten, einer mobilen Anwendung und ihre Interaktion untereinander zu beschreiben. Dieser Zusammhang kann, angelehnt an [KNP15], als ein ungerichteter Graph M, bestehend aus einer Menge von Komponenten (N) einer mobilen Anwendung und einer Menge von Verbindungen (E) zwischen diesen Komponenten, beschrieben werden, die wie folgt definiert sind:

$$M = (N,\ E,\ c_\mathrm{m},\ c_\mathrm{c},\ c_\mathrm{t},\ t_\mathrm{m},\ t_\mathrm{c},\ t_\mathrm{t}) \tag{5.1}$$

$$E \subseteq \{\{n_i, n_j\} : n_i \in N \wedge n_j \in N \wedge n_i \neq n_j\} \tag{5.2}$$

$$c_\mathrm{m}\colon N \to \mathbb{R},\ c_\mathrm{c}\colon N \to \mathbb{R},\ c_\mathrm{t}\colon E \to \mathbb{R} \tag{5.3}$$

$$t_\mathrm{m}\colon N \to \mathbb{R},\ t_\mathrm{c}\colon N \to \mathbb{R},\ t_\mathrm{t}\colon E \to \mathbb{R} \tag{5.4}$$

Für einen Knoten (das heißt eine Komponente) $n \in N$ sind dabei folgende vier Gewichtsfunktionen definiert:

- $t_\mathrm{m}(n)$ entspricht ihrer Ausführungszeit auf dem mobilen Gerät,
- $t_\mathrm{c}(n)$ entspricht ihrer Ausführungszeit auf dem Surrogate,
- $c_\mathrm{m}(n)$ entspricht ihrer Ausführungskosten auf dem mobilen Gerät,
- $c_\mathrm{c}(n)$ entspricht ihrer Ausführungskosten auf dem Surrogate.

Für eine Kante $e \in E$ sind darüber hinaus zwei Gewichtsfunktionen definiert:

- $t_\mathrm{t}(e)$ entspricht der Zeit für die Synchronisation des für den Aufruf einer Komponente erforderlichen Zustands (state) zwischen dem mobilen Gerät und dem Surrogate,

- $c_{\mathrm{t}}(e)$ entspricht den Kosten für die Synchronisation dieses Zustands (beispielsweise der erforderlichen Bandbreite) zwischen dem mobilen Gerät und dem Surrogate.

Hierbei sind allerdings die Kosten für die Übertragung des Zustands zwischen zwei Komponenten auf demselben Gerät (mobil oder Surrogate) zu vernachlässigen. Entsprechend lassen sich für eine Teilmenge von Komponenten auf dem mobilen Gerät (N_{m}) und eine zweite Teilmenge auf einem Surrogate (N_{c}) die Kosten der Auslagerung wie folgt ermitteln:

$$N_{\mathrm{c}} \subset N,\ N_{\mathrm{m}} = N \setminus N_{\mathrm{c}} \tag{5.5}$$

$$E_{\mathrm{t}} = \{\{n_{\mathrm{c}}, n_{\mathrm{m}}\} : \{n_{\mathrm{c}}, n_{\mathrm{m}}\} \in E \wedge n_{\mathrm{c}} \in N_{\mathrm{c}} \wedge n_{\mathrm{m}} \in N_{\mathrm{m}}\} \tag{5.6}$$

$$c = \alpha \sum_{n \in N_{\mathrm{c}}} c_{\mathrm{c}}(n) + \beta \sum_{n \in N_{\mathrm{m}}} c_{\mathrm{m}}(n) + \gamma \sum_{e \in E_{\mathrm{t}}} c_{\mathrm{t}}(e) \tag{5.7}$$

Die Koeffizienten α, β und γ definieren dabei die Gewichtung der einzelnen Größen, aus denen sich die für das jeweilige Ziel der Adaption relevanten Kosten ergeben. Tabelle 5.3 zeigt hierzu exemplarisch die unterschiedlichen Gewichtungen dieser Koeffizienten, die sich im Hinblick auf das jeweilige Adaptionsziel ergeben. Beispielsweise wird eine Einsparung von Energie auf dem mobilen Gerät dadurch realisiert, dass der Parameter β, verantwortlich für die Gewichtung der Ausführungskosten (in diesem Fall die Energie) auf dem mobilen Gerät, maximiert wird.

Optimierungsziel	α	β	γ
Einsparung von Energie (mobil)	0	1	0
Einsparung von Bandbreite	0	0	1
Kombinierte Einsparung	0,4	0,3	0,3

Tabelle 5.3.: Exemplarische Gewichtung im Hinblick auf das jeweilige Adaptionsziel

Diese graph-basierte Modellierung ermöglicht es, ein Optimierungsproblem zu formulieren. Beispielsweise lässt sich so die Teilmenge von Komponenten finden, die in der Infrastruktur ausgeführt werden kann, während die Ausführungszeit unterhalb eines bestimmten Maximums $t_{\max}$ gehalten wird:

$$\min_{N_{\mathrm{c}} \in \mathbb{P}(N)} \alpha \sum_{n \in N_{\mathrm{c}}} c_{\mathrm{c}}(n) + \beta \sum_{n \in N_{\mathrm{m}}} c_{\mathrm{m}}(n) + \gamma \sum_{e \in E_{\mathrm{t}}} c_{\mathrm{t}}(e) \tag{5.8}$$

Wobei $\mathbb{P}(N)$ die Potenzmenge der Knotenmenge N darstellt. Der Schwellwert $t \leq t_{\max}$ kann dabei beispielsweise auf die durchschnittliche Ausführungszeit der jeweiligen Anwendung auf dem mobilen Gerät gesetzt werden, um zu verhindern, dass es zu einer subjektiven Einschränkung der Dienstqualität kommt.

Alternativ lassen sich die (gewichteten) Kosten auch nicht nur, wie in Gleichung 5.7 definiert wurde, kollektiv betrachten, sondern auch pro Kante des definierten Abhängigkeitsgraphen, um eine rein graph-basierte Trennung durchzuführen. In diesem Zusammenhang hat sich die Anwendung des MIN-CUT-Algorithmus [SW97] zur binären Partitionierung als nützlich erwiesen, um entweder die Interaktion zwischen den so entstehenden Partitionen der mobilen Anwendung zu optimieren [CM10] oder unter Berücksichtigung eines Adaptionsziels jeweils die Modellierung des hierbei entstehenden Abhängigkeitsgraphen dieser mobilen Anwendung zu konkretisieren, wie es in einer Reihe verwandter Arbeiten vorgeschlagen wurde [OYL06, OYZ07, GNM+03, MGB+02, WKWS16].

Nach [LAS+15] kann ein solches graph-basiertes Modell, unabhängig davon, ob es grob- oder feingranular ist, einen effizienten Weg zur Partitionierung einer mobilen Anwendung darstellen und soll entsprechend näher vorgestellt werden.

Partitionierung auf Basis linearer Programmierung Lineare Programmierung stellt ein etabliertes Verfahren dar, um Optimierungsprobleme, zum Beispiel die Minimierung der Ausführungszeit im mobilen Cloud Computing, zu lösen. Hierzu existiert eine Zielfunktion, deren Ergebnis es zu maximieren bzw. minimieren gilt. In [KCK11] wird diese Modellierung als Optimierungsproblem mit den in Bezug auf das jeweilige Ziel der Adaption relevanten Parametern gezeigt. Diese Gruppe von Verfahren liefert so zwar immer ein optimales Ergebnis für eine bestimmte Zielfunktion, nach [LAS+15] verhindert der damit einhergehende hohe Berechnungsoverhead allerdings oft die praktische Nutzbarkeit dieser Verfahren in ihrer reinen Form.

Hybride Verfahren zur Partitionierung Hybride Verfahren kombinieren die beiden zuvor genannten Konzepte. So wird unter anderem in [CBC+10] zunächst mithilfe eines Profilings ein Aufrufgraph zwischen den einzelnen Methoden einer mobilen Anwendung erstellt, um hierauf basierend im Anschluss ein Optimierungsproblem zu lösen. Als effizienteste Variante dieser Kategorie von Partitionierungsverfahren wird in [LAS+15] die Kombination einer ganzzahligen linearen Optimierung mit einer graph-basierten Abbildung des Datenflusses einer mobilen Anwendung angesehen, wie sie beispielsweise in [GRJ+09] umgesetzt ist.

5.3.4.3. Weitere Unterscheidungsmerkmale von Partitionierungsverfahren

Die verschiedenen Verfahren zur Partitionierung einer mobilen Anwendung können weitergehend darin unterschieden werden, ob sie die Partitionierung statisch oder dynamisch durchführen [LAS+15]. Die statische Partitionierung, wie sie beispielsweise in [CIM+11, SSSL12, NTG+09, BGSH07] Verwendung findet, stützt sich dabei auf eine Analyse des Quellcodes oder des Bytecodes einer bereits kompilierten Anwendung. Die dynamische Partitionierung hinge-

gen nutzt zur Analyse das in Abschnitt 5.3.3 beschriebene Software-Profiling und untersucht das Laufzeitverhalten, um Rückschlüsse auf eine sinnvolle Aufteilung der Geschäftslogik im Hinblick auf die vorgegebenen Adaptionsziele zu ermitteln.

Im Weiteren können Partitionierungsverfahren nach [LAS+15] auch danach unterteilt werden, ob sie das Profiling vollständig automatisiert durchführen oder ob sie entsprechende Hinweise des Entwicklers in Form von Annotationen berücksichtigen. Letztere Variante kann díe Ermittlung der Partitionierung deutlich vereinfachen, da beispielsweise Teile der Geschäftslogik, die aus Wahrnehmung der Nutzer für eine nahtlose und verzögerungsfreie Ausführung der Anwendung verantwortlich sind, nicht in die Infrastruktur verlagert werden. Ein Kriterium, das mithilfe eines vollautomatisierten Profilings nur schwer ermittelt werden kann.

5.3.4.4. Implikationen für diese Arbeit

Im Hinblick auf die Realisierung einer kontextadaptiven Anwendungsarchitektur sollen ebenfalls auf Basis eines Software-Profilings zunächst die Abhängigkeiten zwischen den einzelnen Komponenten, die als Granularität für die Adaption ausgewählt wurden, und den von ihnen bereitgestellten Diensten ermittelt werden.

Abweichend zu den gezeigten Verfahren soll anschließend jedoch zweistufig vorgegangen werden: In einem ersten Schritt soll im Hinblick auf das jeweilige Adaptionsziel eine Unterteilung in lokal und in entfernt ausführbare Teile der Geschäftslogik erfolgen, wie es bereits erfolgreich in verschiedenen existierenden Arbeiten umgesetzt wurde [GNM+03, MGB+02, OYH07]. Entsprechend existiert für jedes zu unterstützende Adaptionsziel jeweils eine Partitionierung der mobilen Anwendung. Diese wird genutzt, um die einzelnen Komponenten auf dem mobilen Gerät und seinem Surrogate entsprechend zu starten. Hierbei wird jedoch versucht, die auf dem Surrogate zu startenden Komponenten ebenfalls auf dem mobilen Gerät zu starten, sofern hierfür ausreichend Ressourcen vorhanden sind. Dies dient der Vorbereitung des zweiten Schrittes.

Um angemessen auf den aktuellen Kontext reagieren zu können, erfolgt anschließend im zweiten Schritt jedoch eine kontextabhängige Entscheidung, die sich ebenfalls an den unterschiedlichen Zielen der Adaption orientiert. Hierzu werden auf dem mobilen Gerät und den verfügbaren Surrogates die jeweils übrigen, also die der jeweils anderen Partition zugehörigen, Komponenten der mobilen Anwendung gestartet. Dies dient dazu, abhängig von Kontextattributen wie der verfügbaren Bandbreite und der Art des jeweiligen Tasks in Bezug auf das entsprechende Ziel der Adaption zu entscheiden, ob der Aufruf einer Komponente auf die lokale oder die auf dem Surrogate vorhandene Instanz weitergeleitet wird. Mithilfe dieses Ansatzes soll die Komplexität der Partitionierung und der darauf aufbauenden Auslagerungsentscheidung möglichst effizient gelöst und trotzdem den formulierten Anforderungen der spontanen Interaktion angemessen gerecht werden.

5.4. Konzeption einer adaptiven Architektur und Systemunterstützung

Nachdem in den vorhergehenden Abschnitten die wesentlichen Teilprobleme und entsprechende Lösungsansätze vorgestellt wurden, wird im nun folgenden Abschnitt ein Konzept für eine kontextadaptive Anwendungsarchitektur entwickelt werden. Im Hinblick auf die in Abschnitt 4.2 ermittelten Anforderungen werden hierzu die in Abschnitt 5.1 entwickelte Basisarchitektur für mobiles Cloud Computing und der dort beschriebene Auslagerungsprozess genutzt, um die Unterstützung der in Abschnitt 5.2 formulierten Adaptionsbedarfe zu etablieren. Anschließend werden wesentliche Aspekte und Komponenten des Konzepts weiter detailliert.

Entsprechend der in Abschnitt 1.5 vorgestellten Entwicklungsmethodik wird hierzu ein evolutionäres Vorgehensmodell angewendet, bei dem auf Grundlage der Basisarchitektur für mobiles Cloud Computing eine Beispielanwendung umgesetzt und sukzessive um prototypische Implementierungen der neu zu entwickelnden Konzepte ergänzt wird. Das hierdurch entwickelte Konzept wird im Folgenden vorgestellt sowie im weiteren Verlauf der Arbeit qualitativ und anhand der Beispielanwendung quantitativ evaluiert.

5.4.1. Erweiterung der Basisarchitektur

Die Entwicklung kontextadaptiver Anwendungen für das mobile Cloud Computing bringt eine Reihe von Herausforderungen mit sich. Einerseits ist es erforderlich, dass diese Anwendung den klassischen Herausforderungen verteilter Systeme wie der Heterogenität der beteiligten Systeme begegnet. Zusätzlich soll sie den in Abschnitt 2.1 beschriebenen Anforderungen an Offenheit, Sicherheit, Skalierbarkeit, Fehlerbehandlung, Nebenläufigkeit, Transparenzeigenschaften und Dienstqualität entsprechnen. Andererseits existiert eine Reihe weiterer Herausforderungen, die sich auf die begrenzten Ressourcen und den schnell wechselnden Kontext mobiler Geräte beziehen, insbesondere die wechselnde Konnektivität dieser Geräte.

Entwickler entsprechender Anwendung sehen sich somit einer großen Anzahl von Herausforderungen gleichzeitig konfrontiert. Etablierte Mechanismen, beispielsweise zur Interprozesskommunikation oder zum entfernten Methodenaufruf (RMI), können in diesem Zusammenhang jedoch oft nicht eingesetzt werden [PRK09]. Um Entwicklern mobiler Anwendungen trotzdem eine angemessene Unterstützung bereitzustellen, setzt der Großteil existierender Ansätze auf eine weitreichende Unterstützung in Form einer Middleware, eines Frameworks oder nutzt eine Form der Virtualisierung auf Anwendungs- oder auf Systemebene (vergleiche Abschnitt 4.3).

Um den zusätzlichen Anforderungen der Kontextsensitivität der Architektur gerecht zu werden, bietet sich insbesondere eine Systemunterstützung in Form einer (mobilen) Middleware an [YLM14, LEMR15]. Der hierdurch mögliche flexible Umgang mit Transparenzeigenschaften im Zusammenhang mit der Um-

setzung kontextbewusster (mobiler) Anwendungen wurde bereits mehrfach erfolgreich gezeigt [HIMB05, PP14, GPZ05, MM09]. Gleichzeitig ermöglicht eine Middleware die einfache Integration eines komponentenbasierten Entwurfs. Entsprechend soll die Umsetzung einer Basisarchitektur für mobiles Cloud Computing auf Basis einer Middleware für mobile Systeme erfolgen, die im Folgenden vorgestellt wird.

Mit Blick auf die qualitative Evaluation existierender Lösungsansätze in Abschnitt 4.4 ist es zur Konkretisierung der Basisarchitektur hin zu einer kontextadaptiven Anwendungsarchitektur vor allem erforderlich, die Anwendungsarchitektur und Systemunterstützung dahingehend zu erweitern, die kurzweilige und spontane Interaktion mit verfügbaren Ressourcen in der Infrastruktur zu unterstützen. Um die in Abschnitt 5.2 ermittelten Adaptionsziele zu unterstützen, ist es gleichzeitig erforderlich, die Fähigkeit zur (proaktiven) Adaption an einen wechselnden Nutzungskontext zu berücksichtigen und hierzu das Kontextmanagement und die Koordinationsfunktion einer entsprechenden Anwendungsarchitektur und Systemunterstützung anzupassen. Hieraus ergibt sich das Erfordernis der Erweiterung der Basisarchitektur um zusätzliche Fähigkeiten des Kontextmanagers. Diese Fähigkeiten umfassen die Akquisition, Speicherung und Auswertung von Kontextdaten, um sie im Hinblick auf Adaptionsentscheidungen zu berücksichtigen. Wie die Kontextinformationen in diesem Zusammenhang verarbeitet werden, wird in Abschnitt 5.5 konkretisiert, in Abschnitt 5.4 wird entsprechend lediglich die Schnittstelle des Kontextmanagers beschrieben.

Wie auf dieser Basisarchitektur Anwendungen für das mobile Cloud Computing entwickelt und ausgeführt werden können und wie diese aufgebaut sowie strukturiert sind, soll nun im Folgenden vorgestellt werden, indem einerseits das Konzept der taskbasierten Adaption eingeführt wird und darauffolgend die erforderlichen Infrastrukturkomponenten einer Systemunterstützung näher vorgestellt werden.

5.4.2. Konzept der taskbasierten Adaption

Die Entscheidung über eine Auslagerung von Berechnungen im mobilen Cloud Computing unter Berücksichtigung des schnell wechselnden Kontextes mobiler Umgebungen wird als eine der zentralen Problemstellungen des mobilen Cloud Computings beschrieben [Fli12]. Diese Problemstellung bezieht sich unter anderem auf die Erreichbarkeit von Surrogates, die sich aufgrund des wechselnden Kontextes und damit auch der wechselnden Konnektivität eines mobilen Gerätes häufig ändert [WHB13] und nur mit verhältnismäßig großem Aufwand überhaupt abzuschätzen ist [MS14, BMV10, GNS+14]. Ebenso wurde in [RVGH11] festgestellt, dass eine Adaptionsentscheidung wie die Auslagerungsentscheidung eine Reihe von Aspekten berücksichtigen sollte, allen voran die Mobilität des Nutzers und seines mobilen Geräts und die damit einhergehenden Wechsel der Standards zur Mobilkommunikation, die zum Einsatz kommen, da diese einen wesentlichen Einfluss auf die Bandbreite und Latenz

der verwendeten Surrogates haben. Entsprechend soll festgehalten werden, dass es relevant ist, den aktuellen Nutzungskontext als Grundlage für eine Adaptionsentscheidung heranzuziehen.

In Bezug auf die Realisierung einer kontextadaptiven Anwendungsarchitektur soll dies dadurch realisiert werden, die Granularität der Adaption nicht alleine auf die einzelnen Komponenten und deren Dienste zu beschränken, sondern zusätzlich den aktuellen Nutzungskontext zu berücksichtigen und hierzu die Granularität der Adaption auf einzelnen Tasks zu erweitern (vergleiche Abschnitt 5.2.6 und 5.3.4). Hierdurch kann beispielsweise erkannt werden, dass eine Verlagerung eines Tasks in die Infrastruktur häufig dann erfolgreich ist, wenn ein mobiles Gerät aufgeladen wird und hierdurch wahrscheinlich entsprechend länger mit demselben Netzwerk verbunden ist, was eine erfolgreiche Rückführung des Ergebnisses der Taskausführung verspricht. Ebenso kann erkannt werden, dass die Verlagerung desselben Tasks häufig scheitert, wenn erkannt wird, dass sich der Nutzer des mobilen Geräts sich bewegt und die Erreichbarkeit unterschiedlicher Surrogates häufig wechselt.

In diesem Zusammenhang soll zur Umsetzung der in Abschnitt 5.2.2 definierten Dimensionen der Adaption das Prinzip der kompositionellen Adaption zum Einsatz kommen. Für jeden zu adaptierenden Task soll dadurch entschieden werden, auf welcher Instanz einer Komponente der Task oder Teilschritte des Tasks verarbeitet werden (Adaptionsform: Verlagerung des Ausführungsortes), wann die Verarbeitung erfolgt (Adaptionsform: zeitliche Verlagerung) und welche Implementierung im Rahmen einer möglichen Fidelity Adaptation genutzt wird. Hierzu durchläuft jeder durch eine mobile Anwendung initiierter Task einen entsprechenden Entscheidungsprozess. Wurde auf Basis dieses Entscheidungsprozesses eine Möglichkeit zur Adaption wie der Verlagerung des Ausführungsortes identifiziert, wird diese weiter dahingehend untersucht, ob eine Auslagerung im Hinblick auf das, für diesen Task definierte Ziel der Adaption sinnvoll ist. Dieses zweistufige Vorgehen vermeidet, dass dieser Koordinationsprozess für Tasks initiiert wird, die das mobile Gerät nicht verlassen sollen oder können. Dieser Koordinationsprozess, der in Abbildung 5.6 veranschaulicht wird, wird durch eine Reihe kontextabhängiger Entscheidungskriterien beeinflusst, die sowohl die aktuellen als auch die zukünftigen Ausprägungen dieser Entscheidungskriterien in Form einer Prognose berücksichtigen. Wie diese ermittelt werden, wird in Abschnitt 5.5 näher erläutert.

Verlagerung des Ausführungsortes In der ersten Variante, der Verlagerung des Ausführungsortes, wird hierzu der Koordinationsprozess durch eine Reihe von Prognosen unterstützt, die, wenn bestimmte Zustände im Prozess der Taskausführung erreicht sind, genutzt werden, um eine kontextabhängige Auslagerungsentscheidung zu unterstützen. Die einzelnen Ausführungszustände und die entsprechenden Prognosen sind wie folgt charakterisiert:

- Task initiiert: Wurde durch den Anwender ein neuer Task initiiert, wird zunächst geprüft, ob dieser überhaupt durch das mobile Gerät selbst aus-

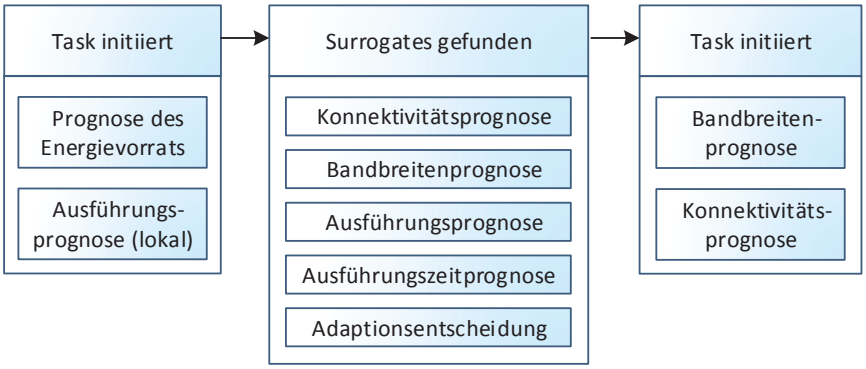

Abbildung 5.6.: Koordinationsprozess für die Verlagerung des Ausführungsortes von Tasks

geführt werden kann. Hierzu werden die folgenden Prognosen durchgeführt:

- ▷ Prognose des Energievorrats: Basierend auf dem verbleibenden Energievorrat und dem prognostizierten zukünftigen Verbrauch erfolgt eine Abschätzung der restlichen Laufzeit des mobilen Geräts.
- ▷ Ausführungsprognose: Es wird vorhergesagt, ob unter Berücksichtigung der verbleibenden Energie und weiterer verfügbarer Kontextinformationen des mobilen Geräts ein erfolgreicher Abschluss des Tasks auf dem mobilen Gerät zu erwarten ist.

► Surrogates gefunden: Wurden im Rahmen der Dienstsuche Surrogates erkannt, die für eine Verlagerung des Ausführungsortes des Tasks in Betracht kommen, wird diese Verlagerung weitergehend untersucht, und es werden hierzu die folgenden Kontextinformation genutzt:

- ▷ Prognose der Konnektivität: Die Konnektivität zum Surrogate für den voraussichtlichen Ausführungszeitraum des Tasks wird abgeschätzt.
- ▷ Prognose der Bandbreite: Die Bandbreite zum Surrogate für den voraussichtlichen Sende- und Empfangszeitraum des Tasks wird abgeschätzt.
- ▷ Ausführungsprognose: Es wird vorhergesagt, ob auf Basis verfügbarer Kontextinformationen und der verbleibenden Energie des mobilen Geräts ein erfolgreicher Abschluss des Tasks auf dem jeweiligen Surrogate zu erwarten ist.
- ▷ Prognose der Ausführungszeit: Es wird vorhergesagt, welche Ausführungszeit für den Task zu erwarten ist.

Unter Berücksichtigung dieser Informationen wird anschließend geprüft, ob eine Auslagerung im Hinblick auf das gewählte Ziel der Adaption sinnvoll ist und gegebenenfalls eingeleitet. Beispielsweise wird für das Adaptionsziel der Minimierung der Ausführungszeit zunächst unter Verwendung der prognostizierten Bandbreite und Ausführungszeit die Dauer der

Taskausführung geschätzt. Anschließend wird geprüft, ob das jeweiliges Surrogate für den Zeitraum der Taskausführung als erreichbar angesehen wird. Mithilfe dieser Informationen wird dann entschieden, ob die Ausführung auf einem der Surrogates vorteilhaft gegenüber der lokalen Ausführung ist und entsprechend initiiert.

Wird hierbei keine Möglichkeit zur Taskausführung erkannt, kann alternativ eine zeitliche Verlagerung oder eine Fidelity-Adaptation durchgeführt werden.

- Verarbeitung beendet: Ist der Task vollständig abgearbeitet, wird unter Berücksichtigung des dann aktuellen Kontextes vorhergesagt, ob zu erwarten ist, dass das Ergebnis vollständig zurückgeführt werden kann. Hierfür wird gegebenenfalls die Prognose der Konnektivität und Bandbreite aktualisiert.

- Ausführung abgeschlossen: Sobald der Task abgeschlossen ist, werden noch blockierte Ressourcen freigegeben.

Zeitliche Verlagerung Wurde neben der Verlagerung des Ausführungsortes zusätzlich oder alternativ eine zeitliche Verlagerung im Hinblick auf die Adaptionsziele als sinnvoll erkannt, so wird für diese ebenfalls ein Koordinationsprozess durchlaufen, der wie folgt charakterisiert ist:

- Task initiiert: Ist identisch zur Initiierung des Tasks in der Verlagerung des Ausführungsortes.

- Ermittlung der passenden Ausführungszeit: Ist eine Ausführung des Tasks aktuell nicht möglich oder im Hinblick auf die Adaptionsziele nicht förderlich, wird die Ausführbarkeit innerhalb zukünftiger Zeiträume untersucht. Innerhalb welches Zeitraums diese Verzögerung möglich ist, ist hierbei als Parameter in Form eines Multiplikators der durchschnittlichen Tasklaufzeit konfigurierbar. Hierzu werden die folgenden Prognosen durchgeführt:

  - Ausführungsprognose: Es wird vorhergesagt, ob auf Basis verfügbarer Kontextinformationen und der verbleibenden Energie des mobilen Geräts ein erfolgreicher Abschluss des Tasks im entsprechenden Zeitintervall zu erwarten ist.
  - Gegebenenfalls werden im Hinblick auf das jeweilige Adaptionsziel weitere Prognosen durchgeführt. Beispielsweise werden für das Adaptionsziel der Einsparung von Bandbreite die vom Task benötigte Bandbreite und die in den Zeitintervallen verfügbare Bandbreite vorhergesagt.

  Wird eine zeitliche Verlagerung als sinnvoll im Hinblick auf die Ausführungsauswahrscheinlichkeit erkannt, wird der entsprechende Task zur Ausführung in diesem Zeitraum vorgesehen.

- Ausführung abgeschlossen: Sobald der Task abgeschlossen ist, werden noch blockierte Ressourcen freigegeben.

Fidelity-Adaptation Neben den zwei zuvor vorgestellten Adaptionsformen kann zusätzlich oder ergänzend eine Anpassung der Implementierung vorgenommen werden. Hierzu wird ein entsprechender Koordinationsprozess durchlaufen, der wie folgt charakterisiert ist:

- Task initiiert: Ist identisch zur Initiierung des Tasks in der Verlagerung des Ausführungsortes.
- Ermittlung der passenden Implementierung: Ist eine Ausführung des Tasks mit der ursprünglichen Implementierung aktuell nicht möglich oder im Hinblick auf die Adaptionsziele nicht sinnvoll, wird die Ausführbarkeit alternativer vorhandener Implementierungen untersucht. Hierzu werden die folgenden Prognosen durchgeführt:
  - Ausführungsprognose: Es wird vorhergesagt, ob unter Berücksichtigung verfügbarer Kontextinformationen und der verbleibenden Energie des mobilen Geräts eine erfolgreiche Ausführung der jeweiligen Implementierung zu erwarten ist.
  - Gegebenenfalls werden im Hinblick auf das jeweilige Adaptionsziel weitere Prognosen durchgeführt. Beispielsweise wird für das Adaptionsziel der Optimierung der Ausführungszeit die erwartete CPU-Zeit der jeweiligen Implementierung vorhergesagt.

  Wird eine Fidelity-Adaptation als sinnvoll im Hinblick auf die Ausführungsauswahrscheinlichkeit erkannt, wird die entsprechende Implementierung zur Ausführung des Tasks genutzt.
- Ausführung abgeschlossen: Sobald der Task abgeschlossen ist, werden noch blockierte Ressourcen freigegeben.

Koordination der Tasks Um die zuvor beschriebenen Adaptionsformen durchzuführen, durchläuft ein Task verschiedene Zustände. In welcher Kombination und Priorisierung die verschiedenen Adaptionsformen eingesetzt werden, wird im Zusammenhang mit der prototypischen Implementierung der für die Koordination der Tasks verantwortlichen Komponente in Abschnitt 5.4.4.5 erläutert.

Die taskbasierte Umsetzung der Adaptionsformen ermöglicht es, den schnell wechselnden Kontext angemessen zu berücksichtigen und so die Ausführungswahrscheinlichkeit zu maximieren, indem sie es erlaubt, kontextabhängig die Adaptionsstrategien einzelner Tasks anzupassen. Hierdurch soll erreicht werden, dass ein Task in Bezug auf den Ausführungsort, die Ausführungszeit und die genutzte Implementierung möglichst sinnvoll im Hinblick auf ein definiertes Adaptionsziel ausgeführt wird. Wie die kompositionelle Adaption hierzu beiträgt und wie die entsprechenden Komponenten hierzu aussehen, wird im nun folgenden Abschnitt beschrieben.

5.4.3. Komponentenmodell

In Abschnitt 5.2 wurde herausgearbeitet, warum Komponenten nicht nur im Hinblick auf die Auslagerung von Berechnungen, sondern auch im Hinblick auf die weiteren zu unterstützenden Adaptionsziele eine sinnvolle Granularität darstellen. Ebenso wurde in Abschnitt 5.3.4 herausgearbeitet, dass die taskbasierte Adaption eine sinnvolle Ergänzung dieser Partitionierung darstellt, um Anwendungen für das mobile Cloud Computing kontextabhängig zu partitionieren.

Hierbei ist wesentlich, dass Komponenten es erlauben, das Expertenwissen der Entwickler im Rahmen der Entwicklung von mobilen Anwendungen zu berücksichtigen [PRK09], und dass die Ermittlung der Auslagerungsentscheidung auf Basis dieser Granularität vergleichsweise schnell möglich ist. Entsprechend kann der Rechenaufwand zur Ermittlung einer sinnvollen Partitionierung einer mobilen Anwendung deutlich reduziert werden. Gleichzeitig erlauben Komponenten es, ein ursprünglich für die sequenzielle Verarbeitung ausgelegtes Programm verhältnismäßig einfach in eine Variante mit Parallelverarbeitung zu überführen [PRK09].

Ebenso wurde von Giurgiu et al. [GRA12] und in *μCloud* [MGL+11] gezeigt, dass die lose Kopplung von Komponenten eine wichtige Eigenschaft ist, um in Szenarien mit wechselnder Konnektivität eine möglichst hohe Unabhängigkeit der einzelnen Komponenten voneinander zu erreichen. Damit einhergehend können entsprechend auch geringere Synchronisationsaufwände im Hinblick auf Häufigkeit und Größe, der zwischen den einzelnen Komponenten einer mobilen Anwendung ausgetauschten Nachrichten erwartet werden.

Im Idealfall ist ein Entwickler somit nur dazu aufgefordert, die Geschäftslogik seiner Anwendung zu entwickeln und sie im Rahmen der bekannten Entwicklungsprinzipien für komponentenbasierte Softwareentwicklung entsprechend auf Komponenten aufzuteilen. Die Komponenten der in dieser Arbeit vorgestellten kontextadaptiven Anwendungsarchitektur, insbesondere der Koordinations-Komponente, sind dafür verantwortlich, die einzelnen Komponenten und entsprechenden Dienstaufrufe auf das mobile Gerät und die Infrastruktur zu verteilen, um die Adaptionsziele möglichst gut umzusetzen. Hierzu ist es allerdings erforderlich, dass sich der Entwickler darauf konzentriert, die Abhängigkeiten zwischen den verschiedenen Komponenten möglichst gering zu halten. Hierfür wurden mit Blick auf die entwickelten Beispielanwendungen drei verschiedene Komponententypen identifiziert, die im Folgenden vorgestellt werden:

Stateless Component Die zustandslose Komponente stellt den vorzugsweise zu verwendenden Typ einer Komponente dar, da sie ihren internen Zustand nicht verändert, wenn einer ihrer Dienste aufgerufen wird. Diese Idempotenz der Dienstaufrufe auf einer Komponente erlaubt es, mehrere Instanzen auf mehreren Surrogates vorzuhalten und im Rahmen der taskbasierten Adaption zu entscheiden, zu welcher Instanz der Komponente ein Aufruf umgeleitet

werden soll. Dies ermöglicht es, die aufgrund der wechselnden Konnektivität häufigen Verbindungsabbrüche zu kompensieren und die Aufrufe zu wiederholen oder sie auf andere Instanzen derselben Komponente umzuleiten.

Eine zustandslose Komponente sollte entsprechend immer dann genutzt werden, wenn der Ausführungszustand einer mobilen Anwendung in einem Fehlerfall dadurch wiederhergestellt werden kann, dass die aktuellen Dienstaufrufe auf andere Instanzen einer Komponente selben Typs umgeleitet werden können. Zustandslose Komponenten sind somit vorzugsweise zu verwenden, da sie die höchsten Freiheitsgrade in Bezug auf die Erreichung der Adaptionsziele bieten.

Stateful Component Zustandsbehaftete Komponenten stellen eine Erweiterung zustandsloser Komponenten dar und unterscheiden sich im Wesentlichen dadurch, dass die aufrufende Anwendung davon ausgehen kann, dass der interne Zustand einer Komponente dieser Art über einen einzelnen Aufruf hinweg aufrechterhalten wird. Entsprechend werden Tasks einer mobilen Anwendung stets auf dieselbe Instanz einer solchen Komponente weitergeleitet. Sie finden Verwendung, wenn ein bestimmter Zustand über einen Dienstaufruf hinaus aufrechterhalten werden soll, und entsprechend existiert für jede Instanz einer Anwendung für das mobile Cloud Computing nur ein Exemplar einer solchen Komponente. Dies hat zur Folge, dass im Fall eines Verbindungsabbruchs zu einer zustandsbehafteten Komponente die koordinierende Komponente versucht, die Verbindung wiederherzustellen, und entsprechend die zu adaptierenden Tasks pausieren. Von der Koordinations-Komponente werden zustandsbehaftete Komponenten nur auf Surrogates instanziiert, die als zuverlässig erreichbar angesehen werden und bei denen davon ausgegangen werden kann, dass sie im Falle eines Verbindungsabbruchs sofort wieder erreichbar sind, sobald das mobile Gerät selbst wieder eine Verbindung zum Internet hat. Dabei handelt es sich um eine Entscheidung, die auf Basis der zum jeweiligen Surrogate verfügbaren Kontextinformationen getroffen wird. Die Stateful Component kann durch dieses Verhalten dazu genutzt werden, Teile des Ausführungszustands einer mobilen Anwendung dauerhaft innerhalb der verwendeten Infrastruktur vorzuhalten.

Data Access Object Die dritte Art von Komponente stellt das Data Access Object (DAO) dar, welches einen effizienten Zugriff auf die gerätespezifischen Funktionen und lokalen Ressourcen eines mobilen Gerätes bietet, zum Beispiel auf die Sensoren oder den internen Speicher des Gerätes. Hierdurch ist es möglich, dass die zustandslosen und zustandsbehafteten Komponenten, die sich nicht auf dem mobilen Gerät befinden, mithilfe von Dienstaufrufen einen Zugriff auf die gerätespezifischen Ressourcen dieses Gerätes erhalten.

Verteilungsmuster und Eigenschaften der Komponenten Nachdem die verschiedenen Komponententypen vorgestellt wurden, sollen im Folgenden nun

weitergehende Regeln vorgestellt werden. Diese Regeln definieren, anhand welcher Bedingungen eine taskbasierte Adaption stattfindet:

- Taskbasierte Adaptionsprozesse gehen stets von einer primären Instanz aus, die den Task initiiert und die in diesem Fall durch das mobile Gerät übernommen wird. Nehmen diese Adaptionsprozesse zur Ausführung eines Tasks weitere Surrogates zu Hilfe, so gehen im Falle eines Verbindungsabbruchs zwischen primärer Instanz und beteiligten Surrogates notwendige Schritte zur erneuten Ausführung des Tasks von dieser primären Instanz aus. Dies dient dazu, den Koordinationsverlust zu vermeiden, der ansonsten eintreten würde und insbesondere bei der Nutzung zustandsbehafteter Komponenten zu einem unerwünschten Verhalten der mobilen Anwendung führen kann.
- Der Zugriff auf geräte- oder systemspezifische Ressourcen darf innerhalb kontextadaptiver Tasks nur über DAO-Komponenten erfolgen, die diese Ressourcen über technologieneutrale Schnittstellen zugänglich machen.
- Aufrufe eines Tasks, die nicht von der primären Instanz und damit von einem Surrogate ausgehen und Dienste anderer Komponenten zum Ziel haben, sollten nach Möglichkeit nur dann auf eine Komponente eines weiteren Surrogates adressiert werden, wenn die Konnektivität zwischen diesen beiden Surrogates als so stabil anzunehmen ist. Dies hat zum Ziel, dass der Dienstaufruf inklusive Rückführung der Ergebnisse erfolgreich durchgeführt werden kann, andernfalls sollte der Dienstaufruf lokal auf dem ersten Surrogate ausgeführt werden. Dies dient dazu, den Abhängigkeitsgraphen eines einzelnen kontextadaptiven Tasks in Situationen mit wechselnder Konnektivität zwischen den beteiligten Geräten auf möglichst wenig beteiligte Geräte zu reduzieren und somit die Ausführungswahrscheinlichkeit des Tasks zu erhöhen. Ein Fall, der allerdings durch die genutzte Partitionierungsstrategie als selten anzunehmen ist und nur dann auftritt, wenn die Ressourcen eines Surrogates nicht ausreichen. Ziel dieser Restriktion ist es, die Robustheit einer mobilen Anwendung zu erhöhen.
- Zustandslose Komponenten können nur Dienste anderer zustandsloser Komponenten aufrufen. Hierdurch wird sichergestellt, dass die primäre Instanz einen Aufruf einer zustandsbehafteten Komponente erkennt und bei einem erfolglosen Aufruf entsprechende kompensierende Aktionen einleiten kann, um ein unerwünschtes Verhalten der Anwendung zu vermeiden.
- Tasks werden nicht immer in der Reihenfolge abgeschlossen, wie dies bei einer lokalen, nebenläufiger Ausführung zu erwarten ist. Dies ist durch die Entwickler entsprechender Anwendungen zu berücksichtigen.

Wie die einzelnen Komponenten zusammen innerhalb einer entsprechenden Ausführungsumgebung ausgeführt werden und welche zusätzlichen Kompo-

nenten die Ausführung dieser Komponenten koordinieren, soll im folgenden Abschnitt beschrieben werden.

5.4.4. Komponenten einer Systemunterstützung

Zur Ausführung von Komponenten der zuvor vorgestellten Typen kommt üblicherweise eine entsprechende Laufzeitumgebung zum Einsatz. Zusammen mit den zur Unterstützung der kontextadaptiven Anwendungsarchitektur erforderlichen Infrastrukturkomponenten, die im Zusammenhang mit der Basisarchitektur ermittelt wurden, werden die Laufzeitumgebung und diese Komponenten gesamthaft als Systemunterstützung bezeichnet. Die Interaktion der einzelnen Komponenten untereinander werden in Abbildung 5.7 veranschaulicht und im Folgenden jeweils näher vorgestellt.

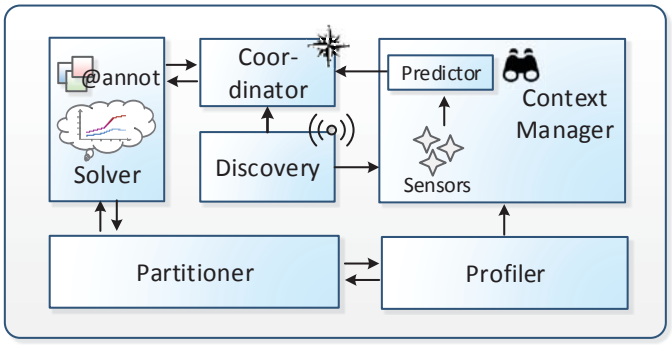

Abbildung 5.7.: Erweiterung der Basisarchitektur für mobiles Cloud Computing

5.4.4.1. Profiling-Komponente

Um das vorgestellte Konzept der taskbasierten Adaption zu unterstützen, bei der die Adaptionsregeln aus der Statistik vergangener Taskausführungen gelernt werden, sollen von der Profiling-Komponente eine Reihe von Daten zur Taskausführung gesammelt werden. Hierzu ist es notwendig, alle Kontextinformationen, die direkt einer Taskausführung zugeordnet werden können, zu erfassen, um sie für zukünftige Ausführungen eines gleichen oder ähnlichen Tasks im Hinblick auf dasselbe oder alternative Adaptionsziel bestmöglich bewerten zu können. Entsprechend steht der Profiler in Abhängigkeit zum Koordinator, von dem er weitere Informationen, beispielsweise das aktuell gewählte Adaptionsziel, übermittelt bekommt, um diese Informationen nach erfolgreichem Abschluss oder endgültigem Abbruch des Tasks an den Kontextmanager zu übergeben, der diese Information zusammen mit weiteren Kontextinformationen speichert und auswertet, um sie dem Koordinator für zukünftige Adaptionsentscheidungen zur Verfügung zu stellen, wie in Abbildung 5.8 veranschaulicht wird. Die Funktionsweise der verschiedenen Profiler in Bezug auf die jeweiligen zu erhebenden Kontextinformationen soll dazu im Folgenden kurz vorgestellt werden.

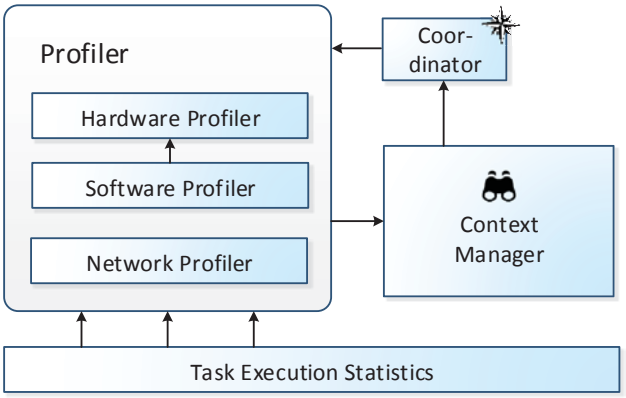

Abbildung 5.8.: Detaillierung der Profiling-Komponente

Hardware-Profiler Zur Erfassung allgemeiner Informationen hinsichtlich der aktuellen Auslastung der CPU und des Speichers des mobilen Gerätes nutzt die Profiling-Komponente im Wesentlichen existierende Schnittstellen des Betriebssystems des mobilen Gerätes.

Software-Profiler Der Software-Profiler überwacht die Warteschlange der für die Ausführung vorgesehenen Tasks und initiiert für die Übergabeparameter eine Ermittlung der Objektgrößen, indem er das übergebene Objekt auf seine Größe hin untersucht und Referenzen verfolgt, um diese ebenfalls auf ihre Größe hin zu untersuchen. Relevant im Hinblick auf den Overhead der Profiling-Komponente ist diese Methode lediglich dann, wenn größere und stark verschachtelte Objektstrukturen als Parameter übergeben werden.

In Bezug auf die Messung des Energieverbrauchs der jeweiligen Taskausführung wird auf die in Abschnitt 5.3.3 vorgestellte Variante der Abschätzung des Energiebedarfs mithilfe eines Energie-Profils zurückgegriffen, um eine möglichst feingranulare und verursachungsgerechte Zuordnung der verbrauchten Energie zu ermöglichen. Zusätzlich, oder sofern kein Energieprofil vorhanden ist, erfolgt zur Plausibilisierung eine Abschätzung des Energieverbrauchs auf Basis der Sende- und Empfangszeiten der jeweils genutzten Übertragungsmedien und der zur Ausführung benötigten CPU-Zeit.

Netzwerk-Profiler Der Netzwerk-Profiler überwacht den gesamten Verkehr direkt an den Netzwerkschnittstellen, um hierdurch sowohl die übertragene Datenmenge als auch die jeweils realisierte Sende- und Empfangsgeschwindigkeit auf dem jeweiligen Übertragungsmedium zu ermitteln. Die Messung erfolgt dabei standardmäßig passiv und wird lediglich vor Beginn einer Taskausführung um eine aktive Messung ergänzt, wie in Abschnitt 5.3.3 beschrieben wurde.

Zusammenfassung Die Profiling-Komponenten messen so alle direkt einem Task zuzuordnenden Informationen. Mit Blick auf die in Abschnitt 5.2.3 vorge-

stellten Adaptionsziele wird hierzu die in Tabelle 5.4 gezeigte Menge an Kontextattributen als besonders relevant angesehen und wird entsprechend im Zusammenhang mit der Entwicklung des generischen Prozesses zur Kontextadaption 5.5 erneut aufgegriffen. Hinsichtlich des hier enthaltenen Kontextattributs *TargetPartition* ist zusätzlich anzumerken, dass dieses erst nach der Partitionierung der mobilen Anwendung vorliegt, die im folgenden Abschnitt 5.4.4.2 beschrieben wird.

Neben der taskbasierten Ermittlung werden Gesamtwerte zur Unterstützung der Entwickler sowie eine Reihe übergreifender Statistiken ermittelt, die jedoch nicht für die eigentliche Adaption der Taskausführung relevant sind und entsprechend nur periodisch ermittelt werden. Diese werden mithilfe der an den Kontextmanager übergebenen Informationen generiert, um eine übergreifende Erfolgsbetrachtung durchführen zu können, und sind in Tabelle 5.5 zusammengefasst. Sollte es durch eine sehr hohe Frequenz ausgeführter Tasks in einem kurzen Zeitraum dazu kommen, dass das Profiling einen zu hohen Ressourcenverbrauch erzeugt, werden die besonders rechenintensiven Methoden zur Ermittlung der Objektgrößen deaktiviert.

5.4.4.2. Partitionierungs-Komponente

Gilt es für einen bestimmten Task zu ermitteln, ob eine Adaption entlang der Dimension des Ausführungsortes sinnvoll ist, so ist zu erkennen, ob ein bestimmter Teil dieser mobilen Anwendung in die Infrastruktur verlagert werden kann. Unter Berücksichtigung dieser Information wird von der Koordinations-Komponente ermittelt, ob ein bestimmter Task ganz oder teilweise in der Infrastruktur ausgeführt werden soll.

Das Ziel der Partitionierung ergibt sich dabei aus dem jeweiligen Ziel der Adaption. Hierzu wird der MIN-CUT-Algorithmus auf den Abhängigkeitsgraphen eines typischen Nutzungsmusters der mobilen Anwendung angewendet, wobei die jeweiligen Gewichte der Knoten und Kanten des Graphen in Abhängigkeit des jeweiligen Zieles der Adaption auszugestalten sind, wie es in Abschnitt 5.3.4 erläutert wurde und in Abbildung 5.9 veranschaulicht ist.

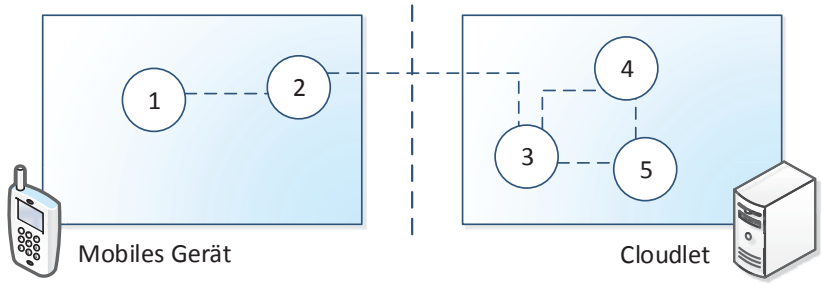

Abbildung 5.9.: Partitionierung einer mobilen Anwendung

Obwohl der in Abschnitt 5.3.4 identifizierte Lösungsansatz eine binären Partitionierung liefert, ist die Interaktion einer mobilen Anwendung mit der Infrastruktur keineswegs auf ein Surrogate beschränkt. Da die Tasks einer mobilen

Attribut	Beschreibung
ScenarioID	Eindeutiger Identifier des simulierten Szenarios.
CallID	Eindeutiger Identifier eines Tasks.
InvocationTime	Die bis zum aktuellen Status vergangene Ausführungszeit.
TimeDate	Der Startzeitpunkt der Ausführung.
UsageProfile	Das gewählte Adaptionsziel.
CallingApplication	Die aufrufende Anwendung.
ViewID	Die Einheit der Anwendung, die den Task initiiert hat.
CallingService	Die aufrufende Komponente oder Anwendung.
TargetPlatform	Das mobile Gerät oder Surrogate, an das der Aufruf gerichtet war.
TargetPartition	Das Ergebnis der Partitionierung der mobilen Anwendung
CalledComponent	Der Typ der Komponente, an die der Aufruf gerichtet war.
CalledService	Der Dienst der Komponente, der aufgerufen wurde.
CallSignature	Die Signatur des Aufrufs.
InputSize	Die Größe der in Senderichtung übergebenen Parameter.
SendTime	Die Zeit, die für das Senden eines Tasks zu einem Surrogate benötigt wurde.
PerceivedSendBW	Die durchschnittliche Bandbreite, mit der gesendet wurde.
LocPredExecTime	Die Zeit, die vom Kontextmanager für die lokale Ausführung auf Basis vorhandener Ausführungsstatistiken geschätzt wurde.
RemPredExecTime	Die Zeit, die vom Kontextmanager für die Ausführung auf dem jeweiligen Surrogate auf Basis vorhandener Ausführungsstatistiken geschätzt wurde.
RemPredInvTime	Die gesamte Zeit, die für das Senden, die Ausführung und die Rückführung des Ergebnisses benötigt wird und die vom Koordinator zusammen mit dem Kontextmanager auf Basis der aktuell geschätzten Bandbreite ermittelt wird.
ExecutionTime	Die tatsächliche Ausführungszeit des Tasks.
ExecutionCPUTime	Die für die Ausführung gemessene CPU-Zeit des Tasks.
EnergyConsumed	Die für das Senden, die Ausführung, den Empfang des Ergebnisses und die Koordination der Adaption aufgewendete Energie. Bei der lokalen Ausführung wird die Übertragung als energieneutral angesehen.
ReceiveTime	Die Zeit, die für das Empfangen eines Tasks von einem Surrogate benötigt wurde.
ReturnSize	Die Größe der in Empfangsrichtung übergebenen Parameter.
PerceivedRecBW	Die durchschnittliche Bandbreite, mit der empfangen wurde.
ExitCode	Der letzte Status des Tasks. Für abgeschlossene Tasks ist dieser DONE (22).
LocalOverheadTime	Die Zeitspanne, die der Koordinator benötigt hat, um in Bezug auf die Adaptionsziele die richtige Ausführungsstrategie zu ermitteln.
RoundTripTime	Die Paketlaufzeit auf der genutzten Verbindung zum Surrogate.
TotalInvocationTime	Die gesamte Ausführungszeit des Tasks.

Tabelle 5.4.: Vom Profiler erfasste Parameter der Taskausführung

Attribut	Beschreibung
ScenarioID	Eindeutiger Identifier des simulierten Szenarios.
UserID	Eindeutiger Identifier des jeweiligen Nutzers.
UsageProfile	Das gewählte Adaptionsziel.
ActiveTasks	Fehlerhaft beendete Tasks.
CompletedTasks	Anzahl abgeschlossener Tasks.
UnfinishedTasks	Anzahl abgebrochener Tasks.
TotalTasks	Anzahl aller Tasks.
Mobile-Tasks	… davon auf dem mobilen Gerät ausgeführt.
Cloud-Tasks	… davon auf den Cloud-Servern ausgeführt.
Cloudlet-Tasks	… davon auf den Cloudlets ausgeführt.
EnergyConsumed	… hierzu benötigte Energie des mob. Gerätes
LocEnergyConsumed	Zur lokalen Ausführung benötigte Energie.
Sended4G	Gesendete Datenmenge WWAN
SendedWiFi	Gesendete Datenmenge WLAN
SendedTotal	Gesendete Datenmenge insgesamt
SendTasksizeTotal	Für alle Tasks zu sendende Datenmenge.
Received4G	Empfangene Datenmenge WWAN
ReceivedWiFi	Empfangene Datenmenge WLAN
ReceivedTotal	Empfangene Datenmenge insgesamt
RecTasksizeTotal	Für alle Tasks zu empfangende Datenmenge
InvTimeMobile	Durchschn. Ausführungszeit auf dem mobilen Gerät.
InvTimeCloud	Durchschn. Ausführungszeit auf Cloud-Servern.
InvTimeCloudlet	Durchschn. Ausführungszeit auf Cloudlets.
InvTime	Durchschn. Ausführungszeit insgesamt.
InvTimeSuccessful	Durchschn. Ausführungszeit abgeschlossener Tasks.
SuccessRate	Anteil abgeschlossener Tasks.

Tabelle 5.5.: Statistik-Informationen des Netzwerk-Profilers

Anwendung als unabhängig voneinander betrachtet werden, können diese zeitgleich auf die unterschiedlichen verfügbaren Surrogates verteilt werden. Dies ermöglicht es, die Partitionierung zur Entwurfszeit der Anwendung durchzuführen und mithilfe der taskbasierten Adaption gleichzeitig den aktuellen Ausführungskontext angemessen zu berücksichtigen. Am Beispiel einer sinnvollen Partitionierung einer Anwendung zur Bildverarbeitung mit dem Optimierungsziel der Zeiteinsparung würde sich in diesem Fall typischerweise eine Partitionierung ergeben, bei der in der Ausführung nahe beieinanderliegende

Teile der Geschäftslogik in eine Partition gruppiert werden, sodass eine Adaption in Form einer Auslagerung dann stattfindet, wenn sowohl der zu übertragende Zustand als auch die zu erwartende Interaktion zwischen dem auf dem mobilen Gerät verbliebenen Teil der mobilen Anwendung möglichst gering sind, um den schnell wechselnden Kontext innerhalb des mobilen Cloud Computings angemessen zu berücksichtigen.

5.4.4.3. Discovery-Komponente

Im Hinblick auf die Interaktion innerhalb eines mobilen Ad-hoc-Netzwerks, gilt es zunächst, die grundlegende Fähigkeit zur Kommunikation und Dienstsuche zu implementieren. In diesem Zusammenhang wurde in Abschnitt 5.3.2 als Lösungsansatz für die zu realisierende Anwendungsarchitektur festgehalten, dass die Kommunikation auf Basis eines Overlay-Netzwerks zu implementieren ist – eine Funktionalität, die von einer Reihe verteilter Middleware-Lösungen wie [BP11, VNMW+05] bereits gut unterstützt wird. Hierdurch sollen vor allem die Effekte der wechselnden Konnektivität unterhalb der Anwendungsebene – und für diese transparent – kompensiert werden, wie in Abbildung 5.10 veranschaulicht wird.

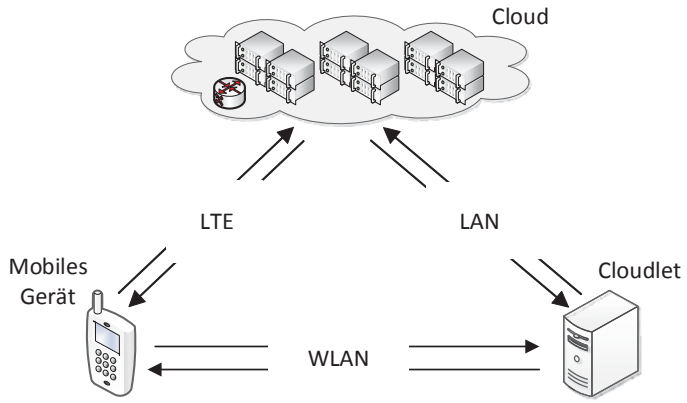

Abbildung 5.10.: Dienstsuche mithilfe der unterschiedlichen Schnittstellen

Hierdurch stellt die Basisarchitektur eine Ende-zu-Ende-Verbindung zwischen den an der kooperativen Ausführung beteiligten Geräten bereit, die unabhängig von der internen Architektur der verfügbaren Netzwerke wie dem Internet oder einem Ad-hoc-Netzwerk und Standards zur Mobilkommunikation wie LTE oder Bluetooth bereit ist, die direkte Kommunikation zwischen mobilen Geräten und Surrogates zu ermöglichen.

Inwieweit existierende Lösungen die existierenden Schnittstellen zur Mobilkommunikation unterstützen oder ob hier die Entwicklung zusätzlicher Adapter notwendig ist, soll im Rahmen der Implementierung im weiteren Verlauf der Arbeit noch näher untersucht werden. Ebenso soll an dieser Stelle untersucht werden, ob existierende Mechanismen zur Fehlerbehandlung genutzt werden können, die der hohen Mobilität der beteiligten Geräte gerecht werden.

Auf Basis der hierdurch etablierten Kommunikation eines mobilen Gerätes zur Infrastruktur kann anschließend die Suche nach benötigten Diensten erfolgen.

5.4.4.4. Kontextmanager

Eine weitere Komponente der zu entwickelnden kontextadaptiven Anwendungsarchitektur stellt der Kontextmanager dar, dessen Aufgabe es ist, die Adaptionsentscheidung des Koordinators, der nachfolgend beschrieben wird, zu unterstützen (vergleiche Abbildung 5.11).

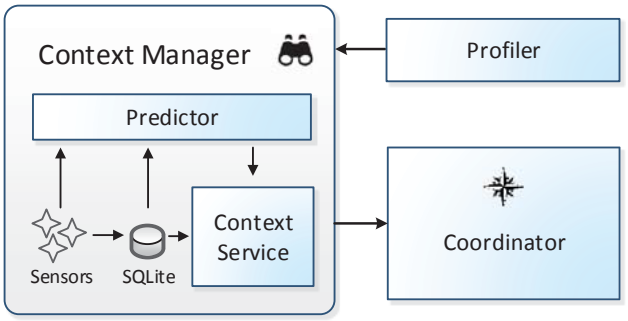

Abbildung 5.11.: Einbettung des Kontextmanagers

Hierzu stellt der Kontextmanager, zusammen mit dem Koordinator, eine generische Adaptionslogik bereit, die es dem Entwickler ermöglicht, die Adaption einer mobilen Anwendung auszulösen, bei der lediglich das Ziel der Adaption vorzugeben, aber keine weitergehende Adaptionslogik zu implementieren ist. In diesem Zusammenhang werden vom Kontextmanager die folgenden vier Aufgaben wahrgenommen:

- Die Speicherung von Ausführungsstatistiken des Koordinators.
- Die Sammlung und Speicherung von Kontextdaten, die nicht in direktem Zusammenhang mit der Taskausführung stehen, die aber über die Sensoren des mobilen Gerätes erfasst werden.
- Die Bereitstellung dieser Kontextinformationen an die übrigen Komponenten und Infrastrukturkomponenten einer kontextadaptiven Anwendungsarchitektur.
- Die Zusammenführung der gesammelten Kontextinformationen zur Generierung höherwertiger Kontextattribute und zur Ermittlung zukünftiger Ausprägungen einzelner Kontextattribute im Rahmen von Prognosen eines zukünftigen Kontextes des mobilen Gerätes. Dies schließt auch anwendungsspezifische Kontextattribute mit ein.

Die ersten zwei Aufgaben lassen sich unter dem Aspekt der Sammlung von Kontextdaten zusammenfassen, die letzten zwei unter dem Aspekt der Bereitstellung von Kontextdaten. Beide Aspekte sollen im Folgenden jeweils kurz anhand ihrer Schnittstelle charakterisiert werden, bevor im weiteren Verlauf der Arbeit die konkrete Funktionsweise näher vorgestellt wird.

Sammlung der Kontextdaten In Bezug auf die Aufgabe der Sammlung und Speicherung der Kontextdaten, die der Kontextmanager mithilfe der verschiedenen Sensoren des mobilen Gerätes wahrnimmt, wird bewusst keine feste Menge an Kontextattributen definiert. Hierdurch soll der hohen Heterogenität der zu unterstützenden mobilen Geräte Rechnung getragen werden. Im Hinblick auf die Unterstützung der definierten Ziele der Adaption orientieren sich diese Kontextattribute an der folgenden Schnittmenge:

- Datum, aktuelles Zeitintervall, Ort.
- Ladezustand des mobilen Gerätes, verbleibender Energievorrat.
- Status der einzelnen Schnittstellen zur Mobilkommunikation wie Signalstärken, erzielte Bandbreite und in den verfügbaren Netzwerken erkannte Geräte.
- Höherwertige Kontextattribute: geschätzte Verweildauer am selben Ort und im selben Netzwerk (abgeleitet aus der Historie), erkannte häufig besuchte Orte und die aktuelle Aktivität des Nutzers (laufend, stehend, sitzend).
- Aus der Ausführungsstatistik der Profiling-Komponente gewonnene Informationen bezüglich aktueller Taskausführungen und die hierfür benötigte Zeit, Energie und Bandbreite.

Die Ermittlung dieser Kontextattribute findet dabei intervallbasiert statt und wird abhängig vom genutzten mobilen Gerät um weitere Sensorinformationen wie beispielsweise den Kalendereinträgen aus der mobilen Anwendung zur Terminverwaltung ergänzt, wie sie häufig auf mobilen Geräten wie Smartphones zu finden sind. Wie die entsprechenden Kontextinformationen durch den Kontextmanager akquiriert, welche Datenquellen und Informationen konkret ausgewählt und wie diese verarbeitet werden, wird im Zusammenhang mit der generischen Kontextadaption in Abschnitt 5.5 vorgestellt.

Bereitstellung von Kontextdaten Das Ziel der beschriebenen Sammlung von Kontextdaten ist es, diese über eine Schnittstelle bereitzustellen und sie im Rahmen der Weiterverarbeitung und Generierung höherwertiger Kontextdaten zusätzlich anzureichern. Im Hinblick auf die geforderte Proaktivität der Adaptionsentscheidungen soll hierzu mithilfe der auf dem jeweiligen mobilen Gerät verfügbaren Kontextdaten und der individuell dort vorgehaltenen Historie dieser Kontextdaten eine Prognose einzelner Kontextattribute bereitgestellt werden. Im Hinblick auf die zu unterstützenden Ziele der Adaption und der in diesem Zusammenhang beschriebenen Adaptionsformen wurden die folgenden Kontextattribute als relevant im Rahmen einer Prognose identifiziert:

- Energievorrat des mobilen Geräts
- Erfolg der Ausführung eines Tasks

- Ausführungszeit eines Tasks
- Bandbreite zu einem Surrogate
- Konnektivität zu einem Surrogate
- Weitere durch den Entwickler definierte Kontextattribute (Beispielsweise anwendungsbezogene Informationen wie die genutzten Daten einer mobilen Anwendung, um diese proaktiv an einem WLAN-Zugangspunkt von der Infrastruktur auf das mobile Gerät zu übertragen.)

Diese Kontextattribute sollen entsprechend vom Kontextmanager für ein bestimmtes Zeitintervall vorhergesagt werden können. Entsprechend soll hierzu im Folgenden Abschnitt ein generischer Prognoseprozess entwickelt werden, der mithilfe der auf dem jeweiligen mobilen Gerät verfügbaren Kontextdaten zunächst die Relevanz der verfügbaren Kontextdaten im Sinne relevanter Eingabedaten im Zusammenhang mit der Prognose eines Kontextattributs bewertet und hieraus ein individuelles Prognosemodell generiert. Dieses Vorgehen soll es ermöglichen, auch in Situationen mit eingeschränkter Datenqualität wie fehlenden Werten einer kurzen Historie oder nur wenigen verfügbaren Kontextattributen eine ausreichend genaue Abschätzung einer zukünftigen Ausprägung eines Kontextattributs zu ermitteln.

Analog zur Dienstsuche arbeitet der Kontextmanager hinsichtlich der Bereitstellung von Kontextdaten aus Gründen der Effizienz rein reaktiv, was die Generierung der Prognosen betrifft. Wird beispielsweise vom Koordinator eine neue Prognose für ein bestimmtes Kontextintervall benötigt, so wird diese in Form eines Dienstaufrufs beim Kontextmanager abgefragt. Hierzu gilt es vorab, je nach mobiler Anwendung entsprechende Prognosemodelle zu generieren. Wie viele Modelle mit jeweils unterschiedlichen Prognosehorizonten und Zielgrößen dabei generiert werden müssen, ist im Wesentlichen durch die zu unterstützende mobile Anwendung beeinflusst. So ist die durchschnittliche als auch die maximale Ausführungszeit von Tasks dieser Anwendung ein guter Indikator für die, durch Generierung entsprechender Prognosemodelle, zu unterstützenden Prognosehorizonte.

Hinsichtlich der Sammlung und Speicherung der Kontextdaten arbeitet der Kontextmanager hingegen mit einer periodischen Abfrage der zu sammelnden Kontextdaten. Dies zielt auf die Generierung einer möglichst langen und vollständigen Historie einzelner Kontextattribute ab. Die im Zusammenhang mit der Taskausführung generierten Statistiken werden hierzu ebenfalls an den Kontextmanager übergeben, die Übermittlung erfolgt hier in Abhängigkeit des Ausführungsfortschrittes des jeweiligen Tasks.

5.4.4.5. Koordinator

Die letzte Komponente im Zusammenhang mit der Realisierung einer kontextadaptiven Anwendungsarchitektur stellt der Koordinator dar. Seine Aufgabe ist es, die Taskausführung zu koordinieren und sie in Zusammenarbeit

mit den übrigen Komponenten, insbesondere dem Kontextmanager, an den aktuellen und zukünftigen Kontext eines mobilen Gerätes und auf die unterschiedlichen Ziele der Adaption hin anzupassen.

Zusammen mit dem Kontextmanager implementiert der Koordinator hierzu eine Adaptionslogik, die darauf abzielt, die Taskausführung im Hinblick auf die Ziele der Adaption anzupassen, ohne dass hierzu durch den Entwickler eine zusätzliche Adaptionslogik zu implementieren ist. Seine Funktionsweise soll im Folgenden vorgestellt werden.

Umsetzung der Ziele der Adaption Zur Umsetzung der einzelnen Ziele der Adaption nutzt der Koordinator den komponentenbasierten Entwurf einer mittels des vorgestellten Komponentenmodells entwickelten mobilen Anwendung, um mithilfe der kompositionellen Adaption eine Adaption entlang der verschiedenen Dimensionen der Adaption durchzuführen. Mithilfe der durch den Kontextmanager bereitgestellten Kontextinformationen steuert der Koordinator so die Adaption entlang der Dimensionen der Zeit (Verschiebung der Ausführung) und der Implementierung (Fidelity Adaptation). Durch die zusätzliche Berücksichtigung der Partitionierung erfolgt zusätzlich die Anpassung entlang der Dimension des Ausführungsortes (Auslagerung von Berechnungen).

Die Ermittlung einer kontextadaptiven Ausführungsstrategie entspricht einem Kompromiss zwischen optimalen und effizient ermittelten Adaptionsentscheidungen. Mit dem Ziel, die Ressourcen des mobilen Gerätes nicht zusätzlich zu belasten, sollen einerseits die einzelnen Adaptionsentscheidungen mit möglichst geringem Ressourcenaufwand ermittelt und hierzu eine Priorisierung der unterschiedlichen Adaptionsziele vorgenommen werden, um nur die Alternativen, die im Hinblick auf ein bestimmtes Adaptionsziel relevant sind, näher untersuchen zu müssen. Entsprechend soll als übergeordnetes Ziel die Anzahl der ausgeführten Tasks maximiert werden und gleichzeitig die Taskausführung ermöglicht werden, die ohne eine Kooperation mit der Infrastruktur von einem mobilen Gerät nicht ausgeführt werden können.

Ebenso soll die Dienstverfügbarkeit des mobilen Gerätes erhöht werden, wobei dies im Wesentlichen dadurch erreicht wird, dass der verbleibende Energievorrat des mobilen Gerätes bei der Taskausführung eine möglichst geringe Belastung erfährt. Gleichzeitig sollte hierbei aus Sicht der Nutzer keine zusätzliche Einschränkung in Form einer zu langen Verzögerung der Taskausführung entstehen, welche sich beispielsweise im Hinblick auf das Adaptionsziel der Einsparung von Energie zwangsläufig ergeben würde, wenn sämtliche Tasks verzögert werden, bis das mobile Gerät erneut mit einer externen Energiequelle verbunden ist. In diesem Zusammenhang wird ein durch den Entwickler oder Nutzer definierter Parameter zur Konfiguration des Koordinators erwartet, der, multiplikativ verknüpft mit der jeweiligen durchschnittlichen Ausführungszeit des Dienstaufrufes eines Tasks, die maximale Verzögerung der Taskausführung darstellt und die der Koordinator im Rahmen seiner Adap-tionsentscheidungen als obere Grenze der Ausführungszeit berücksichtigt.

Eine weitere Restriktion im Hinblick auf die Adaptionsentscheidungen des Koordinators ergibt sich aus der Koordination mehrerer konkurrierender Tasks. Um größere, ressourcenlastige Tasks nicht gegenüber mehreren kleinen Tasks zu benachteiligen und um deren „Verhungern“ (starvation) zu vermeiden, wird der Nutzen der abgeschlossenen Taskausführung für alle Tasks als gleich bewertet. Ihre Ablaufplanung (scheduling) priorisiert sich entsprechend nach dem Zeitpunkt der Initiierung, und sie sind zueinander gleichberechtigt. Hiervon ausgenommen ist das Adaptionsziel der Verbesserung der Performance, entsprechende Tasks werden vorrangig vor Tasks mit davon abweichenden Adaptionszielen ausgeführt.

Als übergeordnetes Ziel der Adaption gilt es entsprechend, möglichst viele Tasks ohne Reduktion der Dienstqualität erfolgreich auszuführen und hierbei entweder die Ausführungsgeschwindigkeit, die Energieeffizienz oder die Einsparung von Bandbreite zu maximieren, jeweils unter Berücksichtigung der maximal zugelassenen Verzögerung der Taskausführung. Basierend auf diesem übergeordneten Ziel sind die übrigen, teils mit dem übergeordneten Ziel konkurrierenden Ziele als alternative Adaptionsziele vorgesehen, die in Abbildung 5.12 veranschaulicht werden.

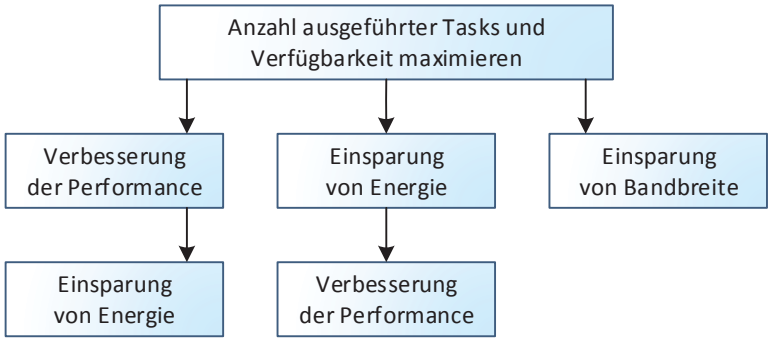

Abbildung 5.12.: Priorisierung der einzelnen Ziele der Adaption

Entsprechend ergeben sich alternative Adaptionsmöglichkeiten, die vom Koordinator anhand des Erfüllungsgrades des vorgegebenen Adaptionsziels abzuschätzen sind. In Bezug auf die Realisierung der Kombination von Adaptionszielen sollen hierzu jeweils nur die in Abschnitt 5.2.3 als am häufigsten anzuwendenden genannten Ziele herangezogen werden, um der geforderten Effizienz der Adaptionsentscheidung Rechnung zu tragen und den hierdurch verursachten Overhead möglichst gering zu halten. Die hieraus resultierenden Varianten sollen im Folgenden vorgestellt werden:

Verbesserung der Performance Eine Verbesserung der Performance kann, wenn keine Reduktion der Dienstqualität vorgenommen werden soll, vor allem mithilfe einer Kooperation mit der Infrastruktur erreicht werden. Die Qualität dieser sogenannten Auslagerungsentscheidung ist im Wesentlichen von zwei Faktoren abhängig: der Abschätzung der erwarteten Ausführungszeit eines Tasks auf dem mobilen Gerät und den zur Verfügung stehenden Surrogates

und den vorliegenden Informationen über die (zukünftige) Konnektivität zu den relevanten Surrogates. Hierzu werden im Wesentlichen die Paketumlaufzeit (RTT) und die zwei Zeitintervalle berücksichtigt, in denen die initiale und die abschließende Synchronisation mit dem jeweiligen Surrogate erfolgt.

$$SyncTime = \frac{InputSize}{UploadBW} + \frac{OutputSize}{DownloadBW} + RTT$$

Zur Beschleunigung der Taskausführung wird entsprechend für alle verfügbaren Surrogates eine Prognose des Erfolges der Ausführung eines Tasks, der Ausführungszeit und der Konnektivität vom Kontextmanager angefordert, um anschließend auf Basis der folgenden Entscheidungsfunktion die Ausführung zu initiieren:

$$LocalExecutionTime < RemoteExecutionTime + SyncTime$$

Ausgehend von mehr als einer Alternative, die eine beschleunigte Ausführung erwarten lässt, wird anschließend die Alternative gewählt, bei der die Auslagerung den geringsten zu erwartenden Energieverbrauch erwarten lässt. Wird während der nun folgenden Ausführung des Tasks erkannt, dass kein erfolgreicher Abschluss der Taskausführung zu erwarten ist, wird diese abgebrochen und gegebenenfalls auf einem weiteren verfügbaren Surrogates gestartet. Sollte aktuell keine Taskausführung möglich sein, wird diese im Rahmen der zulässigen Verzögerung verschoben, bevor die Taskausführung als abgebrochen und gescheitert vermerkt wird.

Einsparung von Energie Eine Einsparung von Energie, ebenfalls ohne Reduktion der Dienstqualität, kann vor dem Hintergrund der zuvor eingeführten übergreifenden Ziele der Taskausführung im Wesentlichen dadurch realisiert werden, dass die Ausführung eines Tasks in die Infrastruktur verlagert wird. Hierbei gilt es, analog zum vorgenannten Adaptionsziel der Verbesserung der Performance, die Auslagerungsentscheidung im Hinblick auf die Einsparung von Energie dahingehend zu bewerten, dass die zur lokalen Ausführung benötigte Energie geringer ist als die Energie, die für die Übertragung und Synchronisation des Tasks in die Infrastruktur benötigt wird:

$$SyncEnergy = SyncTime \cdot f_{TransferEnergy}(TransferMedium)$$

$$LocalExecutionEnergy < SyncEnergy$$

Bei Verfügbarkeit mehrerer Surrogates wird anschließend die Variante ausgewählt, bei der die Verbesserung der Performance der Taskausführung als am höchsten angesehen wird. Im Hinblick auf die Fehlerbehandlung wird analog zu der Variante der Verbesserung der Performance vorgegangen.

Einsparung von Bandbreite Eine Einsparung der Bandbreite lässt sich durch eine Partitionierung der mobilen Anwendung realisieren, bei der die Partition, die für die Ausführung in der Infrastruktur vorgesehen ist, eine höhere zu transferierende Datenmenge in Richtung der übrigen Knoten eines Netzwerks aufweist als die Datenmenge, die es in diesem Fall mit dem mobilen Gerät auszutauschen gilt. Wie im Zusammenhang mit den Adaptionszielen aufgezeigt wurde, kann eine mobile Anwendung zum Durchsuchen und Filtern von Nachrichtenportalen so partitioniert werden, dass die für den Download und das Filtern verantwortliche Komponente der mobilen Anwendung in die Infrastruktur verlagert wird. Hierdurch müssen nicht länger alle Informationen auf das mobile Gerät transferiert werden, sondern lediglich die im Rahmen der Suchkriterien als relevant identifizierten. In diesem Fall kann eine entsprechende Partitionierung der Anwendung herangezogen werden (vergleiche Abschnitt 5.4.4.2), die darauf abzielt, die Interaktion zwischen den zwei enstandenen Partitionen zu minimieren.

In Bezug auf dieses Optimierungsziel wird eine Minimierung der Interaktion als äquivalent im Hinblick auf das vorgelagerte Ziel der Maximierung der Anzahl erfolgreich ausgeführter Tasks angesehen. Entsprechend ist im Anschluss an die Partitionierung in diesem Fall keine Entscheidungsfunktion nötig, da die zwischen den einzelnen Komponenten ausgetauschten Datenmengen, relativ betrachtet, als konstant angesehen werden können. So kann die zuvor erwähnte mobile Anwendung zur Suche und Aggregation verschiedener Informationsdienste (beispielsweise ein Aggregator zum Lesen von Web-Feeds) dahingehend partitioniert werden, dass die bandbreitenintensive Aggregation in die Infrastruktur ausgelagert wird. Auf dem mobilen Gerät verbleibt dabei lediglich die Anzeigefunktion, die die aggregierenden Inhalte durchsucht.

Zusammenfassung Die gezeigten Varianten erlauben es somit auf Basis der Statistik vergangener Taskausführungen, mithilfe aktueller und zukünftiger relevanter Kontextinformationen wie der Bandbreite zu bestimmten Surrogates zu ermitteln, ob und mithilfe welcher Adaptionsstrategie ein bestimmtes Adaptionsziel erreicht werden kann. Dies wiederum erfordert es, dass eine entsprechende Historie an Kontextinformationen und Ausführungsstatistik vorhanden ist. Hierzu können einerseits die Nutzungsdaten anderer Anwender genutzt werden oder das Nutzungsprofil von einem weiteren mobilen Gerät desselben Anwenders genutzt werden, um ein initiales Modell zu generieren. Dieser Vorgang ist allerdings nur dann notwendig, wenn eine mobile Anwendung erstmalig auf einem Gerät ausgeführt (Ausführungsstatistik) oder die Systemunterstützung erstmalig genutzt wird (Kontextinformationen).

Wie die hierzu benötigten Prognosen einzelner Kontextattribute ermittelt werden, soll im nun folgenden Abschnitt im Zusammengang mit der Vorstellung eines generischen Prozesses zur Kontextadaption beschrieben werden.

5.5. Konzeption der generischen Kontextadaption

Im folgenden Abschnitt soll die in Abschnitt 5.4.4.4 vorgestellte Komponente des Kontextmanagers dahingehend erweitert werden, nicht nur historische und aktuelle Ausprägungen von Kontextattributen bereitzustellen, sondern ebenso neue höherwertige Kontextattribute und zukünftige Ausprägungen dieser Kontextattribute im Rahmen einer Prognose ermitteln zu können. Diese Erweiterung zielt darauf ab, proaktive Adaptionsentscheidung des Koordinators der kontextadaptiven Anwendungsarchitektur angemessen zu unterstützen.

Vor dem Hintergrund der Problemstellung, die zukünftigen Ausprägungen unterschiedlicher Kontextattribute aus einer ebenfalls unterschiedlichen Menge an weiteren Kontextattributen vorherzusagen, soll hierzu ein generischer Prozess entworfen werden, der in der Lage ist, automatisiert Prognosemodelle durch die Auswahl unterschiedlicher Lernverfahren zu entwickeln und bereitzustellen. Die Generizität des zu entwerfenden Prognoseprozesses zielt hierbei auf die unterschiedlichen zu unterstützenden Datenquellen und Zielvariablen der Prognose ab.

Hierzu werden zunächst im Rahmen einer Anforderungsanalyse sowohl generelle Anforderungen als auch im Hinblick auf die Adaption wichtige zu prognostizierende Kontextattribute definiert, anhand derer anschließend existierende Lösungsansätze zur Prognose von Kontextattributen auf ihre Übertragbarkeit hin untersucht werden. Um die Nutzbarkeit des zu entwerfenden Ansatzes sicherzustellen, wird anschließend das in Abschnitt 1.5 vorgestellte Vorgehensmodell angewendet, welches einen explorativen und datenanalytischen Ansatz fokussiert, um zunächst eine repräsentative Menge an Kontextdaten zu untersuchen und deren konkrete Modellierung im Anschluss weiter zu detaillieren. Hierzu werden die in Abschnitt 3.2.2 vorgestellten Konzepte der Kontextmodellierung genutzt und anschließend die in Abschnitt 3.2.3 diskutierten Verfahren zur Erkennung von Mustern in Kontextdaten verwendet, um die automatisierte Erzeugung von Prognosemodellen für unterschiedliche Kontextattribute zu ermöglichen.

5.5.1. Anforderungsanalyse

Wie zuvor beschrieben besteht die Anforderung an den Prozess zur generischen Kontextadaption darin, Prognosemodelle bereitzustellen, die aus einer beliebigen Menge von Kontextattributen für ein bestimmtes Attribut eine Prognose über dessen Ausprägung in einem zukünftigen Zeitintervall treffen (vergleiche Kriterien A3 und P3 in den Abschnitten 4.2 und 5.2.3). Hierbei ist zusätzlich zu gewährleisten, dass die entwickelten Prognosemodelle ressourcenschonend sind, um auf einem mobilen Gerät ausgeführt werden zu können und somit eine schnelle Auslagerungsentscheidung zu unterstützen. Im Rahmen der vom Koordinator benötigten Kontextprognosen sollen hierzu die folgenden zwei Ty-

pen von Modellen zur Prognose der jeweiligen *Zielwerte* in Form von Vorhersagen der jeweiligen Kontextattribute bereitgestellt werden:

- ein binärer Klassifikator zur Prognose der:
  - Der Erfolg der Ausführung eines Tasks
  - Konnektivität zu einem Surrogate
  - weiterer Klassifikationsprobleme
- ein Regressionsschätzer zur Prognose der/des:
  - Ausführungszeit eines Tasks
  - Bandbreite zu einem Surrogate
  - Energievorrates des mobilen Gerätes
  - weitere Regressionsprobleme

Die Entwicklung dieser Prognosemodelle erfolgt dabei unter Berücksichtigung der in Abschnitt 4.2 entwickelten Anforderungen, aus denen sich die folgenden nichtfunktionalen Kriterien ergeben:

- Die verwendeten Verfahren für das Reasoning sollten möglichst nicht proprietär sein und eine Implementierung vorweisen, die auf mobilen Geräten ausführbar und mit den dort verwendeten Systemplattformen kompatibel ist.
- Das Reasoning soll aus vergangenen Situationen und Ereignissen lernen und hierzu die Historie der Kontextzustände berücksichtigen.
- Das Reasoning soll möglichst robust sein und sich nicht von falschen, fehlenden oder untypischen Daten stören lassen.
- Das Reasoning soll deterministisch sein, dies bedeutet, es soll auf denselben Trainingsdaten dasselbe Modell ableiten und auf denselben Eingangsdaten dieselbe Prognose erstellen.
- Das Reasoning-Verfahren soll möglichst keine oder wenige Parameter besitzen, die im Rahmen des Trainings des Modells angepasst werden müssen.
- Das Reasoning sollte möglichst sparsam in Bezug auf die Ressourcennutzung sein, insbesondere bei der Anwendung des trainierten Modells.

Die wesentliche Anforderung des zu entwickelnden Prozesses lässt sich entsprechend damit zusammenfassen, dass dieser die jeweils verfügbaren Kontextattribute hinsichtlich ihrer Datenqualität und des Vorhandenseins einer Historie auswählt, um sie im Hinblick auf die Prognose eines jeweiligen einzelnen Kontextattributes zu nutzen. Hierdurch soll sichergestellt werden, dass auch im Fall weniger Kontextattribute oder im Fall schlechter Datenqualität noch Prognosen möglich sind.

Die Erstellung der Prognosemodelle ist hierbei bewusst nicht auf die zuvor aufgeführten Zielwerte beschränkt, sondern ermöglicht generell die Prognose jedwediger reellwertiger (real-valued) oder binärer Kontextattribute. Im Hinblick auf die definierten Anforderungen sollen im Folgenden eine Reihe existierender Lösungsansätze untersucht werden, die diese Problemstellung oder verwandte Problemstellungen behandeln. Die Untersuchung konzentriert sich dabei einerseits auf die verwendeten Reasoning-Verfahren und andererseits auf die für das Reasoning verwendeten Kontextattribute, um diese im Hinblick auf die Eignung in der Entwicklung eines generischen Prozesses zur Kontextadaption zu bewerten.

5.5.2. Existierende Lösungsansätze

Mit Blick auf existierende Lösungsansätze lassen sich eine Reihe verwandter Arbeiten identifizieren, die die zu unterstützenden Kontextprognosen oder hieran angrenzende Problemstellungen untersuchen. Diesen Arbeiten ist dabei gemein, dass sie Sensordaten mobiler Geräte verwenden, um mithilfe unterschiedlicher Verfahren zur Mustererkennung eine bestimmte Aktivität des Nutzers, seine Mobilität oder seinen aktuellen bzw. zukünftigen Standort versuchen vorherzusagen. Eine weitere Gruppe verwandter Arbeiten versucht hiervon abweichend den Kontext mobiler Geräte in allgemeinerer Form vorherzusagen. Diese Gruppe untersucht hierzu die Zusammenhänge in historischen Kontextdaten. Die unterschiedlichen Gruppen von Arbeiten sollen im Folgenden vorgestellt werden:

Prognose von Adaptionsbedarfen im mobilen Cloud Computing *Serendipity* [SLAZ12] ist ein Framework, welches versucht die verteilte Ausführung von Tasks in Abhängigkeit von der Zuverlässigkeit einer Verbindung zu einem bestimmten Surrogate zu planen. Wird diese als weitgehend stabil angesehen, so wird zunächst ein *Organisationskanal* (*control channel*) etabliert, um die weitere Ausführung zu planen und in diesem Zusammenhang beispielsweise Metadaten zur Konnektivität auszutauschen und hierdurch die zukünftige Verfügbarkeit eines Surrogates abzuschätzen. Wird die Zuverlässigkeit der Verbindung dagegen als gering angesehen, wird entsprechend ein fehlertoleranteres und robusteres Vorgehen gewählt, durch welches die Tasks an ein Surrogate delegiert werden. Ähnlich versucht auch *IC-Cloud* [SAZN12, Shi14] der Problematik der wechselnden Konnektivität dadurch zu begegnen, dass versucht wird, verschiedene Klassen zu bilden, die die Zuverlässigkeit der Konnektivität beschreiben.

Mit demselben Ziel, die Auslagerung und verteilte Ausführung von Berechnungen zu optimieren, wird in [BDR14a, BDR14b] versucht die Problemstellung der wechselnden Konnektivität zu lösen, indem die Verweildauer an einem bestimmten WLAN- Zugangspunkt vorhergesagt wird, ein Verfahren, das von Berg et al. als *preemptive offloading* bezeichnet wird. Zu diesem Zweck wird ein *Hidden Markov Model*, ein stochastisches Modell, in welchem ein Sy-

stem durch eine Markov-Kette mit unbeobachteten Zuständen modelliert wird, genutzt, um abzuschätzen, ob das Ergebnis der ausgelagerten Berechnung vor dem wahrscheinlichen Zeitpunkt des Verlassens der Empfangszone des jeweiligen WLAN-Zugangspunktes vorliegt und rechtzeitig auf das mobile Gerät transferiert werden kann. Ein solcher Ansatz wird auch von *MobAware* [LS13] und von Siris et al. [SA13] genutzt, um zu entscheiden, ob eine Auslagerung im Zusammenhang mit der zukünftigen Konnektivität eines mobilen Gerätes sinnvoll stattfinden kann.

Mit einem ähnlichen Ziel, lediglich die Konnektivität vorherzusagen, wird von *BreadCrumbs* [NN08] versucht, die zukünftige Konnektivität zu WLAN-Zugangspunkten dadurch abzuschätzen, dass die Historie von Kontakten mit bestimmten Zugangspunkten mit den aktuellen GPS-Daten kombiniert werden, um die Bandbreite dse mobilen Gerätes eines Nutzers vorherzusagen. Von der Lösung *Wiffler* [BMV10] wird ein gleichartiges Ziel verfolgt, hier allerdings um die Nutzung der WWAN-Schnittstelle eines mobilen Gerätes dadurch zu entlasten, dass die Kommunikation auf die WLAN-Schnittstelle verschoben wird.

Prognose der Mobilität und Aktivitätenerkennung Zur Prognose der Mobilität eines Nutzers verwendet *NextPlace* [SMM+11] die Historie der Mobilität eines Nutzers, um hieraus Ähnlichkeiten in Bezug auf seine aktuelle Bewegung zu erkennen und hierdurch den Zeitpunkt und die Verweildauer an einem nächsten, von ihm häufig besuchten Ort festzustellen. Mit demselben Ziel wird von Anagnostopoulos et al. [AAH+09, AAH11] versucht, den nächste Aufenthaltsort eines mobilen Nutzers durch die Wiedererkennung von Bewegungsabläufen vorherzusagen, die zuvor auf Basis einer Historie von Bewegungsabläufen mit bekanntem Ziel gelernt wurden. Weitere Ansätze zur Bestimmung eines zukünftigen Aufenthaltsortes eines Nutzers oder zur Verweildauer finden sich in *PnLUM* [NCW+12] und weiteren Arbeiten wie [PEN10, TCMA12, LHL12, DDMGP15, EKK+13, GPK13, MSCN13], die sich jeweils durch die verwendeten Reasoning-Verfahren und die herangezogenen Kontextattribute unterscheiden.

Eine weitere Gruppe von Lösungsansätzen umfasst die Erkennung von (zukünftigen) Aktivitäten und soll ebenso Berücksichtigung bei der Generierung von Kontextattributen finden. Diese Gruppe zielt darauf ab, die aktuelle Aktivität eines Nutzers zu erkennen, und nutzt hierzu Positionsdaten wie die GPS-Koordinaten [MSP14] oder den Beschleunigungssensor und weitere Sensoren [YHE14, Oh15, CDC10, KMYI09] eines mobilen Gerätes. Die einzelnen Lösungen innerhalb dieser Gruppe unterscheiden sich im wesentlichen darin, welche Kontextattribute sie zum Erkennen und zum Erlernen einzelner Orte verwenden.

Generische Kontextprognose Neben den zuvor vorgestellten Arbeiten, die sich auf die Prognose einzelner Kontextattribute, beispielsweise die Aktivität eines Nutzers konzentrieren, existiert eine weitere, weitaus kleinere Gruppe

verwandter Arbeiten, die es zum Ziel hat, beliebige Kontextattribute zu prognostizieren und sich hierzu unterschiedlicher Eingabegrößen in Form von Kontextattributen zu bedienen. In dieser Gruppe existieren zwei Forschungsarbeiten, die von Sigg [Sig08] und die von Mayrhofer [May04].

Zusammenfassung Mit Blick auf die verwandten Arbeiten soll zunächst festgehalten werden, dass keine von ihnen direkt auf die hier vorliegende Problemstellung übertragen werden kann, da sie sich mehrheitlich auf die Vorhersage spezifischer Kontextattribute beschränken oder eine bestimmte Menge an Eingabevariablen voraussetzen. Entsprechend können lediglich die im Zusammenhang mit den verwandten Arbeiten verwendeten und erzeugten Kontextattribute und die in diesen Arbeiten ausgewählten oder entwickelten Verfahren im Hinblick auf die in dieser Arbeit vorliegende Problemstellung genutzt werden.

Lediglich die letztgenannte Gruppe der generischen Verfahren bietet eine generelle Nutzbarkeit, ist allerdings nicht spezifisch auf das Anwendungsgebiet des Mobile Computings und der hier vorhandenen Ressourcenbeschränkungen ausgerichtet. Im Folgenden sollen die für die hier behandelte Problemstellung relevanten Kontextattribute ermittelt werden.

5.5.3. Auswahl relevanter Kontextattribute

Auf Basis der zuvor vorgestellten Arbeiten sollen im Folgenden die für die Zielwerte der Prognose relevanten Einflussgrößen ermittelt werden. Es soll hierbei festgestellt werden, welche Kontextattribute sich als sinnvoll im Hinblick auf die Prognose der dortigen, verwandten Problemstellungen erwiesen haben und entsprechend für die in dieser Arbeit zu unterstützenden Zielwerte der Prognose Relevanz besitzen können. Hierzu werden die verwendeten Kontextinformationen jeweils in Gruppen eingeordnet, welche in ortsbezogene, zeitliche, nutzer- und gerätebezogene Informationen unterteilen. Hierbei soll in Bezug auf die Auswahl der jeweiligen Kontextattribute auch Kombinationen von Kontextattributen gebildet werden, da solche Kombinationen in vielen Fällen die Prognoseleistung erhöhen [PEN10].

Zielwerte der Prognose Entsprechend den in in Abschnitt 5.5.1 formulierten Anforderungen resultieren hieraus die im Folgenden aufgelisteten Kontextattribute:

- Konnektivität zu einem Surrogate
- Bandbreite in Sende- und Empfangsrichtung
- Erfolg der Ausführung eines Tasks
- Ausführungszeit
- Energievorrat

Kontextinformationen zur WWAN-Schnittstelle Die Prognosen der Konnektivität, Bandbreite und Erfolg der Ausführung eines Tasks sind durch die Kommunikation eines mobilen Gerätes über seine Schnittstellen zu Mobilkommunikation geprägt. Durch die hohe Verfügbarkeit von Weitverkehrsnetzen (WWAN) ist die Mobilkommunikation vor allem von der Erreichbarkeit dieser Netzwerke abhängig. Entsprechend sollen Merkmale wie der Verbindungsstatus, die Signalstärke und der verwendete Mobilfunkstandard der aktuell genutzten GSM-Zelle sowie auch weiterer sichtbarer Zellen Berücksichtigung in der Modellierung finden.

Ähnlich wie von Shin et al. in [SHD12] beschrieben sollen hierzu die häufig erkannten GSM-Zellen von den seltener erkannten GSM-Zellen im Rahmen einer Konstruktion entsprechender höherwertiger Kontextattribute separiert werden. Ebenso soll auch die subjektive Signalstärke, gekennzeichnet durch die Anzahl der Balken im Display, als zusätzliche Kontextinformation genutzt werden. Zudem soll die geschätzte Verweildauer in der aktuellen GSM-Zelle unter Berücksichtigung der durchschnittlichen Verweildauer als neues Kontextattribut konstruiert werden. Zusammenfassend wurden hierbei die im entsprechenden Abschnitt von Tabelle 5.13 beschriebenen Kontextattribute abgeleitet.

Kontextinformationen zur WLAN-Schnittstelle Analog zur WWAN-Schnittstelle soll in Bezug auf diese Schnittstelle zur Mobilkommunikation ermittelt werden, mit welchem Zugangspunkt ein mobiles Gerät verbunden ist, getrennt nach logischer Identifikation (SSID) und physischer Identifikation (MAC-Adresse). In Form eines zusätzlichen neuen Kontextattributs soll zudem berücksichtigt werden, welche weiteren verfügbaren WLAN-Zugangspunkte in Reichweite sind. Zusätzlich soll auch hier die verbleibende Verweildauer in der Reichweite des aktuell verbundenen Zugangspunktes ermittelt werden. Die hieraus abgeleiteten Kontextattribute finden sich im entsprechenden Abschnitt von Tabelle 5.13.

Kontextinformationen zur WPAN-Schnittstelle Obwohl über das aktuell häufig genutzte Bluetooth LE keine besonders hohe Bandbreite erzielt werden kann, geben der Status der Schnittstelle, gefundene Bluetooth-Geräte und deren Signalstärke wertvolle Informationen im Hinblick auf den Kontext eines mobilen Gerätes. Der geringe Energiebedarf dieser Schnittstelle macht es dabei besonders attraktiv, diese aktiv zu halten. Die so erfassten Informationen können anschließend beispielsweise dazu genutzt werden, die zukünftigen Kontakte zur opportunistischen und gemeinsamen Ressourcennutzung abzuschätzen [VREXC11, VRC11]. Die Nutzung der so erfassten Kontextinformationen wird dabei in [JLJ+10] insbesondere dann als nützlich angesehen, wenn diese mit weiteren Kontextattributen kombiniert wird. Der entsprechende Abschnitt von Tabelle 5.13 zeigt die hierzu als relevant angesehenen Kombinationen von Kontextattributen.

Ortsbezogene Kontextinformationen Neben der Information zur Konnektivität ist der Aufenthaltsort eines Nutzers ein wichtiges Indiz in Bezug auf Konnektivität und Ressourcenverfügbarkeit für mobile Geräte. In diesem Zusammenhang bieten sich verschiedene Verfahren an. Neben den bereits mit der WWAN- und WLAN-Schnittstelle eingeführten ortsabhängigen Kontextattributen soll hierzu der häufig in mobilen Geräten vorhandene GPS-Sensor herangezogen werden. Obwohl der GPS-Sensor nur außerhalb von Gebäuden genutzt werden kann, kann die hohe Präzision dieses Sensors dazu dienen, den zukünftigen Aufenthaltsort eines Nutzers vorherzusagen [LHL12] woraus sich wiederum Rückschlüsse auf seine Konnektivität ziehen lassen.

Hierzu ist es erforderlich, die GPS-Koordinaten entsprechend aufzubereiten. In [KMYI09] werden GPS-Informationen beispielsweise dahingehend diskretisiert, dass daraus verschiedene Zonen ermittelt werden. Von Tran et al. wird in [TCMA12] zusätzlich die Semantik eines aktuellen Ortes (beispielsweise „zu Hause“ oder „bei der Arbeit“) als hilfreich in Bezug auf die Prognose der Mobilität eines Nutzers betrachtet. Auch in diesem Fall soll die geschätzte durchschnittliche Verweildauer und verbleibende Zeit an diesem Aufenthaltsort berücksichtigt werden. Der entsprechende Abschnitt von Tabelle 5.13 zeigt die hieraus abgeleiteten Kontextattribute.

Zeitbezogene Kontextinformationen Das Verhalten von Menschen ist oft durch zeitliche Muster geprägt. Entsprechend wurden in einer Reihe von verwandten Arbeiten diesbezügliche High-Level-Kontextattribute entwickelt, die versuchen, diese Muster abzubilden und für Lernverfahren erkennbar zu machen. Dies beginnt bei einfachen Strukturierungsmerkmalen wie der Tageszeit und dem Wochentag [ZZLY13] oder weitergehenden Klassifizierungen, wie sie in [TCMA12] vorgenommen werden, um Feiertage und Ferien zu berücksichtigen. Hierbei besteht die Annahme, dass sich das Verhalten eines Nutzers mobiler Geräte sich zwischen Werktagen und Wochenendtagen deutlich unterscheidet.

Nutzerbezogene Kontextinformationen Nutzerbezogene Kontextinformationen stellen eine weitere Gruppe von möglichen Kontextattributen dar, die Rückschlüsse auf die zukünftige Konnektivität eines mobilen Gerätes und auf die Ausführungswahrscheinlichkeit von Tasks zulassen. Diese Informationen werden häufig über die Aktivität des Nutzers (laufend, sitzend, stehend) abgebildet, wozu üblicherweise die Low-Level-Kontextdaten des Beschleunigungssensors genutzt werden, um auf den High-Level-Kontext in Form der Aktivität des Nutzers zu schließen, wie es in [MGC13] gezeigt wird. In [CDC10] werden zur Erkennung der Aktivität eines Nutzers zusätzlich zeitliche Merkmale wie der Wochentag verwendet; diese Information ist allerdings bereits innerhalb der Kategorie zeitbezogener Kontextinformationen modelliert und wird entsprechend nicht erneut berücksichtigt.

Gerätebezogene Kontextinformationen Neben den Schnittstellen zur Mobilkommunikation kann ebenso der Zustand eines mobilen Gerätes selbst einen Hinweis darauf geben, in welchem Kontext das Gerät verwendet wird, wie die zukünftige Bandbreite ist und ob Taskausführungen erfolgreich sein werden. In diesem Zusammenhang sind insbesondere der verbleibende Energievorrat und die Information, ob das Gerät mit einer externen Energiequelle verbunden ist, von Interesse. Zusätzlich sollen allgemeine Informationen zur Gerätenutzung berücksichtigt werden. Dies umfasst, wann das Gerät zuletzt benutzt wurde, ob es in einen lautlosen Betriebsmodus geschaltet wurde, ob und wann der Bildschirm eingeschaltet war und wann Nachrichten oder Anrufe gesendet bzw. getätigt oder empfangen wurden. Der entsprechende Abschnitt von Tabelle 5.13 fasst die hierfür bereitgestellten Kontextattribute zusammen.

Taskbezogene Kontextinformationen und Partitionierung Neben den zuvor vorgestellten Gruppen von Kontextinformationen, die als relevant im Hinblick auf die Prognose der unterschiedlichen Zielwerte identifiziert wurden, soll ebenso die vergangene Ausführung von Tasks selbst Berücksichtigung finden. Hierzu werden die von der Profiling-Komponente erfassten und in Tabelle 5.4 zusammengefassten Kontextattribute herangezogen.

Diese Informationen werden zusätzlich um das Ergebnis der Partitionierung (vergleiche Abschnitt 5.3.4 und 5.4.4.2) der mobilen Anwendung ergänzt. Hierzu wird anhand der von einem Task als erstes aufgerufenen Komponente geprüft, in welcher Partition diese Komponente (lokal oder entfernt) liegt. Entsprechend ergibt sich ein binäres Merkmal, welches das Ergebnis der Partitionierung widerspiegelt.

Nachdem die wesentlichen Kontextattribute zur Entwicklung der Prognosemodelle identifiziert wurden, soll sich im Folgenden die Entwicklung der Prognosemodelle anschließen.

5.5.4. Entwurf eines Prozesses zur Kontextprognose

Nachdem eine Datenbasis konzipiert wurde, welche einerseits Daten zu den Zielwerten der Kontextprognosen und andererseits die für diese Prognosen relevanten Kontextattribute enthält, soll im Folgenden ein Prozess entworfen werden, der einerseits die relevanten Kontextattribute auswählt, anschließend aus einer Reihe von Reasoning-Verfahren ein passendes Verfahren zur Mustererkennung wählt und letztlich als Ergebnis ein entsprechendes Prognosemodell generiert. Die Entscheidung für diesen generischen Ansatz soll eine Umsetzbarkeit des entwickelten Konzepts auf einer Vielzahl von mobilen Geräten ermöglichen. Auch auf solchen, auf denen unter Umständen nur eine Teilmenge der als relevant identifizierten Kontextattribute verfügbar ist. Dies zielt auf die Anforderung ab, dass auch bei einer eingeschränkten Verfügbarkeit von Kontextdaten noch Prognosen ermöglicht werden sollen.

	Attribut(-Gruppen)	Beschreibung
Ort	DistanceFromTop[N=1..3]Location	Entfernung vom Top[N]-Standort in Metern
	IsAtTopPlace[N=1..3]	Gerät ist am N-häufigsten Ort
Zeit	BankHoliday	Aktuelles Zeitintervall liegt auf einem Feiertag
	Month[N=1..12]	Zeitintervall liegt im Monat N eines Jahres
	SchoolVacation	Aktuelles Zeitintervall liegt in Schulferien
	TimeHour[N=0..23]	Binäre Modellierung der Tageszeit time_hour_0: N:00 - N:59
	TimeSlot[N=1..6]	Modellierung einzelner Tageszeiten timeslot_N: 0:00 - 3:59
	TimeUntilLeave	Geschätzte Zeit in Minuten bis zum Verlassen des aktuellen Ortes
	Weekday[N=1..7]	Zeitintervall liegt in Tag N einer Woche
	Weekend	Zeitintervall liegt auf einem Wochenende
Kalender	CalendarInEvent	Ein Kalenderereignis findet aktuell statt
	CalEventEndingIn[N=10,20,..60]	Kalenderereignis endet in N Minuten
	CalEventIn[N=10,20,60]InTop[M=1..3]	Kalendereintrag in N Minuten für Top M-Standort
	CalLeaveTime	Minuten bis zum Ende des Kalenderereignisses
Aktivität und gerätebezogene Daten	ActivityEvent	Ergebnis der Activity-Prediction eines mobilen Geräts
	BatteryCharging	Mobiles Gerät ist mit externer Energiquelle verbunden
	BatteryLevel	Energievorrat des Akkus in %
	BatteryLevel[N=10,20,70]	Energievorrat des Akkus in % über dem Schwellwert von N %
	CallReceivedLast[N=10,20,60]	Gerät hat einen Anruf in den letzten N Minuten empfangen
	ChargingCompleted	Akku ist vollständig geladen
	FreeMemory	Freier Arbeitsspeicher des mobilen Gerätes in Kilobyte
	InactiveTime	Zeit in Sekunden, die das mobile Gerät nicht benutzt wurde
	ScreenOn	Bildschirm des Geräts ist eingeschaltet
	SilentModeSwitch	Mobiles Gerät ist im lautlosen Modus
WPAN	BluetoothActivated	Bluetooth-Schnittstelle ist eingeschaltet
	BluetoothMacCount	Anzahl über Bluetooth gefundener MAC-Adressen
WWAN	GSMAverageStrength	Durchschnittliche Signalstärke der verbundenen GSM-Zelle
	ConnectionType	Aktuell verwendeter Mobilfunkstandard der WAN-Schnittstelle
	GSMActivated	GSM-Schnittstelle ist eingeschaltet
	GSMBars	GSM-Signalstärke als Anzahl der angezeigten Balken
	GSMConnected	Gerät ist mit einer GSM-Zelle verbunden
	GSMLeavingIn	Geschätzte Zeit in Minuten bis zum Verlassen der aktuellen GSM-Zelle
	GSMSignalCount	Anzahl sichtbarer GSM-Zellen
	GSMTop[N=1..3]Visible	Sichtbarkeit der Top N-GSM-Zelle
	GSMTop3Visible	Sichtbarkeit einer der drei am häufigsten genutzen Zellen
WLAN	WiFiAverageStrength	Durchschnittliche Signalstärke des verbundenen WLAN-Zugangspunkts
	WiFiActivated	WLAN-Schnittstelle ist eingeschaltet
	WiFiConnected	Gerät ist mit einem WLAN-Zugangspunkt verbunden
	WiFiLeavingIn	Geschätzte Zeit in Minuten bis zum Verlassen des aktuellen WLANs
	WiFiSignalCount	Anzahl sichtbarer WLAN-Zugangspunkte
	WiFiVisibleTop[N=1..3]	Sichtbarkeit des Top N-WLAN-Zugangspunkts
Bandbreite	BandwidthDown	Durchschnittliche Bandbreite im Download in Kbit/s
	BandwidthUp	Durchschnittliche Bandbreite im Upload in Kbit/s
	Trusted	Generierung dieses Datensatzes war möglicherweise fehlerbehaftet

Abbildung 5.13.: Liste der Kontextattribute als Basis für die Entwicklung der Prognosemodelle

Mit Blick auf die im vorgehenden Abschnitt vorgestellte Menge von Komponenten der Systemunterstützung wird dieser Prozess im Wesentlichen durch die im Kontext-Manager enthaltene Prognoselogik abgebildet, die in Abbildung 5.14 veranschaulicht wird.

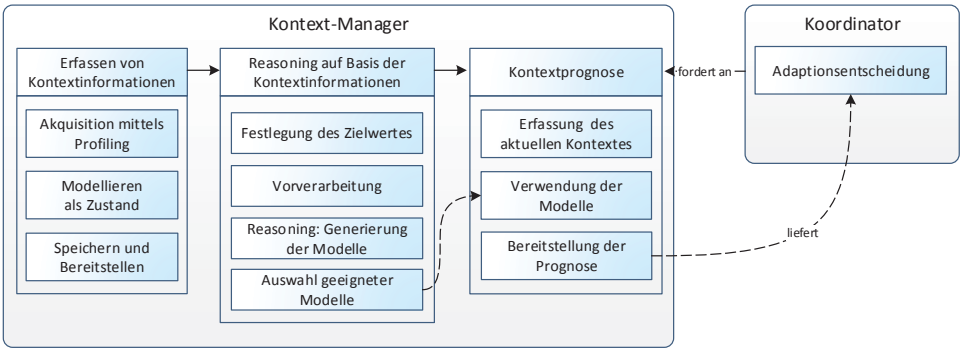

Abbildung 5.14.: Interaktion zwischen dem Koordinator und der Kontextprognose

Methodisch orientiert sich das nun folgende Vorgehen dabei an dem in [NMF+05] vorgestellten Prozess zur Vorhersage von Kontextattributen und dem in Abschnitt 3.2 beschriebenen Vorgang des Reasonings auf Kontextdaten.

Akquisition von Kontextdaten Vor dem Hintergrund, dass die zu unterstützenden mobilen Geräte ihren Kontext über eine Vielzahl von Sensoren wahrnehmen können, soll sich auch die Nutzung der im Hinblick auf die Prognosen verwendeten Kontextattribute auf eine möglichst breite Basis stützen und daher alle im Hinblick auf die zu prognostizierenden Kontextattribute relevanten Einflussfaktoren berücksichtigen. Aus konzeptioneller Sicht gilt es an dieser Stelle, die im vorhergehenden Abschnitt ermittelten Kontextinformationen zu erfassen und auf dem mobilen Gerät zu speichern.

Vorverarbeitung Dem vorgestellten Prozess der Kontextverarbeitung folgend beginnt die Verarbeitung von Kontextdaten mit deren Akquisition, an die sich deren Modellierung anschließt. Im Folgenden sollen entsprechend wesentliche Vorverarbeitungsschritte beschrieben werden, die als Grundlage für die anschließende Generierung der Prognosemodelle dienen.

Transformation und Aggregation Aus konzeptioneller Hinsicht sollen für das Reasoning-Verfahren eine Reihe von Verfahren zur Mustererkennung genutzt werden, die aus dem Bereich des überwachten maschinellen Lernens stammen. Eine Entscheidung, die im Verlauf dieses Kapitels noch begründet wird. Entsprechend geht es darum, die Daten im Hinblick auf die Verwendung dieser Gruppe von Verfahren aufzubereiten.

Das Ziel des maschinellen Lernens ist die Erkennung eines Musters in gegebenen Beispieldaten mithilfe eines Lernalgorithmus, um basierend

auf dem Muster neue Daten klassifizieren bzw. approximieren zu können. Das Muster wird dabei in der Trainingsphase aus den Daten „gelernt“ und in einem Modell festgehalten. Beim überwachten Lernen bestehen die Beispiele dabei aus Eingabedaten und den gewünschten Ausgabedaten. In Bezug auf die in dieser Arbeit betrachteten Verfahren wird beim sogenannten Lernen durch Minimierung einer Fehlerfunktion eine Entscheidungsfunktion gesucht, welche die Zusammenhänge zwischen Eingabe- und Ausgabedaten in geeigneter Weise beschreibt. Um das dabei gelernte Wissen nutzbar zu machen, werden dem in der Trainingsphase erzeugten Modell in einer zweiten Phase, die Eingabedaten neuer Beispiele präsentiert, für die das Modell nun die gewünschten Ausgabedaten liefern soll. In dieser Einsatz- oder Testphase sollen also neue Beispiele durch das im Modell enthaltene Wissen ausgewertet werden.

Die Trainingsdaten bestehen dabei aus n Beispielen der Form $(x_1, y_1), \ldots, (x_n, y_n)$. Die Trainingsdaten bestehen aus m-dimensionalen Vektoren der Form $x \in \mathbb{R}^m$ und Zielwerten y, welche für das für Klassifikationsprobleme diskret (zum Beispiel $y \in \{+1, -1\}$) oder für Regressionsprobleme stetig sein ($y \in \mathbb{R}$) können. Das einfachste Beispiel ist damit ein binäres Klassifikationsproblem, dessen Entscheidungsfunktion nur zwei mögliche Werte hat, welche die Klassenzugehörigkeit kennzeichnen: $y \in \{+1, -1\}$.

Um die Daten für die unterschiedlichen Reasoning-Verfahren so aufzubereiten, dass eine hohe Erkennungsleistung ermöglicht wird, gilt es, diese in eine passende Form zu transformieren. Hierzu werden die Daten der einzelnen Kontextattribute üblicherweise in Zeitintervallen zusammengefasst (vergleiche unter anderem [SMM+11, DDMGP15]); mit dem Ziel, diese in eine allgemeine Beschreibung von Kontextzuständen umzuwandeln. Der Füllgrad der einzelnen Kontextattribute in den einzelnen Kontextzuständen wird hierdurch erhöht, da deren Wert innerhalb eines jeweiligen Zeitintervalls als konstant betrachtet wird [HS08]. Entsprechend werden die Messwerte im Rahmen einer Intervallbildung transformiert, wie in Abbildung 5.15 veranschaulicht wird. Im Hinblick auf das konkret zu wählende Zeitintervall orientieren sich die verwandten Arbeiten dabei einerseits an der zu erzielenden Genauigkeit und andererseits an der verfügbaren Auflösung der Sensordaten.

Fehlerbereinigung Aus konzeptioneller Sicht gilt es, fehlerhafte Messwerte zu korrigieren, die einen negativen Einfluss auf die Entwicklung der Prognosemodelle besitzen können. Welche Fehler es hierbei zu korrigieren gilt, ist allerdings stark von der Art der Erfassung der Sensordaten abhängig.

Im Rahmen der Analyse unterschiedlicher Datensätze wurde eine Reihe von fehlerhaften Messwerten identifiziert, die teilweise auf die zur Messung verwendete Software [MAE14] zurückzuführen sind. Diese Messwerte wurden entsprechend korrigiert und, falls keine sinnvolle Korrektur erfolgen konnte, entsprechend markiert (vergleiche Kontextat-

USERID	TIMEDATE	GSM_BARS	GSM_BW	WIFI_BW	IS_ONLINE	...
5479	02.09.2019 14:34	7	9141	0	true	
5479	02.09.2019 14:36	7	9141	0	true	
5479	02.09.2019 14:38	7	9625	0	true	
5479	02.09.2019 14:40	7	10109	0	true	
5479	02.09.2019 14:42	7	10109	0	true	
5479	02.09.2019 14:44	7	10109	0	true	
5479	02.09.2019 14:46	7	8162	0	true	
5479	02.09.2019 14:48	7	6118	0	true	

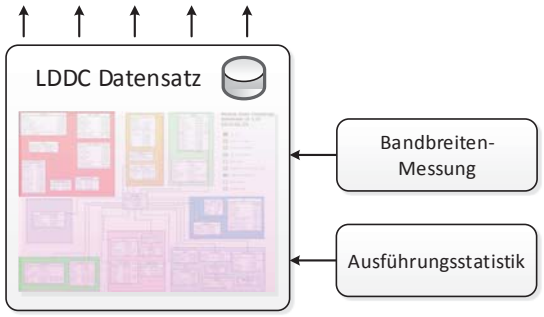

Abbildung 5.15.: Transformation der Kontextdaten für das Reasoning

tribut *Trusted* in Tabelle 5.13). Diese Korrekturen erfolgen vor allem für Zeiträume, in denen der verbleibende Energievorrat des mobilen Gerätes gering war und sich das Messverhalten der zur Erfassung der Kontextdaten verwendeten Software verändert hat.

Diskretisierung Von Kamikasa et al. [KMYI09] wird die Diskretisierung von Kontextattributen generell als sinnvoll im Hinblick auf das Reasoning mittels Machine Learning gesehen. In einfachen Fällen werden hierzu von Shin et al. in [SHD12] numerische Kontextattribute mithilfe eines sogenannten *Binnings* in eine ordinale Darstellung umgewandelt, beispielsweise wird die Anzahl der Bluetooth-Kontakte so den Klassen „sehr viele“, „viele“ bis „sehr wenige“ zugeordnet.

Bei bestimmten Kontextattributen wie den GPS-Koordinaten gilt es allerdings, weitergehende Verfahren zur Diskretisierung zu verwenden und diese, wie in [CTSC11] beschrieben, semantisch sinnvoll zuzuordnen, beispielsweise in verschiedene Zonen die sich aus der Entfernung zu häufig besuchten Orten ergeben [SHD12]. Aus konzeptioneller Sicht wurden hierzu im vorhergehenden Abschnitt bereits eine Reihe von High-Level-Kontextattributen durch eine Diskretisierung abgeleitet. Um allerdings die Diskretisierung auf beliebige Kontextattribute erweitern zu können, wäre es erforderlich, ein automatisiertes Binning durchzuführen. Im Hinblick auf einen begrenzten Nutzen dieses automatisierten Binnings soll auf eine solche Diskretisierung der Kontextattribute im Rahmen dieser Arbeit allerdings verzichtet werden.

Normalisierung Eine Normalisierung numerischer Kontextattribute wird bei distanzbasierten Verfahren, gerade wenn unterschiedliche Kontextattribute zueinander gleichwertig dargestellt werden sollen, generell als sinn-

voll vor dem Einsatz von Reasoning-Verfahren angesehen und daher in dem hier vorgeschlagenen Verfahren für alle Attribute durchgeführt. Hierbei wird eine *z-Transformation* angewendet, bei welcher die resultierende Variable des Kontextattributes den Erwartungswert null und die Varianz eins besitzt.

Auswahl relevanter Kontextattribute Im Hinblick auf die unterschiedlichen Zielwerte der Prognose sind mitunter nicht alle in Tabelle 5.4 und 5.13 zusammengestellten Kontextattribute erforderlich. Zusätzlich behindert das Vorhandensein einer großen Anzahl nichtrelevanter Kontextattribute in manchen Fällen die Erkennung relevanter Muster. Weiterhin soll vor dem Hintergrund der nichtfunktionalen Anforderungen der Effizienz die zu verarbeitende Datenmenge möglichst klein gehalten werden. In diesem Zusammenhang sollen die in Tabelle 5.4 und 5.13 identifizierten Kontextattribute zwar erfasst und gespeichert werden, die Nutzung im Hinblick auf die Generierung der Prognosemodelle soll sich jedoch auf die jeweils für eine gute Prognosequalität relevanten bzw. ausreichenden Merkmale beschränken. Entsprechend soll eine Auswahl relevanter Attribute (feature selection) als letzter Schritt der Generierung der Prognosemodelle vorausgehen, die im Folgenden beschrieben wird. Die häufig genutzten Verfahren zur Auswahl relevanter Attribute lassen sich nach [GE03] drei verschiedenen Klassen zuordnen:

Die erste Gruppe stellen filterbasierte Ansätze dar, bei denen für jedes einzelne Attribut ein Maß zur Beurteilung der Qualität desselbigen ermittelt wird. Hierzu kann entweder ein Streuungsmaß wie die Varianz herangezogen werden, um den Informationsgehalt abzuschätzen (univariate Verfahren), oder ein korrelationsbasierter Ansatz gewählt werden, bei dem der Informationsgehalt eines Attributs in Abhängigkeit zu weiteren Attributen gesetzt wird (multivariate Verfahren). Anschließend wird dieses Maß zur Abzählung der n wichtigsten Attribute verwendet, wobei die Bestimmung des Parameters n durch den Anwender vorzunehmen ist. Der Vorteil dieser Verfahren liegt in ihrer einfachen Anwendbarkeit. Ein generelles Problem dieser Form der Merkmalsselektion kann allerdings darin bestehen, dass nichtlineare Zusammenhänge zwischen Kombinationen von Attributen und dem Zielwert nicht mehr erkannt werden können, da das Reasoning-Verfahren nur noch die Teilmenge der Attribute erhält, die in einer direkten Abhängigkeit zum Zielwert stehen. Ebenso berücksichtigt diese Gruppe von Verfahren nicht die Auswahl eines bestimmten Reasoning-Verfahrens, sondern versucht lediglich den allgemeinen Informationsgehalt bestimmter Attribute abzuschätzen. Im Hinblick auf das in dieser Arbeit relevante Problemfeld der Verarbeitung von Kontextdaten auf mobilen Geräten kann damit die Einfachheit der Verfahren und der daraus resultierende geringe Rechenaufwand als vorteilhaft angesehen werden. Die aufgezeigten Begrenzungen hinsichtlich der Erkennung kombinierter Abhängigkeiten sind im Hinblick auf die mög-

liche Komplexität der zu erkennenden Zusammenhänge jedoch kritisch zu beurteilen. Beispielsweise können zwei Kontextattribute wie die Tageszeit und der aktuelle Energievorrat eines mobilen Gerätes für sich genommen nur bedingt aussagekräftig im Hinblick auf die Vorhersage der verbleibenden Laufzeit eines mobilen Geräts erscheinen. Die Kombination beider Merkmale kann diese Vorhersage deutlich verbessern, da anzunehmen ist, dass die Nutzung des mobilen Geräts (und damit die Inanspruchnahme des Energievorrats) von der Tageszeit abhängig ist.

Die zweite Gruppe von Verfahren stellen eingebettete Verfahren dar, bei denen die Ermittlung der Teilmenge relevanter Attribute direkt mit dem entsprechenden Reasoning-Verfahren verbunden ist. Beispielsweise kann hierzu die vom Reasoning-Verfahren der Support-Vektor-Maschine ermittelte Gewichtung der einzelnen Attribute genutzt werden. Diese Gruppe von Verfahren ist, sofern das verwendete Reasoning-Verfahren diesen Zusammenhang erkennt, in der Lage, die im Zusammenhang mit den filterbasierten Ansätzen genannte Kombination von Attributen angemessen zu berücksichtigen. Zudem ist davon auszugehen, dass die ermittelte Teilmenge im Hinblick auf die Mustererkennung besonders gut vom verwendeten Reasoning-Verfahren verwendet werden kann. Im Umkehrschluss gilt es jedoch zu berücksichtigen, dass das ermittelte Ergebnis stark vom verwendeten Lernalgorithmus abhängig ist. Im Hinblick auf die geforderte Generizität der Kontextadaption und die hierzu eingesetzten unterschiedlichen Reasoning-Verfahren wäre es entsprechend erforderlich, für jedes der zu nutzenden Lernalgorithmen ein entsprechendes eingebettetes Verfahren zur Merkmalsselektion nutzen zu können.

In der dritten Gruppe von Verfahren erfolgt die Selektion geeigneter Attribute mithilfe eines Wrappers, der eine Reihe von Attribut-Teilmengen auf das jeweils zu verwendende Reasoning-Verfahren anwendet. Die initiale Bildung dieser Teilmengen erfolgt dabei entweder zufällig, oder es wird eine minimale oder maximale Menge von Attributen als Startpunkt festgelegt. Anschließend wird diese Teilmenge verändert, wobei zum Entfernen oder Hinzufügen zusätzlicher Attribute entweder Heuristiken zum Einsatz kommen oder einzelne Attribute iterativ ausgewählt und anschließend entfernt oder hinzugefügt werden. In beiden Varianten wird dabei nach jeder Veränderung der Menge von Attributen die Qualität des durch den Lernalgorithmus erzeugten Modells beurteilt. Ebenso wie die eingebetteten Verfahren besteht der Vorteil dieser dritten Gruppe von Verfahren darin, dass sie durch ihre direkte Interaktion mit dem Lernalgorithmus auszeichnen und die gebildeten Teilmengen von Attributen gleichermaßen gut vom verwendeten Reasoning-Verfahren verwendet werden kann. Gegenüber der ersten Gruppe von Verfahren besteht ihr Vorteil zusätzlich darin, dass sie redundante Attribute erkennen und entsprechend entfernen können. Der Nachteil dieser Verfahren ist hingegen vor allem im für die Nutzung notwendigen Rechenaufwand zu sehen.

Dieser kann in Abhängigkeit von der zu untersuchenden Anzahl von Attributen schnell ansteigen, sofern alle möglichen Teilmengen untersucht werden sollen. Beim Einsatz entsprechender Heuristiken, die lediglich eine Teilmenge der möglichen Kombination untersuchen, um diesen Rechenaufwand zu minimieren, besteht hingegen das Risiko nur ein lokales Optimum zu finden.

Das gemeinsame Qualitätskriterium der verschiedenen Gruppen von Verfahren ist in der Auswahl einer geeigneten Menge von Attributen zu finden, die die vom Reasoning-Verfahren zu erkennenden Vorhänge bestmöglich abbilden. Im Hinblick auf das hier relevante Anwendungsgebiet mobiler Geräte sind allerdings weitere Kriterien ausschlaggebend für die Eignung eines bestimmten Verfahrens: so ist beispielsweise die Leistungsbeschränkung mobiler Geräte oft entscheidend für die Auswahl eines Verfahrens [KMS+10]. Mit Blick auf verwandte Arbeiten konzentrieren sich die konkreten Implementierungen entsprechend oft auf einfache Verfahren, beispielsweise erfolgt in [CDC10]die Auswahl der Kontextattribute mobiler Geräte basierend auf ihrer informationstheoretischen Relevanz, bei gleichzeitig minimaler Korrelation zu den übrigen Kontextattributen.

Entsprechend bietet sich zur Auswahl ein Verfahren wie die Gewichtung mithilfe einer Hauptkomponententransformation an, bei dem die n ersten, maximal dekorellierten Hauptkomponenten der ursprünglichen Menge von Kontextattributen gebildet werden. Der Vorteil dieses Ansatzes ist es, dass keine Information verloren geht, sich jedoch gleichzeitig die in den folgenden Schritten zu verarbeitende Datenmenge deutlich reduziert.

Generierung der Prognosemodelle Nachdem die Daten in eine geeigneter Weise vorverarbeitet wurden, werden anschließend die unterschiedlichen Reasoning-Verfahren ausgewählt. Zuvor sollen allerdings die unterschiedlichen Vorgehensweisen in Bezug auf die Ebenen der Kontextattribut-Prognose bewertet werden.

Es besteht hierbei einerseits die Möglichkeit, mehrstufig vorzugehen und zunächst die zukünftige Ausprägung eines Low-Level-Kontextattributes vorherzusagen und anschließend auf ein hieraus abgeleitetes höherwertiges Kontextattribute zu schließen. So wurde von Sohn et al. in [SVL+06] versucht, die Mobilität eines Nutzers mithilfe der Verbindungshistorie zu GSM-Masten herzustellen. Hier wird zunächst vorhergesagt, ob sich ein Nutzer bewegt, um dann in einer zweiten Stufe vorherzusagen, welche Form der Bewegung vorliegt.

Andererseits besteht auch die Möglichkeit, direkt vorzugehen und aus der aktuellen Ausprägung einer Menge von Low-Level- und High-Level-Kontextattributen direkt auf die zukünftige Ausprägung eines bestimmten High-Level-Kontextattributs zu schließen, wie beispielsweise in [MGC13] im

Zusammenhang mit der Vorhersage der zukünftigen Aktivität eines Nutzers vorgegangen wird. Abbildung 5.16 veranschaulicht diesen Zusammenhang und zeigt auf, dass ebenso Mischformen dieser zwei Varianten möglich sind.

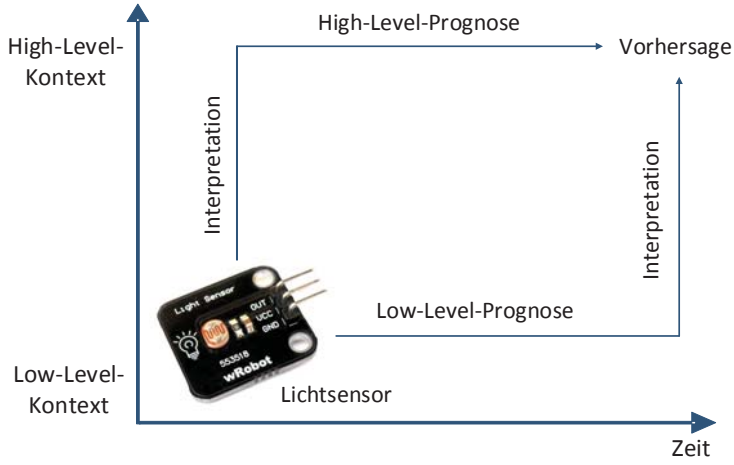

Abbildung 5.16.: Vorhersage der Kontextattribute

Mit Blick auf die Untersuchung von Sigg in [Sig08] soll hierbei festgehalten werden, dass die direkte Vorhersage eines High-Level-Kontextes oft zu besseren Resultaten geführt hat als eine indirekte, insbesondere dann, wenn die Anzahl der hierfür verwendeten Kontextattribute hoch war. Dieses Vorgehen begünstigt ebenso die Nutzung beliebiger weiterer verfügbarer Low-Level-Kontextattribute, die andernfalls mit separaten Modellen vorhergesagt werden müssten, und kommt damit der hier vorliegenden Anforderung eines möglichst generischen Ansatzes zur Kontextprognose entgegen.

Entsprechend werden die für die Generierung der Prognosemodelle zur Verfügung stehenden Reasoning-Verfahren zunächst jeweils auf historischen Kontextinformationen trainiert und im Anschluss deren Qualität mithilfe einer Testmenge evaluiert, wie in Abbildung 5.17 veranschaulicht ist.

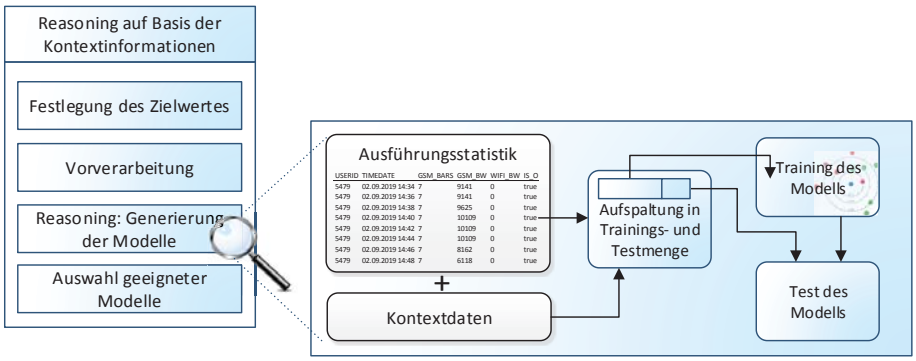

Abbildung 5.17.: Beispiel für das Training eines Prognosemodelle

Bei der Generierung dieser Modelle gilt es zu unterscheiden, inwieweit der jeweilige Zielwert der Prognose zwar kontextabhängig, jedoch unabhängig vom

einem bestimmten zeitlichen Horizont vorhergesagt werden soll. Der erste Fall trifft beispielsweise auf die in Abbildung 5.18 gezeigte Nutzung eines Klassifikators zur Vorhersage einer Taskausführung zu. Der zweite Fall, in dem eine solche Abhängigkeit besteht, ist in Abbildung 5.19 gezeigt.

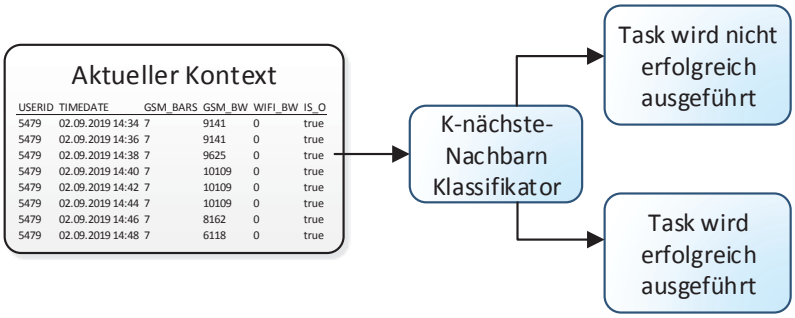

Abbildung 5.18.: Beispiel für das Training eines Klassifikators: Vorhersage der Taskausführung

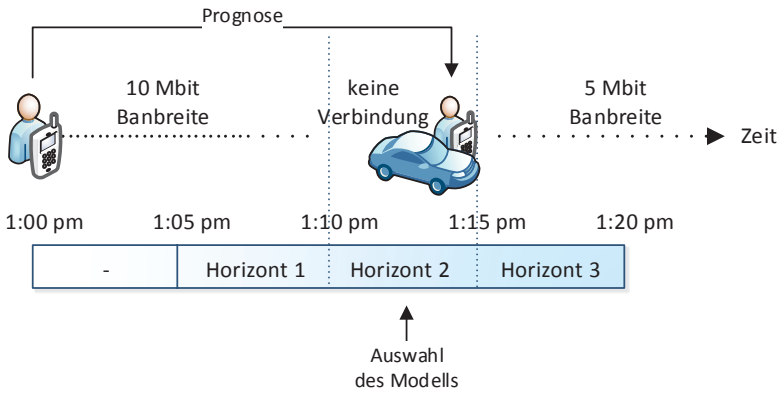

Abbildung 5.19.: Beispiel für den Bedarf unterschiedlicher Prognosemodelle: Vorhersage der Bandbreite

Hierbei wird ersichtlich, dass im zweiten Fall unterschiedliche Modelle generiert werden müssen.Wie die Anzahl der für diesen Fall vorzuhaltenden Modelle bestimmt wird, ist dabei vom konkreten Anwendungsfall und den sich hieraus ergebenden prognostischen Anforderungen abhängig. Die Ermittlung relevanter Prognosehorizonte wird entsprechend im Rahmen der prototypischen Implementierung erneut aufgegriffen.

Auswahl geeigneter Lernverfahren Im Hinblick auf die geforderte Generizität und die Anpassungsfähigkeit in Bezug auf unterschiedliche Eingabevariablen sowie unterschiedliche Zielwerte der Prognose soll als Bestandteil des generischen Prozesses zur Kontextadaption kein bestimmtes Reasoning-Verfahren vorab festgelegt werden, sondern stets das für das jeweilige Prognose-Szenario im Hinblick auf die Prognosequalität passendste Verfahren dynamisch ausgewählt werden. Aus konzeptioneller Sicht soll hierzu im Rahmen des Trainings jeweils eine Auswahl von Reasoning-Verfahren gegeneinan-

der evaluiert werden, um hieraus ein spezifisches Prognosemodell zu generieren, das auf dem jeweils am besten geeigneten Reasoning-Verfahren basiert.

Für diese Auswahl sollen vornehmlich bereits in unterschiedlichen Forschungsarbeiten angewendete Verfahren herangezogen werden, da deren Nutzung im Hinblick auf die hier vorhandenen Ziele der Prognose als erfolgversprechend angesehen wird. Aus den Abschnitt 3.2.3 vorgestellten Kategorien im Bereich der Mustererkennung in Kontextdaten wurde von Lim und Dey in [LD10] untersucht, welche konkreten hierfür vornehmlich Anwendung finden. In [PZCG14] wurde diese Auswertung wieder aufgegriffen und hierbei die folgenden Verfahren als häufigste identifiziert:

- Entscheidungsbäume (15 %)
- Regelbasierte Systeme (54 %)
- Hidden Markov Models (13 %)
- Naive Bayes (13 %)
- Support-Vektor-Maschinen (4 %)
- K-nächste-Nachbarn (2 %)

Hierbei ist zu erkennen, dass sich die konkret angewendeten Verfahren auf vergleichsweise wenige konzentrieren und diese andererseits vornehmlich der Kategorie des überwachten Lernens zugerechnet werden können (vergleiche Abschnitt 3.2.3). Im Hinblick auf die nichtfunktionalen Anforderungen, die im Zusammenhang mit dem Adaptionsprozess definiert wurden, ergeben sich allerdings weitere Restriktionen, die in erster Linie auf die begrenzten Ressourcen mobiler Geräte zurückzuführen sind. Hieraus wiederum ergeben sich die anhand ihrer Laufzeit (sowohl im Training und in der Prognose) ausgewählten folgenden Kandidaten von Reasoning-Verfahren:

- Logistische Regression (Klassifikation)
- Entscheidungsbäume (Klassifikation)
- Naive Bayes (Klassifikation)
- Support-Vektor-Maschinen (Klassifikation und Regression)
- K-nächste-Nachbarn (Klassifikation und Regression)
- Regelbasierte Systeme (Klassifikation)
- Lineare Regression (Regression)

Abschließend soll im folgenden Abschnitt bewertet werden, anhand welcher Kriterien die vom zuvor konzipierten Prozess generierten Modelle ausgewählt werden.

5.5.5. Berücksichtigung der Cold-Start-Problematik

Die Situation, in der ein selbstlernendes System nur im Rahmen seiner begrenzten Wissensbasis Zusammenhänge erkennen kann, wird als Problem des *Cold Start* bezeichnet [SPUP02]. Diese aus dem Bereich der Empfehlungssysteme (recommender systems) stammende Problematik bezieht sich darauf, dass für Objekte oder Situationen, zu denen noch keine Beobachtungen in Form von beschreibenden Informationen vorliegen, Empfehlungen zu generieren sind. In Bezug auf die in dieser Arbeit behandelte Problemstellung der generischen Kontextadaption bezieht sich die Cold-Start-Problematik darauf, das Kontextattribute eine Prognose zu erstellen ist, zu denen noch keine Historie vorliegt.

Hinsichtlich der Kontextdaten eines mobilen Gerätes kann diese Historie weitestgehend passiv und durch Auslesen der entsprechenden Sensoren erzeugt werden. Die Ausführungsstatistik einer mobilen Anwendung hingegen kann lediglich durch die tatsächliche Ausführung von Tasks generiert werden. Hierfür ist es erforderlich, zu Beginn sämtliche existierenden Varianten der Taskausführung zu beobachten und dieses Verhalten bei zunehmend großer Historie entsprechend zu reduzieren. Dieses Verhalten kann entsprechend auch bei einer ausreichend großen Ausführungsstatistik nicht gänzlich unberücksichtigt bleiben, da diese Fähigkeit zur Exploration sicherstellt, dass veränderte Bedingungen existierender Surrogates oder gänzlich neue Surrogates auch tatsächlich berücksichtigt werden.

Im Hinblick auf die Adaptionsentscheidungen einer kontextadaptiven Anwendungsarchitektur soll dieser Problematik entsprechend durch das regelmäßige Explorieren alternativer, vermeintlich suboptimaler Adaptionsstrategien vorgebeugt werden. Diese Exploration führt im Hinblick auf die Generierung der Ausführungsstatistik dazu, dass abhängig vom Adaptionsziel pro Task mehrere Datensätze erfasst werden, die sowohl identische als auch unterschiedliche Statuscodes für dieselbe Taskausführung erzeugen können. Entsprechend werden in diesem Fall mehrere Datensätze an den Kontextmanager übertragen, um zukünftige Adaptionsentscheidung bestmöglich unter Berücksichtigung sowohl erfolgreicher als auch nicht erfolgreicher Adaptionen optimieren zu können.

Nachdem aufgezeigt wurde wie die fehlenden Daten generiert werden, soll abschließend skizziert werden welches konkrete Vorgehen gewählt wird, um bereits ohne Vorhandensein einer Historie Prognosen von Kontextattributen bereitstellen zu können. Analog zum von Schein et al. in [SPUP02] vorgeschlagenen hybriden Ansatz soll an dieser Stelle ein mehrstufiges Verfahren genutzt werden bei dem triviale Prognosen herangezogen werden, um diese bei einer ausreichenden Datenversorgung durch den eigentlichen, zuvor beschriebenen, spezifisch trainierten Modelle zu ersetzen. Wie dies konkret umgesetzt ist, wird im Zusammenhang mit der Implementierung in Abschnitt 6.4.6 beschrieben.

5.5.6. Auswahl geeigneter Modelle

Zur Auswahl geeigneter Modelle gilt es zunächst, domänenspezifische Erfolgskriterien festzulegen, welche sich üblicherweise durch ein geeignetes Fehlermaß beschreiben lassen. Diese sollen im Folgenden in Abhängigkeit von der Art des jeweiligen Zieles der Prognose ermittelt werden.

Binäre Klassifikation Als Fehlermaße zur Beurteilung der Qualität einer binären Klassifikation bieten sich *Precision* und *Recall* an. Ersteres beschreibt in Form einer Quote den Anteil der korrekt der positiven Klasse zugehörig klassifizierten Ergebnisse an. Zweites gibt den Anteil der korrekt als positiv klassifizierten Objekte an der Gesamtheit aller positiv klassifizierten Objekte an und bezeichnet damit die Quote, mit der das jeweilige Beispiel richtig klassifiziert wird. Entsprechend wird das F-Maß, oder genauer der F1-Score, welches dem gewichteten harmonischen Mittel entspricht, als kombinierter Güteschätzer dieser beiden Merkmale herangezogen.

Regression Als Fehlermaß zur Beurteilung der Qualität der Regressionsschätzung soll die mittlere absolute Abweichung betrachtet werden. Einerseits deswegen, weil ein Über- und Unterschätzen gleich bewertet werden soll, und andererseits, weil die Kosten einer Fehleinschätzung mit Blick auf die Ziele der Prognose als linear anzusehen sind.

Die ausgewählten Fehlermaße erlauben es entsprechend, die vom Prozess der generischen Kontextadaption unterstützten Ziele der Prognose in Bezug auf die Qualität der jeweiligen Vorhersage zu bewerten.

5.5.7. Zusammenfassung der generischen Kontextadaption

Das entwickelte Konzept erlaubt es hierdurch mobilen Anwendungen, sich proaktiv an ihren Kontext anzupassen, die Potenziale einer Kooperation mit der Infrastruktur zu nutzen und sich zu adaptieren. Da das Formulieren entsprechender Adaptionsstrategien vor dem Hintergrund nicht absehbarer Nutzungsszenarien mobiler Anwendungen den Entwickler vor eine große Herausforderung stellen würde, erfolgte die Auswahl eines Ansatzes bei dem die entsprechenden Adaptionsstrategien auf Basis des Kontextes mobiler Geräte durch die Anwendungsarchitektur selbst gelernt werden. Hierzu wurden zunächst die wesentlichen Informationen identifiziert, die eine Entscheidung innerhalb einer Adaptionsstrategie beeinflussen, wie beispielsweise die Konnektivität und Bandbreite zu benötigten Surrogates. Diese Informationen werden anschließend für den erwarteten Zeitraum einer Taskausführung prognostiziert, wozu die jeweils auf dem mobilen Gerät vorhandenen Kontextinformationen genutzt werden, um mithilfe statistischer Lernverfahren ein entsprechendes Prognosemodell zu generieren. Dieser Prozess der Generierung von Prognosemodellen ist dabei generisch in Bezug auf die Auswahl und Gewichtung der zu berücksichtigenden Kontextattribute ausgelegt, um die individu-

ellen Nutzungsmuster eines mobilen Geräts und die unterschiedliche Menge vorhandener Einflussgrößen angemessen zu berücksichtigen.

5.6. Zusammenfassung

Nachdem in Kapitel 4 verschiedene existierende Lösungsansätze vorgestellt und im Hinblick auf die Nutzbarkeit zur Realisierung einer kontextadaptiven Anwendungsarchitektur untersucht wurden, erfolgte in diesem Kapitel die Auswahl entsprechender Lösungsalternativen mit dem Ziel der Entwicklung eines eigenen Lösungsansatzes.

Vor diesem Hintergrund wurden zunächst eine Basisarchitektur entwickelt, welche die in Abschnitt 4.2 ermittelten Anforderungen an eine kontextadaptive Anwendungsarchitektur prinzipiell erfüllt. Anhand unterschiedlicher Anwendungsbeispiele wurde anschließend untersucht, welche unterschiedlichen Formen der Adaption sich im Hinblick auf das mobile Cloud Computing generell anbieten, um den Restriktionen mobiler Geräte entgegenzuwirken. Anschließend wurde herausgearbeitet, wie sich diese Adaptionsformen auf unterschiedliche konkrete Adaptionsziele abbilden lassen. In diesem Zusammenhang wurde ebenso untersucht, welche Ebene und Granularität der Adaption sich zur Entwicklung von Anwendungen für das mobile Cloud Computing eignen.

Aus den verschiedenen Kategorien existierender Lösungsansätze konnten zwei Gruppen identifiziert werden, die sich prinzipiell gut für diese Problemstellung eignen. Die erste Gruppe nutzt als Ebene der Adaption das Konzept der Virtualisierung, welche entweder auf Anwendungs- oder auf Systemebene erfolgt und bei der entweder Threads [CIM+11] oder Methodenaufrufe [CBC+10] als Granularität der Adaption verwendet werden, um auch nichtmodifizierte mobile Anwendungen innerhalb einer Infrastruktur für mobiles Cloud Computing zu verteilen. Aufgrund der Nachteile dieser Ebene der Adaption in Bezug auf die zu erwartenden hohen Synchronisierungsaufwände zwischen einem mobilem Gerät und seiner Infrastruktur, wurde dieser Ansatz unter Berücksichtigung der in dieser Arbeit formulierten Ziele der Adaption nicht weiter verfolgt. Entsprechend wurde die zweite Kategorie von Lösungsansätzen, die eines komponentenbasierten Entwurfes einer mobilen Anwendung, gewählt, einerseits um das Expertenwissen der Entwickler im Hinblick auf eine sinnvolle Partitionierung der Anwendung zu berücksichtigen und andererseits um hierdurch eine möglichst lose Kopplung der einzelnen Komponenten einer mobilen Anwendung zu erreichen und damit durch geringere zu erwartende Synchronisationsaufwände der wechselnden Konnektivität mobiler Umgebungen gerecht zu werden.

Anschließend erfolgte der Entwurf eines eigenen Ansatzes, basierend auf der in Abschnitt 5.1 entwickelten Basisarchitektur. Hierzu wurde zunächst das Konzept der taskbasierten Adaption eingeführt, um die einzelnen Ziele und Dimensionen der Adaption durch die entwickelte Architektur angemessen zu erfüllen. Anschließend wurde der komponentenbasierte Entwurf dahingehend konkretisiert, dass in sich abgeschlossene (self-contained) Container-

Komponenten zur Komposition von Anwendungen definiert wurden, um anschließend die übrigen Infrastrukturkomponenten einer Basisarchitektur zu konzipieren.

Zur Berücksichtigung der in Abschnitt 4.2 erwähnten Anforderung, flexibel auf den aktuellen und zukünftigen Ausführungskontext reagieren zu können, wurde anschließend das Konzept der generischen Kontextadaption eingeführt, welches den im Rahmen der Basisarchitektur vorgestellten Kontextmanager in die Lage versetzt, eine breite Zahl unterschiedlicher Mobilitätsszenarien zu unterstützen. In diesem Zusammenhang wurde ein möglichst generischer Prozess zur Prognose unterschiedlicher Kontextattribute entwickelt, um eine breite Menge an Situationen und Geräten zu unterstützen und dabei sowohl mit unterschiedlichen verfügbaren Kontextattributen als auch schlechter und wechselnder Datenqualität umzugehen.

Weiterhin wurden wesentliche Teilprobleme im Hinblick auf die Heterogenität mobiler Clouds, der Suche und Integration neuer Ressourcen, der Analyse des Laufzeitverhaltens und der Partitionierung mobiler Anwendungen untersucht und entsprechende Lösungsansätze herausgearbeitet. Für Teilprobleme wie beispielsweise die Partitionierung und die Dienstsuche konnten bestehende Lösungsansätze identifiziert werden, die im Rahmen des Konzepts integriert werden konnten. Gleichzeitig wurden für die kontextabhängige Verarbeitung von Tasks eigene Konzepte vorgeschlagen, beispielsweise der Kontextmanager und die Adaptionsstrategien für die taskbasierte Kooperation mit der Infrastruktur.

Nachdem nun im vorliegenden Kapitel die in Abschnitt 1.2 formulierten Forschungsfragen auf konzeptioneller Ebene untersucht wurden, soll im nun folgenden Kapitel 6 die Untersuchung der praktischen Umsetzbarkeit des entwickelten Konzepts in Form einer prototypischen Implementierung vorgenommen werden.

6. Entwurf und Implementierung der entwickelten Architektur

Nachdem im vorhergehenden Kapitel das Konzept einer kontextadaptiven Anwendungsarchitektur entwickelt wurde, soll im Folgenden der konkrete systembezogene Entwurf erarbeitet und die Implementierung einer konkreten Architektur umgesetzt werden.

Hierzu erfolgt zunächst die Auswahl eines konkreten Software-Entwicklungsparadigmas, um anschließend eine hierzu passende Ausführungsumgebung auszuwählen. Anschließend wird die exemplarische Integration der ausgewählten Ausführungsumgebung in ein aktuelles Betriebssystem für mobile Geräte beschrieben. In einem weiteren Schritt schließt sich die Umsetzung der Konzepte der entwickelten kontextadaptiven Anwendungsarchitektur in Form der prototypischen Realisierung der einzelnen Infrastrukturkomponenten an.

6.1. Softwareentwicklungsparadigmen

Für die Entwicklung verteilter Systeme haben sich in den letzten Jahrzehnten eine Reihe von Softwareentwicklungsparadigmen herausgebildet. Die Nutzung dieser Paradigmen im Hinblick auf die Entwicklung mobiler verteilter Systeme bringt jedoch eine Reihe zusätzlicher Herausforderungen mit sich, die sich im Wesentlichen der hohen Dynamik und dem schnell wechselnden Kontext mobiler Clouds zurechnen lassen und die bestehenden Paradigmen oft an die Grenze ihrer Beherrschbarkeit bringen.

Entsprechend bedarf es bei der Entwicklung mobiler verteilter Systeme nicht nur der Berücksichtigung der üblichen nichtfunktionalen Aspekte stationärer verteilter Systeme wie der Fähigkeit zur nebenläufigen oder parallelen Verarbeitung, der Skalierbarkeit, der Robustheit und der Sicherheit, wie sie in [BP11] definiert werden. Sondern es gilt gleichzeitig erweiterte Mechanismen in Bezug auf die Dienstsuche, Koordination und Fehlerbehandlung, die die häufig wechselnde Topologie dieser Ad-hoc-Netzwerke erforderlich macht, zu berücksichtigen. In diesem Zusammenhang sollen im Folgenden eine Reihe prominenter Entwicklungsparadigmen für verteilte Systeme vorgestellt werden, um im Anschluss ihre Eignung im Hinblick auf die Umsetzung der kontextadaptiven Anwendungsarchitektur zu bewerten.

Objektorientierte Softwareentwicklung Das Paradigma der Objektorientierung stellt seit langer Zeit das bekannteste Paradigma in der Entwicklung von Softwaresystemen dar. In der objektorientierten Programmierung stellen

Objekte die Entität erster Ordnung (first order entity) dar, deren interner Zustand mithilfe lesender oder schreibender Methoden beobachtet bzw. manipuliert werden kann. Auf diese Weise verbirgt ein Objekt seinen internen Zustand und sein internes Verhalten vor anderen Objekten und bietet lediglich über eine definierte Schnittstelle Methoden zur Beobachtung oder Manipulation an [SGM02]. Die nebenläufige Verarbeitung innerhalb objektorientierter Softwaresysteme wird durch eine Aufspaltung des Kontrollflusses in Threads realisiert, welche anschließend ihren jeweils eigenen Kontrollfluss besitzen. Dies macht es erforderlich, dass der Entwickler den Zugriff auf gemeinsam genutzte Ressourcen synchronisiert und gegebenenfalls die verschiedenen Kontrollflüsse wieder zusammenführt.

Im Hinblick auf die verteilte Ausführung einer objektorientierten Anwendung existieren Konzepte wie der entfernte Methodenaufruf (remote method invocation), welcher es erlaubt, Methoden auch auf Objekten aufzurufen, die sich auf entfernten Knoten eines verteilten Systems befinden. Das Auffinden der entsprechenden Objekte erfolgt dabei über einen Namensdienst, in der Programmiersprache Java beispielsweise über die sogenannte *RMI Registry* [PM01]. Die anschließende Interaktion wird dabei, für den Entwickler transparent, durch einen sogenannten *Stub* auf Client-Seite abgebildet. Kommt es bei diesem entfernten Aufruf zu Fehlern, müssen diese jedoch durch den Entwickler abgefangen und entsprechend behandelt werden.

Komponentenorientierte Softwareentwicklung Der grundlegende Gedanke der komponentenbasierten Softwareentwicklung stellt die Komposition von Anwendungen mithilfe wiederverwendbarer Komponenten dar [SGM02]. Im Gegensatz zu einem Objekt zeichnet sich eine Komponente entsprechend dadurch aus, dass sie stärker von einem Gegenstandsbereich abstrahiert [SGM02]. Komponenten dienen damit nicht primär dazu, reale Weltentitäten abzubilden, sondern sollen den Entwickler dazu motivieren, einzelne Funktionalitäten mit hoher Abhängigkeit zueinander gemeinsam und wiederverwendbar in Form einer Komponente bereitzustellen. Ebenso wie Objekte kapseln Komponenten dabei ihren internen Zustand über Schnittstellen, mithilfe derer sie ihre Funktionalität anbieten. Die Konstruktion komponentenbasierter Software geschieht dabei durch die Komposition der vorgefertigten Komponenten durch den Entwickler. Ihre Ausführung basiert auf dem Prinzip der Umkehrung des Kontrollflusses, bei dem die Infrastruktur den Kontrollfluss koordiniert und dabei die entsprechenden Komponenten aufruft [PB13].

Die komponentenbasierte Softwareentwicklung bildet im Hinblick auf die verteilte Ausführung keine explizite Synchronisation einzelner Komponenten zueinander ab und überlässt dies ebenfalls der Ausführung der genutzten Ausführungsumgebung. Die Eignung zur Nutzung einer komponentenbasierten Anwendung ergibt sich dabei im Wesentlichen durch die verwendete Ausführungsumgebung. Entsprechende Ausführungsumgebungen wie die *IBIS Middleware* [VNMW+05] sind auf eine Nutzung innerhalb verteilter Systeme vorbereitet und bieten entsprechende Mechanismen zur Fehlerbehandlung an.

Dienstorientierte Architekturen Dienstorientierte Architekturen stellen ein Architekturmuster dar, welches explizit auf die Nutzung innerhalb verteilter Systeme ausgerichtet ist. Ihre Strukturierung orientiert sich oft an Geschäftsprozessen, die eine logische Abfolge von Aktivitäten darstellen und bei denen die einzelnen Aktivitäten häufig durch einen Dienst realisiert sind [HS05]. Ähnlich wie Komponenten stellen Dienste ihre Funktionalität dabei über eine definierte Schnittstelle bereit.

In Bezug auf die Verteilung kann bei dienstorientierten Architekturen davon ausgegangen werden, dass die einzelnen Dienste üblicherweise durch eine entsprechende Infrastruktur in einem Netzwerk bereitgestellt werden. Ein wichtiges Prinzip der Dienstorientierung ist die dynamische Suche der für die Ausführung eines Geschäftsprozesses benötigten Dienste in einem entsprechenden Dienstverzeichnis.

Aktive Komponenten Aktive Komponenten stellen eine Kombination verschiedener Entwicklungsparadigmen für verteilte Systeme dar und kombinieren die Konzepte der Objektorientierung mit den Konzepten der Komponenten-, Dienst- und Agentenorientierung [BP11]. Je nach Anwendungsbereich und Auswahl der internen Architektur stellen aktive Komponenten entweder rein passiv Dienste über ihre Schnittstelle bereit oder bilden Proaktivität im Rahmen einer Agentenorientierung ab, wie in Abbildung 6.1 dargestellt.

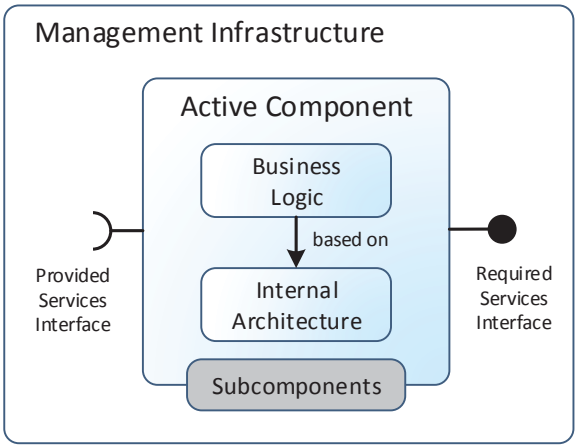

Abbildung 6.1.: Abbildung einer aktiven Komponente, nach (BP11)

Durch ihre Konzeption im Hinblick auf den Einsatz innerhalb verteilter Systeme stellen aktive Komponenten eine Reihe von Mechanismen im Hinblick auf die asynchrone Verarbeitung von Dienstaufrufen mithilfe des Aktormodells (actor model) [HBS73] bereit. Diese entlasten den Entwickler von den bei der Objektorientierung üblichen Aufgaben der Synchronisation nebenläufiger Kontrollflüsse und erlauben gleichzeitig eine nicht-blockierende Aufrufsemantik durch Nutzung von Platzhaltern, sogenannten *Futures*, für noch nicht bekannte Ergebnisse [BP11].

Auswahl eines geeigneten Entwicklungsparadigmas Obwohl sich alle vorgestellten Software-Entwicklungsparadigmen sich prinzipiell für die Nutzung innerhalb verteilter Systeme eignen, so bestehen doch gleichzeitig spezifische Restriktionen, welche ihren Einsatz im Bereich mobiler verteilter Systeme erschweren oder verhindern.

Wie in Abbildung 6.2 veranschaulicht bieten sowohl die Objektorientierung als auch die Dienstorientierung Konzepte zum verteilungstransparenten Aufruf über Systemgrenzen hinweg. Wie in [BP11] gezeigt wird, werden von diesen Paradigmen allerdings oft keine adäquaten Konzepte für den Umgang mit Nebenläufigkeit bereitgestellt. Zur Realisierung nicht-funktionaler Kriterien wie der Berücksichtigung von Adaptionszielen einer kontextadaptiven Anwendungsarchitektur ergeben sich zusätzliche Anforderungen in Bezug auf die Unterstützung querschnittlicher Belange (cross-cutting concern), die wiederum lediglich vom dienst- und komponentenorientierten Paradigma angemessen erfüllt werden [BP11].

Das Entwicklungsparadigma der aktiven Komponenten unterstützt als einziges sowohl einen Großteil der erhobenen Anforderungen als auch Konzepte für den Umgang mit Verteilung und Nebenläufigkeit, um die Unterstützung nicht-funktionaler Anforderungen beherrschbar zu machen. So ist es beispielsweise zur Umsetzung der konzipierten Koordinationsprozesse erforderlich, dass das genutzte Entwicklungsparadigma nichtfunktionale Kriterien wie Annotationen und Mechanismen zur Trennung von Geschäfts- und Adaptionslogik (vergleiche Abschnitt 3.3.2) bietet. Eine Anforderung, die innerhalb des Entwicklungsparadigmas der aktiven Komponenten durch die Umsetzung des Entwurfsmusters des *Interceptors*[1] abgebildet wird. Entsprechend sollen aktive Komponenten zur Realisierung einer prototypischen Implementierung herangezogen werden.

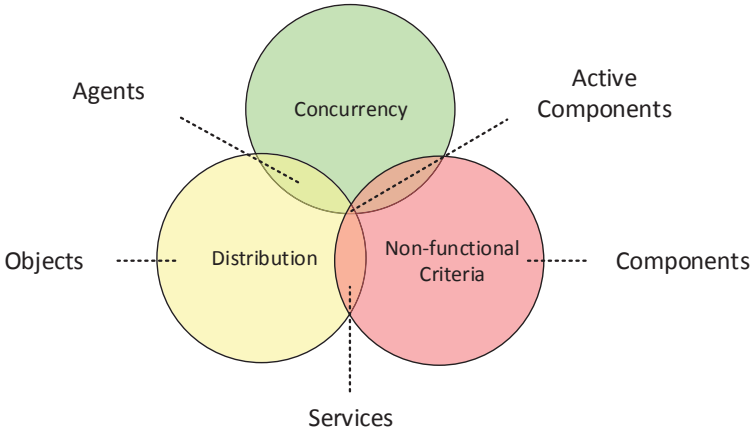

Abbildung 6.2.: Abdeckung der Anforderungen durch aktive Komponenten, nach (BP11)

[1] Ein Interceptor ist ein Entwurfsmuster, welches beispielsweise zur Erweiterung eines Frameworks genutzt werden kann, ohne das Framework selbst dabei anzupassen [SSRB13].

6.2. Auswahl einer Ausführungsumgebung

Um einen Entwurf auf Basis aktiver Komponenten umsetzen zu können, ist eine entsprechende Ausführungsumgebung nötig. Um eine robuste und skalierbare Infrastruktur bereitzustellen, mithilfe welcher die prototypische Realisierung einer kontextadaptiven Anwendungsarchitektur umgesetzt werden kann, bietet sich die Referenzimplementierung der Ausführungsumgebung für aktive Komponenten an, das *Jadex Active Components Framework* [PB13]. Diese Auswahl bringt den Vorteil mit sich, dass in der Jadex-Middleware sowohl bestehende Funktionalität in Bezug auf die Verteilung und Suche von Diensten als auch die Möglichkeit zur Fehlerbehandlung und zum Debugging genutzt werden können.

Hinsichtlich der Verteilung der Komponenten einer mobilen Anwendung unterstützt die Jadex-Middleware das Aufspannen eines Overlay-Netzwerks oberhalb der genutzten Schnittstellen zu Mobilkommunikation, die eine Erreichbarkeit der verschiedenen Instanzen der Jadex-Middleware, den sogenannten Plattformen, innerhalb eines verteilten Systems sicherstellt. Hierbei können durch sogenannte Relay-Server auch Verbindungen zwischen zwei, durch *Firewalls* oder *Network Address Translation*, isolierten und nicht direkt erreichbaren Plattformen realisiert werden.

Darüber hinaus unterstützt die Jadex-Middleware mithilfe der sogenannten *Awareness*-Funktionalität die Erkennung und Einbindung weiterer Plattformen über die unterschiedlichen Schnittstellen zur Mobilkommunikation, um so die Grundlage für eine spontane Interaktion innerhalb von Ad-hoc-Netzwerken zu realisieren. Sämtliche Kommunikation erfolgt dabei verschlüsselt, was den in Abschnitt 4.2 formulierten Anforderungen im Hinblick auf die Informationssicherheit Rechnung trägt. Im Hinblick auf die geforderte Effizienz der Kommunikation und Ressourcennutzung bietet Jadex darüber hinaus die Möglichkeit zum asynchronen Nachrichtenaustausch zwischen aktiven Komponenten und setzt hierfür auf eine effiziente Binärkodierung der Nachrichten [JL13].

Weiterhin erlaubt es die im Framework enthaltene Funktionalität zur ereignisbasierten Simulation, die prototypische Umsetzung der entwickelten Konzepte direkt für eine Evaluation nutzbar zu machen. Hierdurch kann für die Evaluation der prototypischen Implementierung der entwickelten Konzepte und für die anschließende Simulation unterschiedlicher Nutzungsmuster auf dieselbe Implementierung zugegriffen werden, wodurch eine weitgehende Übertragbarkeit der Ergebnisse gewährleistet wird.

Im Hinblick auf die Entwicklerunterstützung soll an dieser Stelle darauf hingewiesen werden, dass die Möglichkeit, Abläufe nicht in Realzeit, sondern beschleunigt ablaufen zu lassen, ebenso dazu genutzt werden kann, die Entwicklung konkreter mobiler Anwendungen dahingehend zu unterstützen, dass deren Nutzung innerhalb unterschiedlicher realer Nutzungsszenarien effizient simuliert werden kann. Hierdurch kann beispielsweise bereits früh im Entwicklungsprozess untersucht werden, ob eine mobile Anwendung in der Lage

ist, sinnvoll mit ihrer Infrastruktur zu kooperieren. Ebenso kann die Simulation wechselnder Kontextzustände dazu genutzt werden, im Entwicklungsprozess Mechanismen zur Fehlerbehandlung zu testen.

Entsprechend stellt die Jadex-Middleware die Umgebung dar, in der sowohl die Infrastrukturkomponenten einer kontextadaptiven Anwendungsarchitektur als auch die komponentenbasierten mobilen Anwendungen selbst realisiert werden sollen, wie in Abbildung 6.3 gezeigt wird.

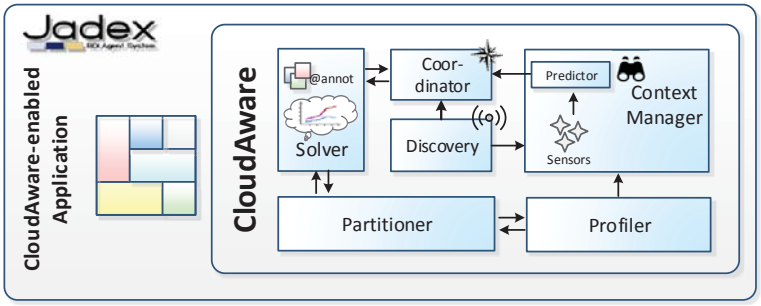

Abbildung 6.3.: Realisierung der kontextadaptiven Anwendungsarchitektur auf Basis der Jadex-Middleware

Die Voraussetzung für die Lauffähigkeit der Jadex-Middleware beschränkt sich dabei auf das Vorhandensein einer Java Virtual Machine. Inwieweit die Integration dieser Ausführungsumgebung in ein repräsentatives Betriebssystem für mobile Geräte möglich ist, soll im Folgenden untersucht werden.

6.3. Integration in eine Systemplattform für mobile Geräte

Zur Integration der Implementierung der gewählten Ausführungsumgebung in eine Systemplattform für mobile Geräte ist es erforderlich, dass diese einerseits die direkte Ausführung von Programmcode unterstützt, ohne dass dieser durch den Hersteller der Systemplattform signiert und freigegeben werden muss. Andererseits soll sie möglichst quelloffen sein, um sinnvoll eine Erweiterung durchführen zu können.

Im Abgleich mit dem in Abschnitt 2.2.4 durchgeführten Abgleich verfügbarer Mobilplattformen bietet sich entsprechend die von der *Open Handset Alliance* für mobile Geräte entwickelte Systemplattform Android an. Diese stellt momentan mit einem Marktanteil von 86,2 Prozent das am weitesten verbreitete Betriebssystem auf Smartphones dar (vergleiche Abschnitt 2.2.4). Zusätzlich handelt es sich bei der Android-Systemplattform um freie Software, die quelloffen unter der *Apache Software License* entwickelt wird und sich damit gut zur Entwicklung und Erweiterung anbietet [Cin12]. Im Hinblick auf die Entwicklung kontextadaptiver Anwendungen und die Integration der Jadex-Middleware sollen im Folgenden wesentliche Aspekte der Android-Systemarchitektur vorgestellt werden.

6.3.1. Systemarchitektur

Die Android-Systemarchitektur, dargestellt in Abbildung 6.4, setzt als unterste Schicht auf dem Linux-Kernel auf, der von der geräteabhängigen Hardware mobiler Geräte mithilfe entsprechender *Gerätetreiber* abstrahiert. Auf der darüberliegenden Ebene stellt die Android-Systemarchitektur eine Reihe von Bibliotheken bereit, unter anderem für die Verschlüsselung (*SSL*), den Datenbankzugriff (*SQLite*) und die Anzeige von Webseiten (*WebKit*).

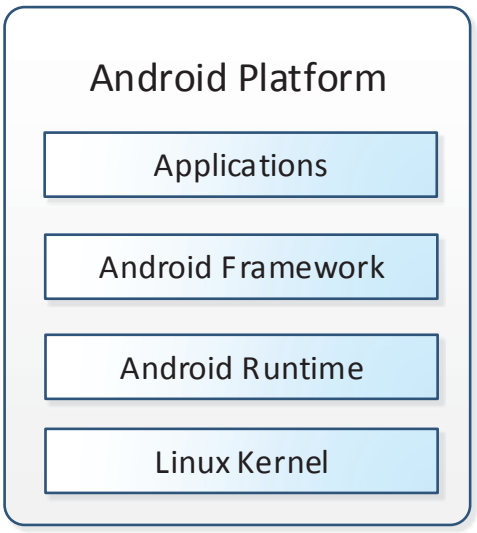

Abbildung 6.4.: Komponenten der Android-Systemplattform (Android 4.4), nach (Cin12)

Ebenfalls auf dieser Ebene eingeordnet befindet sich die Android-Laufzeitumgebung zur Ausführung mobiler Anwendungen, die *Dalvik Virtual Machine* (DVM). Die DVM ist in der Lage, Bytecode, der aus Quellcode der Programmiersprache Java übersetzt wurde, auszuführen. Dieser wird ähnlich wie zur Ausführung innerhalb der Java Virtual Machine beim Aufruf interpretiert oder kompiliert [2]. Im Gegensatz zur Java Virtual Machine ist die DVM allerdings, passend zu der in mobilen Geräten verwendeten ARM-Mikroarchitektur, als Registermaschine ausgelegt, um eine ressourcenschonende und performante Ausführung mobiler Anwendungen zu ermöglichen [Cin12]. Zum Start einer mobilen Anwendung wird dabei jeweils eine eigene Instanz der DVM verwendet, wodurch die einzelnen mobilen Anwendungen voneinander isoliert ausgeführt werden (*sandboxing*) – eine Eigenschaft, die es im Hinblick auf die Kommunikation mobiler Anwendungen untereinander zu berücksichtigen gilt.

In den beiden darüberliegenden Schichten werden von der Android-Systemarchitektur einzelne Anwendungen, zum Beispiel ein Internet-Browser oder Dienste in Form des sogenannten *Android-Frameworks*, bereitgestellt, welches von Anwendungsentwicklern zu nutzen ist, um beispielsweise in den

[2] In Version 5.0 der Android-Systemplattform kommt zu diesem Zweck die Android Runtime zum Einsatz. Bei dieser wird nicht länger auf Just-in-time-Kompilierung gesetzt, sondern eine mobile Anwendungen während der Installation einmalig in nativen Binärcode übersetzt. Hierdurch soll der Start einer mobilen Anwendung beschleunigt werden.

von ihnen entwickelten Anwendungen den aktuellen Ort des mobilen Geräts durch die jeweiligen gerätespezifischen Lokalisationsverfahren zu bestimmen (Location-Manager) oder abzufragen oder zwischen den verschiedenen, dem Nutzer angezeigten Bildschirmelementen zu wechseln (Activity-Manager). Hierdurch definieren die Dienste auf der Ebene des Android-Frameworks auch die Struktur, in der Android-Anwendungen entwickelt werden, auf die im folgenden Abschnitt näher eingegangen werden soll.

6.3.2. Struktur von Android-Anwendungen

Die prototypische Umsetzung der entwickelten Konzepte mithilfe aktiver Komponenten (vergleiche Abschnitt 6.1) zusammen mit der Anforderung zur Trennung von Geschäfts- und Adaptionslogik legen den Einsatz einer objektorientierten Programmiersprache nahe, die die Abbildung nicht-funktionaler Eigenschaften unterstützt. Abgeleitet aus den Anforderungen zur einfachen Wartbarkeit und Portierbarkeit sollte diese Sprache möglichst einfach zu nutzen sein und sich gut zur Entwicklung mobiler Anwendungen auf Basis der Android-Systemplattform eignen. In diesem Zusammenhang bietet sich der Einsatz der Programmiersprache Java und die Nutzung von Java-Annotationen zur Abbildung nicht-funktionaler Eigenschaften an.

Der Sprachumfang umfasst dabei einen Großteil der Bibliotheken der Java Standard Edition (Java SE), allerdings wurden einige der üblicherweise verfügbaren Bibliotheken durch Android-spezifische APIs ersetzt oder erweitert [Cin12]. Aufgrund der im vorhergehenden Abschnitt beschriebenen Bereitstellung anwendungsnaher Dienste und Schnittstellen durch das Android-Framework gliedern sich vom Entwickler erstellte Anwendungen üblicherweise in vier Arten von Bausteinen, welche in Anlehnung an [All16] im Folgenden beschrieben werden:

- Activities: *Activities* bilden die grafische Benutzeroberfläche einer Android-Anwendung ab und sind für die Interaktion mit dem Nutzer zuständig. Eine einzelne Activity beschreibt dabei das Layout einer grafischen Benutzeroberfläche. Sie besitzt einen festgelegten Lebenszyklus, der vom Application Framework vorgegeben wird (vergleiche Abbildung 6.5).

  Zur Beeinflussung dieses Lebenszyklusses stehen eine Reihe von Lebenszyklusmethoden bereit, die benötigt werden, falls der Nutzer einer mobilen Anwendung zwischen verschiedenen Activities wechselt oder Anwendungen aufgrund von Speicherknappheit durch das Android-Betriebssystem beendet werden müssen.

- Services: *Services* dienen einer Android-Anwendung zur Abbildung von Hintergrundaufgaben, entsprechend können sie im Gegensatz zu Activities auch im Hintergrund aktiv bleiben, um beispielsweise Berechnungen durchzuführen, Daten zu verarbeiten oder diese über Netzwerkschnittstellen mit der Infrastruktur auszutauschen. Sie besitzen keine

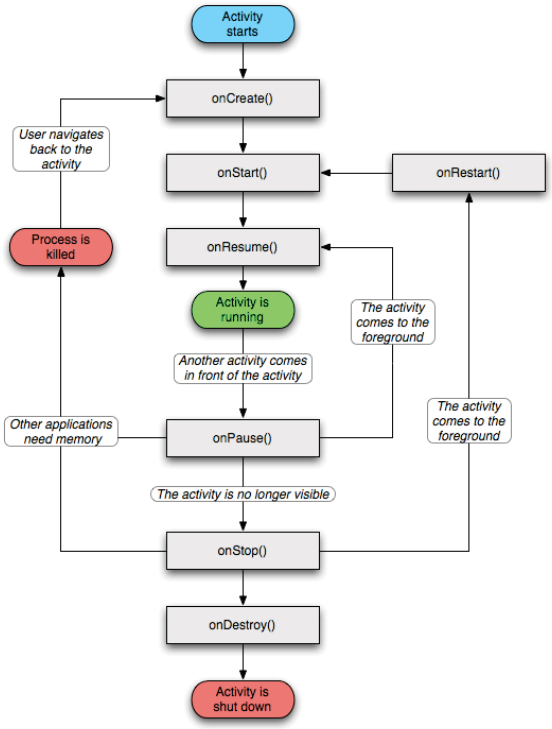

Abbildung 6.5.: Lebenszyklus von Android-Activities, entnommen aus (All16)

eigene Benutzeroberfläche, sondern stellen ihre Funktionalität über eine Schnittstelle bereit.

- Content Provider: *Content Provider* stellen einer oder mehreren Anwendungen Daten bereit, die entweder im Dateisystem, einer Datenbank oder in der Infrastruktur persistent abgelegt sind.
- Broadcast Receiver: Sie erlauben der Android-Anwendung auf äußere Ereignisse zu reagieren, die von der Android-Systemplattform bereitgestellt werden. Diese können das Ergebnis eines veränderten Kontextes sein, zum Beispiel die Information, dass der Energievorrat des mobilen Gerätes nahezu erschöpft ist.

Mit Blick auf die in Abschnitt 5.2.3 beschriebenen Formen und Ziele der Adaption mobiler Anwendungen zeigt sich, dass sich nicht jeder dieser Bausteintypen zur Integration der Konzepte einer kontextadaptiven Anwendungsarchitektur anbietet.

Würden GUI-Bausteine wie die Activities im Rahmen der unterschiedlichen Adaptionsformen ausgelagert oder verzögert werden, so könnte nicht sichergestellt werden, dass der Nutzer einer entsprechenden mobilen Anwendung keine Einschränkung der Dienstqualität in Form einer verzögert oder nicht reagierenden Benutzeroberfläche erfährt. Dies kann beispielsweise bei den Adaptionsformen der Auslagerung im Falle eines Verbindungsabbruchs zur Infrastruktur auftreten und dazu führen, dass die Ausführung der Anwendung

nicht fortgesetzt werden kann. Ebenso problematisch ist die Auslagerung eines Content Providers, da dieser auf geräte-spezifische Daten und Funktionalität zugreift[3].

Letztlich bietet sich damit lediglich der Baustein-Typ der Services zur Erweiterung mobiler Anwendungen im Hinblick auf die Realisierung einer kontextadaptiven Anwendungsarchitektur an. Wie die Integration der Konzepte abgebildet wird, soll im nun folgenden Abschnitt beschrieben werden.

6.3.3. Einbindung der kontextadaptiven Anwendungsarchitektur

Durch die strikte Trennung zwischen der Nutzerinteraktion (Android Activities) und der Hintergrundverarbeitung (Android-Services) von Android-Anwendungen ist es vergleichsweise einfach, eine im Rahmen der unterschiedlichen Adaptionsformen entwickelte Anpassungsstrategie einzubinden, ohne dass hierbei der Bedarf an einer grundlegend anderen Architektur der mobilen Anwendungen auftritt oder die Dienstqualität aus Nutzerwahrnehmung dadurch beeinträchtigt wird, dass die Nutzerinteraktion durch zusätzliche Geschäftslogik verlangsamt wird.

Entsprechend soll eine Erweiterung der Funktionalität einzelner Android-Services geschaffen werden, die es ermöglicht, die Hintergrundverarbeitung einer mobilen Anwendung entlang der unterschiedlichen Dimensionen der Adaption anpassen zu können. Hierzu bedarf es zweier Dinge, und zwar der Integration der ausgewählten Laufzeitumgebung sowie der Auslagerung der Geschäftslogik in einen komponentenbasierten Entwurf:

Integration der Ausführungsumgebung Zur Integration der Jadex-Middleware in die Android-Systemplattform kommt die entsprechende Portierung *Jadex Android* [Kal14, Act16] zum Einsatz, welche, wie in Abbildung 6.6 veranschaulicht wird, innerhalb eines eigenen Android-Services gestartet wird.

Innerhalb dieser Plattform werden beim Start einer mobilen Anwendung zunächst die zusätzlich erforderlichen Infrastrukturkomponenten wie unter anderem der Koordinator und der Kontextmanager bereitgestellt.

Schnittstelle zwischen Android und Komponenten Um die für die Hintergrundverarbeitung innerhalb eines Android-Services befindliche Geschäftslogik kontextadaptiv ausführen zu können, gilt es, diese innerhalb eines komponentenbasierten Entwurfes zu strukturieren. Hierzu gilt es, die von der Android-Plattform bereitgestellten Schnittstellen zu nutzen. Hierdurch kann auf die von den Komponenten angebotenen Dienste zugegriffen werden, wie dies im folgenden Code-Beispiel aus einer Beispielanwendung veranschaulicht wird:

[3] Im Rahmen des vorgestellten Komponentenmodells wurde für den Fall des Zugriffs auf gerätespezifische Ressourcen der Komponententyp des Data Access Objects vorgesehen.

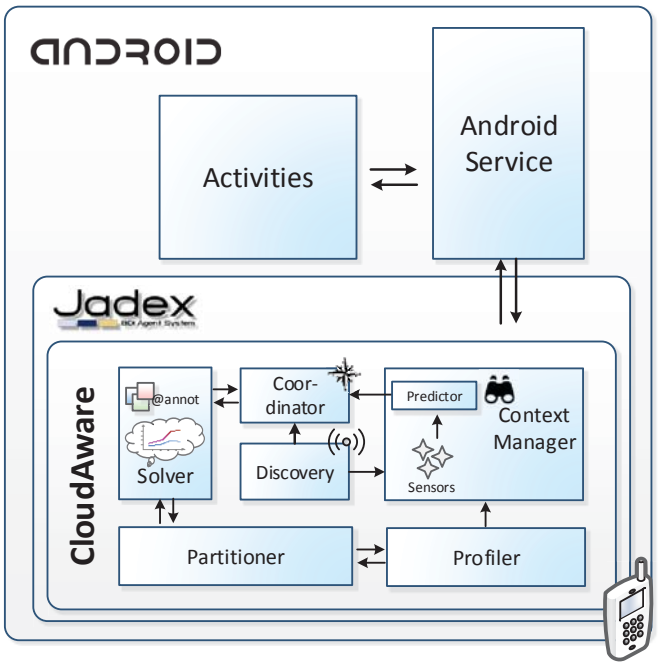

Abbildung 6.6.: Einbindung der Jadex-Android-Plattform auf einem mobilen Gerät

```
public class AndroidFaceDetectionService extends Service {
  ...
public void onStart(Intent intent, int startid) {
  BufferedImage image;
  Bitmap bmp = (Bitmap) intent.getExtras().get("bitmap");
  ... //conversion to BufferedImage
  IFaceDetService fds = getService(IFaceDetService.class).get();
  fds.detectFace(image).addResultListener(
    new IResultListener<Rectangle[]>(){
    ...
    public void resultAvailable(Rectangle[] result){
    //handle the result, when face detection is complete
    }});
  ...
}}
```

Listing 6.1: Implementierung eines Android-Services

Wie die konkrete Nutzung der hierfür bereitgestellten Komponententypen abläuft, wird im Zusammenhang der prototypischen Realisierung der einzelnen Komponenten noch näher beschrieben.

Das gewählte Vorgehen erlaubt es somit dem Entwickler einer mobilen Anwendung, sich in Bezug auf die Implementierung der grafischen Benutzeroberfläche der kompletten Funktionalität des Android-Framework bedienen zu können und an dieser Stelle keinen zusätzlichen weiteren Restriktionen unterworfen zu sein. Gleichzeitig beschränken sich die Änderungen in der Imple-

mentierung einer bestehenden mobilen Anwendung auf den Bausteintyp der Android-Services, welche bei einer bestehenden Anwendung lediglich um den Aufruf der durch die Komponenten bereitgestellten Services zu erweitern sind, wodurch die einfache Portierbarkeit und Wartbarkeit entsprechender Anwendungen sichergestellt werden soll.

Dieser Ansatz erlaubt ebenso die im Rahmen der Anforderungsanalyse geforderte Koexistenz zwischen regulären Android-Anwendungen und solchen, die die Erweiterung einer kontextadaptiven Anwendungsarchitektur nutzen, da er die bestehenden Mechanismen der Android-Plattform in Hinblick auf die Ressourcenteilung und das Power-Management nicht außer Kraft setzt. Zusätzlich macht dieser Ansatz keine Modifikation der Android-Plattform selbst erforderlich, wodurch andernfalls die geforderte Portierbarkeit der Implementierung auf einer breiten Anzahl mobiler Geräte eingeschränkt sein könnte.

6.4. Prototypische Realisierung

Nachdem die einzelnen Bestandteile einer kontextadaptiven Anwendungsarchitektur in Abschnitt 5.4 bereits konzeptionell vorgestellt wurden, soll auf Basis des zur Implementierung ausgewählten Entwicklungsparadigmas im Folgenden eine Umsetzung der wesentlichen, im Rahmen dieser Arbeit entwickelten Konzepte des Komponentenmodells und der einzelnen Infrastrukturkomponenten einer kontextadaptiven Anwendungsarchitektur erfolgen. Hierbei wird jeweils kurz auf die konkret in der prototypischen Implementierung verwendeten Technologien eingegangen.

6.4.1. Komponentenmodell

Im Rahmen der Konzeption einer kontextadaptiven Anwendungsarchitektur wurde ein Komponentenmodell vorgeschlagen, für welches drei unterschiedliche Komponententypen definiert wurden, die zur Implementierung der zu adaptierenden Geschäftslogik (Stateful und Stateless) oder zum Zugriff auf lokale Ressourcen (Data Access Object) genutzt werden können. Die konkrete Implementierung einer Komponente geschieht dabei zweistufig. So gilt es zunächst, den oder die von der Komponente bereitgestellten Dienste zu implementieren, wie das folgende Code-Beispiel 6.2 veranschaulicht:

```
@ProvidedServices({@ProvidedService(name="FaceDetService",
    type=IFaceDetService.class)})
@Service
public class FaceDetService implements IFaceDetService {
  public Future<Rectangle[]> detectFace(BufferedImage image) {
... //detect the face(s)
   Future<Rectangle[]> returnval = new Future<Rectangle[]>();
   returnval.setResult(fds_result);
   return returnval; }}
```

Listing 6.2: Implementierung eines entsprechenden Dienstes

Anschließend gilt es, den oder die entwickelten Services mithilfe des von der Jadex-Middleware bereitgestellten Komponententyps des *Mikroagenten* zu veröffentlichen und hierbei einen von der kontextadaptiven Anwendungsarchitektur bereitgestellten Interceptor zu verwenden, welcher die Schnittstelle zur Koordinations- und zur Profiling-Komponente der kontextadaptiven Anwendungsarchitektur bereitstellt. An dieser Stelle kann ebenso der eingangs erwähnte Komponententyp festgelegt werden, was durch die Nutzung einer entsprechenden Annotation ermöglicht wird (siehe Code-Beispiel 6.2, oberhalb der Klassendefinition).

```
@Agent
@RequiredServices({
  @RequiredService(name = "sharpenService", type =
      IFaceDetService.class,
       binding = @Binding(scope =
          RequiredServiceInfo.SCOPE_PLATFORM))})
@ProvidedServices({
  @ProvidedService(name="blurservice",
           type=IFaceDetService.class,
           implementation=@Implementation(value=FaceDetService.class,
           interceptors=@Value(clazz=GenericInterceptor.class)))
})
@CloudAwareStateless
public class LocalWorkerAgent extends CloudAwareWorkerAgent
{ ... }
```

Listing 6.3: Bereitstellung der entwickelten Services

Für den Fall, dass nicht auf allen zur Nutzung vorgesehenen Surrogates dieselbe Menge an Services angeboten werden soll, steht dabei die Möglichkeit, unterschiedliche Komponenten zu definieren, welche die jeweils gewünschte Kombination an Services bereitstellen oder bei der Bereitstellung gerätespezifische Implementierungen verwenden.

Zusätzlich wurde eine Reihe von Annotationen implementiert, die genutzt werden können, um die Adaptionsentscheidung der Koordinations-Komponente zu beeinflussen:

- Die *local*-Annotation für Komponenten: Sie bewirkt, dass Komponenten auf dem mobilen Gerät verbleiben. Dies kann dazu genutzt werden, die Vertraulichkeit oder die Verfügbarkeit zu erhöhen, wenn der Entwickler davon ausgeht, dass dies erforderlich ist. Diese Annotation schränkt entsprechend die Dimensionen der Adaption ein, kann allerdings bei richtigem Einsatz die Ausführungswahrscheinlichkeit erhöhen.

- Die *timeoutFactor*-Annotation für Dienstaufrufe: Der Wert dieser Annotation gibt an, um welchen Faktor die durchschnittliche Ausführungszeit eines jeweiligen Dienstaufrufes verzögert werden kann, bevor dieser als gescheitert angesehen wird.

- Die *optional*-Annotation für Dienstaufrufe: Vom Entwickler als optional deklarierte Dienstaufrufe können dazu dienen, die Dienstqualität zu erhöhen, um hierdurch das entsprechende Adaptionsziel zu erreichen.
- Die *fidelity*-Annotation für Dienstaufrufe: Sie stellt einen numerischen Wert im Intervall zwischen null und eins dar, bei der ein höherer Wert eine höhere Qualität des Ergebnisses beschreibt – während der Laufzeit wird die Variante ausgewählt, die im Hinblick auf das Adaptionsziel die höhere Qualität bei gleichzeitig zu erwartender erfolgreicher Taskausführung besitzt.

  Bei der Nutzung ist ein Entwickler dazu aufgefordert, mehrere Varianten einer Komponente zu entwickeln und entsprechend zu annotieren.

6.4.2. Profiler

Wie in Abschnitt 5.3.3 beschrieben teilt sich die Erfassung der Profiling-Informationen anhand des jeweils verwendeten Profiler-Typs auf. So sollen rein anwendungs- und taskbezogene Informationen von einer entsprechenden Profiling-Komponente erfasst werden, wohingegen die hardware- und netzwerkbezogenen Profiling-Informationen zusammen mit weiteren Kontextattributen intervallbasiert vom Kontextmanager erfasst werden.

Im Zusammenhang mit der ebenfalls in Abschnitt 5.3.3 beschriebenen Besonderheit der Messung der Energie wird diese in Bezug auf die Taskausführung wie folgt ermittelt: Die Software PowerTutor [ZTD+10, Pow16] wurde erweitert [Bes16], um ein aktuelles mobiles Gerät zu unterstützen und auf diesem mobile Anwendungen bis zur Ebene einzelner Methodenausführungen im Hinblick auf ihren Energieverbrauch abschätzen zu können.

Entsprechend nutzt die Profiling-Komponente den im Zusammenhang mit dem Komponentenmodell beschriebenen Interceptor, um die Basiswerte für die in Tabelle 5.5 aufgeführten taskbasierten Informationen zu erfassen. Die Speicherung der erfassten Daten erfolgt hierbei in einer SQLite[4]-Datenbank auf dem mobilen Gerät. Die Messung der verfügbaren Bandbreite wurde passiv ausgelegt, um die Schnittstellen zur Mobilkommunikation und damit den Energievorrat des mobilen Gerätes nicht zusätzlich zu belasten.

6.4.3. Dienstsuche

Die Aufgabe der Dienstsuche ist es, dem Koordinator die für eine kontextadaptive Anwendung verfügbaren Dienstinstanzen zu liefern. Hierzu nutzt die Dienstsuche die von der Jadex-Middleware bereitgestellte Funktionalität. Die Suche verläuft aus Gründen der Effizienz nur bei Bedarf (on-demand), dies bedeutet, dass ein Dienst erst dann erfolgt, sobald der Koordinator diesen anfordert. Entsprechend wird auf eine regelmäßige Suche nach verfügbaren

[4] SQLite (www.sqlite.org) ist eine Programmbibliothek, die ein relationales Datenbanksystem enthält. Sie unterstützt einen Großteil der im SQL-92-Standard festgelegten SQL-Sprachbefehle und wird von der Android-Systemplattform bereitgestellt.

Diensten, ohne dass ein konkreter Adaptionsbedarf vorliegt, aufgrund des hierfür notwendigen Energiebedarfs zur Aktivierung der einzelnen Schnittstellen zur Mobilkommunikation verzichtet.

6.4.4. Partitionierung

Die Partitionierung soll initial erfolgen, sobald eine ausreichende Menge an Ausführungsstatistiken generiert wurde. Die Partitionierung erfolgt hierbei in Form einer graph-basierten Modellierung der einzelnen Komponenten und ihrer Interaktion untereinander im Hinblick auf die jeweils in Abschnitt 5.4.4.2 beschriebenen Kostenfunktionen und trennt dabei die lokal auszuführenden von den entfernt auszuführenden Komponenten einer mobilen Anwendung. Hierzu wird der MIN-CUT-Algorithmus verwendet, um die so entstehenden zwei Partitionen im Hinblick auf das jeweilige Ziel der Adaption zu optimieren.

Die Partitionierung kann ebenfalls als Mikroagent innerhalb der Jadex-Middleware realisiert werden; aufgrund der begrenzten Komplexität der im Rahmen der Evaluation untersuchten Anwendungsbeispiele wurde auf die Umsetzung der konzeptionell entwickelten Partitionierungskomponente verzichtet. Die generelle Umsetzbarkeit der entsprechenden Komponente wurde allerdings in unterschiedlichen Arbeiten gezeigt [OYL06, OYZ07, GNM+03, MGB+02, WKWS16], weswegen keine prinzipiellen Hindernisse bei der Integration der entsprechenden Funktionalität zu erwarten sind.

6.4.5. Koordinator

Die Koordinations-Komponente klinkt sich über die in der Jadex-Middleware vorhandene Interceptor-Funktionalität ein und steuert hierüber die Koordination der Tasks in Bezug auf die Auswahl der jeweils passenden Instanz für eine Auslagerung in die Infrastruktur. Der Koordinator führt hierzu eine der in Abschnitt 5.4.4.5 beschriebenen Adaptionsstrategien aus, die hierzu vom Entwickler für eine mobile Anwendung festgelegt wird. Entsprechend kann dies, abhängig von der jeweiligen Adaptionsformen auch zu einer Anpassung der Implementierung oder einer Anpassung der Ausführungszeit führen.

Hierbei werden ähnliche Tasks, die kurz aufeinanderfolgen und die eine kurze Ausführungszeit haben, mit derselben Adaptionsstrategie ausgeführt. Dabei sorgt der Koordinator dafür, dass jeweils nur ein Task zurzeit in die Infrastruktur übertragen oder zurückgeführt wird, um das im Konzept beschriebene FIFO-Prinzip umzusetzen.

6.4.6. Kontextmanager

Dem Kontextmanager wurden konzeptionell eine Reihe wichtiger Aufgaben zugeordnet, konkret sind dies: die Akquisition und Speicherung von Kontextdaten sowie die Bereitstellung dieser Daten über eine Schnittstelle an den

Koordinator, sowohl für das aktuelle Kontextintervall als auch für zukünftige Kontextintervalle im Rahmen einer Prognose.

Verarbeitung der Prognosemodelle Für die Erfassung und Speicherung der Sensordaten soll das AWARE-Framework [Cen16] verwendet werden. Dieses erfasst die in Tabelle 5.13 zusammengefassten Kontextattribute. Hinsichtlich der Messung der Bandbreite wurde hierzu eine Erweiterung vorgenommen, um die im Konzept beschriebene periodische Messung der Bandbreite zu realisieren. Die gesammelten Daten werden anschließend ebenfalls in einer SQLite-Datenbank gespeichert. Um die Ressourcen des mobilen Gerätes dabei nicht über Gebühr zu belasten, wird das Intervall, in dem die Kontextdaten gesammelt werden, auf fünf Minuten festgelegt.

In Bezug auf die Verarbeitung, Prognose und Bereitstellung der so ermittelten Kontextdaten erfolgte eine Implementierung auf Basis des in Abschnitt 5.5.4 beschriebenen Prozesses zur Generierung von Prognosemodellen. Die Aufteilung der hierzu erhobenen Daten in eine Trainings- (80 %) und eine Testmenge (20 %) erlaubt anschließend die Auswahl geeigneter Modelle. Das Training der Modelle erfolgt aus Gründen der Laufzeit auf einem mobilen Gerät nur periodisch und in einem wöchentlichen Rhythmus. Ebenso wurde die Merkmalsselektion zugunsten einer Hauptkomponententransformation (principal component analysis) aufgegeben, um die zu verarbeitende Datenmenge möglichst gering zu halten und anschließende Schritte in der Implementierung des Prozesses der generischen Kontextadaption möglichst performant auf dem mobilen Gerät ausführen zu können.

Ermittlung relevanter Prognosehorizonte Zur Ermittlung des Prognosehorizontes, den die trainierten Modelle abdecken sollen, wurde bei aufsteigender Sortierung das 90-Prozent-Quantil der beobachteten Zeitdauern der Taskausführung herangezogen. Dies dient dazu, den üblicherweise zu erwartenden Prognosehorizont angemessen abzudecken. Auf Basis dieses Maximalwertes wurde anschließend jeweils ein Modell pro enthaltenem Kontextintervall $t + n$ generiert, wie es in Abbildung 6.7 veranschaulicht wird. Darüber hinaus kann es sinnvoll sein weitere tageszeitabhängige Modelle auf Basis von Durchschnittswerten bereitzustellen, die über den zuvor ermittelten Zeitraum hinausgehende Prognosen ermöglichen.

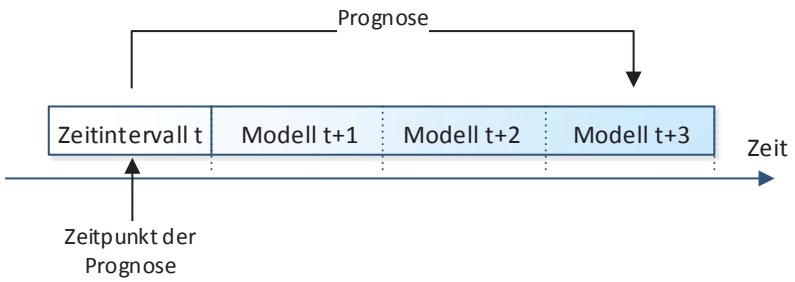

Abbildung 6.7.: Generierung der unterschiedlichen Prognosemodelle

Diese bedarfsorientierte Generierung der Prognosemodelle ist prinzipiell auf dem mobilen Gerät möglich, wurde allerdings parallel auf einem stationären Gerät entwickelt, um die für die Simulation benötigten Prognosen in entsprechend kurzer Zeit generieren zu können.

Umgang mit der Cold-Start-Problematik Im Hinblick auf die Cold-Start-Problematik (vergleich Abschnitt 5.5.5) wurde ein alternatives, rein auf Durchschnittswerten basierendes Prognoseverfahren für beliebige Kontextattribute ergänzt, welches nach dem Sammeln einer ausreichend großen Menge an Kontextdaten durch den eigentlichen generischen Prozess zur Kontextadaption ersetzt wird. Konkret wurde hierzu als Schwellwert die durchschnittliche Prognosequalität der generierten Modelle mit der einfachen, auf Durchschnittswerten basierenden Prognose verglichen. Dies war jedoch nur für Zielwerte der Prognose möglich, die im Anschluss auch tatsächlich beobachtet werden konnten. Waren die generierten Modelle innerhalb des Beobachtungszeitraumes eines Tages im Durchschnitt besser, erfolgte dann jeweils der Wechsel auf das entsprechende Prognosemodell. Die hierzu benötigte Historie von Kontextdaten wurde dabei einerseits durch passive Messungen (beispielsweise die Konnektivität) oder mithilfe einer Exporation (beispielsweise die Ausführungsstatistik) generiert.

6.5. Zusammenfassung

Im aktuellen Kapitel wurde zunächst sowohl die Auswahl eines geeigneten Software-Entwicklungsparadigmas als auch einer passenden Laufzeitumgebung für mobile und stationäre Geräte vorgestellt und deren Auswahl jeweils begründet. In diesem Zusammenhang wurde ebenso die Integration der ausgewählten Laufzeitumgebung in ein mobiles Betriebssystem beschrieben. Anschließend wurden in Form einer prototypischen Implementierung wesentliche Bestandteile des in Kapitel 5 entwickelten Konzepts umgesetzt. Hierdurch sollte gezeigt werden, wie es insgesamt möglich ist, das Konzept zu realisieren, obwohl zur Implementierung des Prozesses der generischen Kontextadaption gewisse Optimierungen im Hinblick auf die Ausführungszeit auf einem mobilen Gerät notwendig waren.

Im Folgenden sollen die in dieser Arbeit vorgestellten Ideen einer kontextadaptiven Anwendungsarchitektur für mobiles Cloud Computing evaluiert werden. Hierzu werden sowohl das Konzept selbst als auch die entsprechende Implementierung anhand einer Beispielanwendung analysiert und bewertet.

7. Evaluation und Bewertung des eigenen Ansatzes

In diesem Kapitel wird eine Evaluation des vorgeschlagenen Konzepts einer kontextadaptiven Anwendungsarchitektur vorgestellt. Ziel dieser Evaluation ist es, aufzuzeigen, ob das vorgestellte Konzept und dessen prototypische Implementierung geeignet sind, die im Rahmen dieser Arbeit als relevant identifizierten Anwendungsbereiche angemessen zu unterstützen. Die Evaluation soll dabei sowohl qualitativ als auch quantitativ erfolgen.

Im Rahmen der qualitativen Evaluation wird in Abschnitt 7.1 deskriptiv bewertet, ob die in der Anforderungsanalyse in Abschnitt 4.2 erhobenen nichtfunktionalen Anforderungen an das Konzept erfüllt wurden und ob sich diese im Rahmen der prototypischen Implementierung umsetzen ließen. Anschließend wird im Rahmen der quantitativen Evaluation bewertet, inwieweit der prototypische Einsatz der entwickelten Lösung möglich ist. Hierbei sollen insbesondere die nichtfunktionalen Anforderungen bewertet werden, die sich mithilfe der Erhebung bestimmter Kennzahlen im Einsatz der Lösung evaluieren lassen. Diese Trennung erfolgt, um nichtfunktionale Kriterien wie die Performance, die sich vergleichsweise einfach quantitativ ermitteln lassen, sowie auch Kriterien wie die Wartbarkeit der Lösung, die sich einfacher qualitativ bewerten lassen, insgesamt angemessen bewerten zu können.

Hierzu erfolgt anhand zweier repräsentativer Anwendungsbeispiele sowohl eine Simulation als auch eine Evaluation des vorgestellten Ansatzes. Der jeweilige Versuchsaufbau, die durchgeführten Versuchsreihen und die ermittelten Ergebnisse werden entsprechend in Abschnitt 7.2 bewertet, bevor in Abschnitt 7.3 die Ergebnisse der Evaluation zusammengefasst werden.

7.1. Qualitative Evaluation

Zur Evaluation des eigenen Ansatzes werden die im Rahmen der Anforderungsanalyse erhobenen Kriterien herangezogen, die zunächst deskriptiv bewertet werden, bevor die Ergebnisse anschließend in Tabelle 7.1 zusammengefasst werden. Eine kommentierte Version der Bewertung findet sich in Anhang A.

Verfügbarkeit Die kontextadaptive Anwendungsarchitektur wurde in Bezug auf die Fehlerbehandlung so entwickelt, dass sie fehlgeschlagene Ausführungen einzelner Tasks versucht, durch eine erneute Ausführung im Rahmen einer Anpassung der Adaptionsstrategie zu kompensieren, wenn diese als erfolgversprechend bewertet wird und wenn die in diesem Zusammenhang zu erwartende Ausführungszeit die insgesamt definierte zulässige Verzögerung nicht überschreitet. Hierdurch werden Ereignisse wie die wechselnde Konnek-

tivität mobiler Geräte oder ein zu geringer verbleibender Energievorrat als zu erwartende Ereignisse im Rahmen eines schnell wechselnden Kontextes mobiler Geräte berücksichtigt (Anforderung A1 und A2).

Zur Beurteilung der Effizienz einer bestimmten Adaptionsstrategie im Hinblick auf das jeweilige Ziel der Adaption und den Erfolg der Ausführung eines Tasks nutzt der Koordinator eine Reihe von Prognosen des Kontextdienstes, beispielsweise die zukünftige Konnektivität eines Surrogates (Anforderung A3). Aus Gründen der Effizienz und um weder das mobile Gerät noch die Infrastruktur über Gebühr zu belasten, werden Tasks erst dann erneut initiiert, wenn ihre aktuelle Ausführung als nicht erfolgversprechend bewertet wird (Anforderung A4). Bereits in der Infrastruktur laufende Tasks können allerdings nur dann abgebrochen werden, wenn zu dem sprechenden Surrogate noch eine Verbindung besteht. Die Erfüllung der Anforderungen im Hinblick auf die Verfügbarkeit des entwickelten Konzepts ist damit insgesamt als gut zu bewerten.

Portierbarkeit Die generelle Fähigkeit zur Integration der Infrastruktur in die kontextadaptive Ausführung wird durch Möglichkeit zur Auslagerung bestimmter Tasks realisiert (Anforderung P1), wozu es allerdings erforderlich ist, dass diese Dienste bereits in der Infrastruktur vorhanden sind. Der Koordinator stellt dabei zusammen mit dem Kontextdienst sicher, dass die unterschiedlichen Tasks sinnvoll im Hinblick auf das jeweilige Adaptionsziel in Bezug auf Ausführungszeit, Ausführungsort und Implementierung angepasst werden (Anforderung P2 und P3). Die kontextadaptive Architektur und ihre Systemunterstützung machen dabei keine exklusive Nutzung von den Ressourcen des mobilen Gerätes und integrieren sich als gleichberechtigter Betriebssystemprozess und nutzen existierende Mechanismen zur Betriebsmittelverwaltung wie den Prozess-Scheduler der Android-Systemplattform. Da die Systemunterstützung jedoch nicht in das Betriebssystem des mobilen Gerätes integriert ist, sondern als reguläre Anwendung in diesem läuft, sind insbesondere Aufgaben wie die Suche nach verfügbaren Surrogates nur bedingt mit den Energiesparmechanismen des mobilen Gerätes vereinbar (Anforderung P4). Ebenso ist eine anwendungsübergreifende Koordination unterschiedlicher kontextadaptiver Anwendungen aufgrund der in der Android-Systemplattform umgesetzten Isolation ausgeführter Anwendungen zueinander nur bedingt möglich. Diese Art des Deployments entspricht allerdings der üblichen Installation mobiler Anwendungen, unterstützt einen einfachen Deployment-Prozess und setzt lediglich das Vorhandensein einer Java Virtual Machine voraus (Anforderung P5 und P6). Insgesamt kann damit die Erfüllung der Anforderungen im Hinblick auf die Portierbarkeit der Lösung ebenfalls als gut bewertet werden.

Skalierbarkeit Im Rahmen der Konzeption und prototypischen Implementierung wurde versucht, den Overhead zu minimieren, der im Wesentlichen durch die Akquisition und Speicherung der unterschiedlichen Kontextattribute ent-

steht. Die entsprechenden Auswirkungen auf den Energie- und Bandbreitenbedarf wurden dadurch auf ein erforderliches Minimum reduziert, das die Erfassung der Kontextdaten auf ein entsprechendes Intervall und die Ermittlung der verfügbaren Bandbreite auf Zeiträume reduziert, in denen eine kontextadaptive Anwendung auch tatsächlich genutzt wird. Qualitativ betrachtet kann der Overhead damit als vertretbar angesehen werden (Anforderung S1), ein Aspekt der im Rahmen der quantitativen Evaluation allerdings noch erneut aufgegriffen wird (vergleiche Abschnitt X). Aus konzeptioneller Sicht wurde sowohl eine proaktive als auch eine rein reaktive Möglichkeit zur Dienstsuche entwickelt, die allerdings aus den zuvor genannten Gründen des Overheads zunächst rein reaktiv implementiert wurde und zusätzlich nur aktiv ist, wenn kontextadaptive Anwendungen ausgeführt werden. Sobald das mobile Gerät mit einer externen Energiequelle verbunden ist, könnte diese zulasten eines höheren Overheads proaktiv die Dienste laufender Anwendungen suchen, um auch spontane Interaktionsszenarien von Tasks mit kurzer Laufzeit sinnvoll im Rahmen einer Auslagerung in die Infrastruktur unterstützen zu können (Anforderung S2).

Das Konzept der taskbasierten Adaption erlaubt es darüber hinaus, die Ausführung von nebenläufigen Tasks zu gestalten, wobei die Korrektheit der nebenläufigen Ausführung durch die Verwendung des Aktormodells von der Jadex-Middleware sichergestellt wird. Entsprechend ist es ohne eine weitere Anpassung der Geschäftslogik durch den Entwickler möglich, eine Parallelverarbeitung von Tasks durchzuführen (Anforderung S3). Die Möglichkeit zum Pausieren ausgelagerter Tasks auf Surrogates, um diese zu entlasten, wurde allerdings im Hinblick auf die geforderte Effizienz und die hierdurch zu erwartenden, geringeren Erfolg der Ausführung eines Tasks nicht umgesetzt (Anforderung S4). Ebenso wurde aufgrund der Einfachheit der Beispielanwendungen auf die Umsetzung einer Möglichkeit zur automatisierten Partitionierung mobiler Anwendungen verzichtet. Die Erfüllung der Kriterien in Bezug auf die Skalierbarkeit können damit ingesamt trotzdem als gut erfüllt angesehen werden.

Nutzbarkeit Wie durch die Umsetzung der Beispielanwendung im Rahmen der quantitativen Evaluation gezeigt werden wird, erfordert die Nutzung des in der prototypischen Realisierung entwickelten CloudAware-Frameworks lediglich grundlegende Kenntnisse im Bereich der komponenten-orientierten Softwareentwicklung (Anforderung U1). Die spezifischen Problemstellungen im Hinblick auf das mobile Cloud Computing konnten dabei weitgehend vor dem Entwickler verborgen werden (Anforderung U2). Das CloudAware-Framework integriert sich dabei weitestgehend in den de-facto standardisierten Entwicklungsprozess mobiler Anwendungen und die dort verwendeten Werkzeuge (Anforderung U3). Ebenso erfolgt die Konfiguration durch für den Endanwender oder den Entwickler verständlich gestaltete Optimierungsziele, wie beispielsweise der Einsparung von Energie, Bandbreite oder Ausführungszeit. Die Nutzbarkeit der Lösung wird damit insgesamt als gut bewertet (Anforderung U4).

Anforderung		Konzeptionell gelöst	Prototypisch realisiert
Verfügbarkeit			
A1	Fehlerbehandlung	++	++
A2	Erweitertes Fehlermanagement	++	++
A3	Prognose der Konnektivität	++	++
A4	Effizienz	+	+
Portierbarkeit			
P1	Generelle Auslagerungsfähigkeit	+	+
P2	Grundlegende Adaptionsfähigkeit	++	++
P3	Erweiterte Adaptionsfähigkeit	++	++
P4	Koexistenz	+	-
P5	Deployment	+	+
P6	Offenheit	++	++
Skalierbarkeit			
S1	Overhead	+	+
S2	Dienstsuche und Integration	+	+
S3	Parallelisierung	+	+
S4	Effizienz	+	-
Nutzbarkeit			
U1	Einfache Nutzbarkeit	++	++
U2	Angemessene Abstraktion	++	++
U3	Unterstützung von Standards	++	++
U4	Automatische Konfiguration	+	+
Wartbarkeit			
M1	Determiniertheit	+	+
M2	Debugging	++	++
M3	Offene Standards	+	+
Sicherheit			
SE1	Schutz der Vertraulichkeit	+	+
SE2	Isolation	-	-

++ vollständig erfüllt + teilweise erfüllt - nicht erfüllt 0 nicht adressiert

Tabelle 7.1.: Evaluation des entwickelten Lösungsansatzes

Wartbarkeit In Bezug auf die Wartbarkeit des entwickelten Ansatzes kann festgehalten werden, dass der fachliche Kontrollfluss der Anwendung unabhängig von den verwendeten Surrogates, des Ausführungskontextes oder der Adaptionsstrategie ist und lediglich im Rahmen der Adaptionsziele verzögert wird (M1). Zusätzlich besteht die Möglichkeit für den Entwickler einer mobilen Anwendung, mehrere Implementierungen im Rahmen einer Fidelity-Adaptation bereitzustellen. Bezüglich des Debuggings und der Wartung der Anwendung integriert sich die vorgestellte Lösung ebenfalls in den Entwicklungsprozess und die (Debugging-)Werkzeuge für mobile Anwendungen. In der prototypischen Implementierung werden entsprechend nur freiverfügbare Basistechnologien eingesetzt, um keine Abhängigkeit zu einem bestimmten Dienstanbieter zu erzeugen (Anforderung M3). Die Wartbarkeit der Lösung kann damit insgesamt als gut betrachtet werden.

Sicherheit Der Schutz der Vertraulichkeit von Informationen, die von einer kontextadaptiv ausgeführten Anwendung verarbeitet werden, kann dadurch hergestellt werden, dass entsprechende Komponenten mit einer *local*-Annotationen versehen werden (Anforderung SE1). Darüber hinaus ist durch die Nutzung der Jadex-Middleware eine sichere Kommunikation innerhalb einer geteilten Infrastruktur und innerhalb öffentlicher Netzwerke wie WLAN-Zugangspunkten sichergestellt. Schutzmechanismen, welche die Seiteneffekte auf Surrogates durch nicht vertrauenswürdigen Codes von kontextadaptiven Anwendungen vermeiden sollen, wurden in der aktuellen Konzeption nicht adressiert (Anforderung SE2). Die mit der Sicherheit verbundenen Kriterien können damit ingesamt als berücksichtigt angesehen werden.

7.2. Quantitative Evaluation

Nachdem der gewählte Ansatz für eine kontextadaptive Anwendungsarchitektur zunächst qualitativ untersucht wurde, soll im Folgenden anhand von zwei Evaluationsprojekten, die stellvertretend für die in Abschnitt 4.1 untersuchten Anwendungsfälle ausgewählt wurden, die praktische Nutzbarkeit des gewählten Ansatzes zunächst geprüft und im Anschluss quantitativ bewertet werden. In diesem Zusammenhang sollen ausgewählte Laufzeitdaten des entwickelten Ansatzes anhand dieser beiden Evaluationsprojekte untersucht werden.

Im ersten Evaluationsprojekt erfolgt hierzu die Entwicklung einer Beispielanwendung auf Basis des in der prototypischen Realisierung umgesetzten komponentenbasierten Entwurfes. Repräsentativ für die in Abschnitt 4.1 vorgestellte Kategorie von Anwendungsfällen im Bereich der Augmented Reality wird hierzu die Auslagerung von Geschäftslogik im Hinblick auf die zeitkritische Verarbeitung von Daten im Rahmen der Objekt- und Gesichtserkennung untersucht. Dieses Evaluationsprojekt zielt entsprechend auf synthetische Benchmarks ab, die für einen konkreten Anwendungsfall die Grenzbereiche einer Kooperation mit der Infrastruktur aufzeigen.

Im zweiten Evaluationsprojekt wird anschließend die in Abschnitt 4.1 vorgestellte Kategorie der Video- und Bildverarbeitung als Grundlage für die Entwicklung einer weiteren Beispielanwendung genutzt. Entgegen dem ersten Evaluationsprojekt wird auf dieser Basis eine Simulation durchgeführt, die darauf abzielt, die breite Anwendbarkeit der vorgeschlagenen kontextadaptiven Anwendungsarchitektur sowohl über längere Nutzungszeiträume als auch im Hinblick auf die individuellen Nutzungsmuster unterschiedlicher Anwender zu untersuchen.

Als Evaluationskriterien sollen die in Kapitel 5.2.3 erarbeiteten Adaptionsziele herangezogen werden. Zu dem jeweiligen Evaluationsprojekt passend werden dabei jeweils unterschiedliche Adaptionsziele ausgewählt und ihre Erreichbarkeit quantitativ bewertet. Die Auswahl der zu messenden Kennzahlen stützt sich entsprechend auf die in Kapitel 5.2.3 ermittelten Ziele der Adaption. Das vorgelagerte Ziel ist dabei, die Anzahl der erfolgreich adaptiv und

kooperativ ausgeführten Tasks zu maximieren. Nachgelagert werden eine Verbesserung der Performance sowie eine Einsparung von Energie untersucht.

7.2.1. Evaluationsprojekt FaceMatch

Wie in Kapitel 4 im Rahmen der Anwendungsbeispiele hervorgehoben wurde, stellt ein wesentlicher Anwendungsbereich mobiler Geräte und ihrer Anwendungen die Bereitstellung einer erweiterten Realität (Augmented Reality) dar. Eine Vielzahl von mobilen Geräten ist hierzu bereits mit einer leistungsfähigen Kamera ausgestattet, wodurch es möglich ist, Bildmaterial aus der direkten Umgebung um zusätzliche Inhalte zu erweitern.

7.2.1.1. Implementierung der Beispielanwendung

Im Evaluationsprojekt FaceMatch wird eine mobile Anwendung entwickelt, die mithilfe der *OpenCV*-Bibliothek [Ope16] unterschiedliche Algorithmen zur Bildverarbeitung sowie Objekt- und Gesichtserkennung[1] nutzt.

Die entwickelte mobile Anwendung wertet hierzu in Echtzeit die aufgenommenen Bilder des mobilen Gerätes aus, um in einem ersten Schritt eine Objekterkennung durchzuführen. Ziel dieser Objekterkennung ist es, Gesichter von Personen in der Szenerie eines Bildes zu erkennen. In einem zweiten Schritt werden als Gesichter erkannten Bildausschnitte im Hinblick auf ihre Ähnlichkeit zu bereits in einer Datenbank vorhandenen Bildern untersucht. Die dabei zu einem Kamerabild gefundenen, bereits bekannten Bilder werden in einem dritten Schritt im Display des mobilen Geräts angezeigt und optional um Meta-Informationen wie den Namen der erkannten Person ergänzt. Abbildung 7.1 zeigt exemplarisch das Ergebnis eines solchen Erkennungsvorgangs.

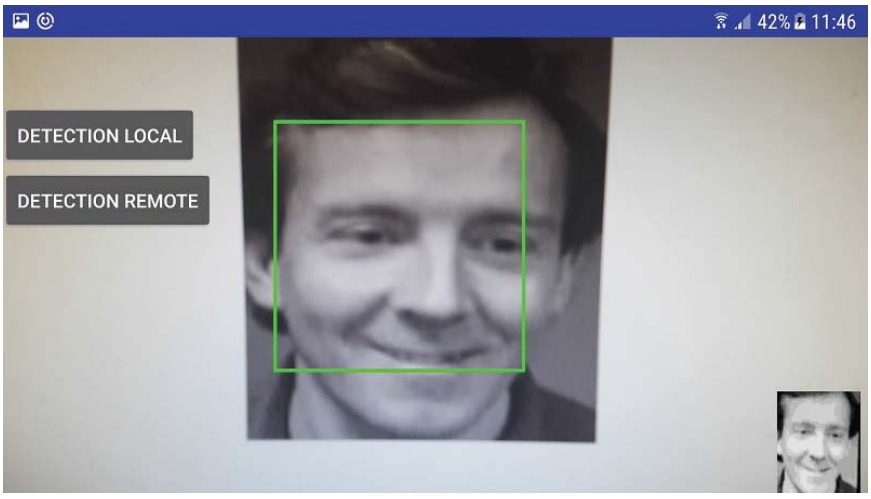

Abbildung 7.1.: Mobile Anwendung FaceMatch: Objekt- und Gesichtserkennung

[1] Für die *face detection* und *face recognition* ist im deutschen Sprachgebrauch der Begriff Gesichtserkennung üblich. Ersteres wird im Folgenden mit Objekterkennung (Erkennen von Mustern mit der Form eines Gesichts) übersetzt und letzteres mit Gesichtserkennung.

Die lokale Ausführung einer Objekt- und Gesichtserkennung ist auf aktuellen mobilen Geräten zwar ohne Unterstützung der Infrastruktur möglich, jedoch ist der Ressourcenbedarf hierbei vergleichsweise hoch und die Auswahl der hierfür nutzbaren Algorithmen auf solche beschränkt, die qualitative Einschränkungen im Vergleich zu denen besitzen, die auf stationären Geräten zum Einsatz kommen. Entsprechend nutzt diese Anwendung die Adaptionsform der Verschiebung des Ausführungsortes im Zusammenhang mit einer Fidelity-Adaptation, um eine Kooperation mit der Infrastruktur zu realisieren.

Hierzu wurde ein komponentenbasierter Entwurf genutzt, bei dem die Funktionalität der Objekterkennung von der Funktionalität der Gesichtserkennung in Form zweier unterschiedlicher Komponenten getrennt wurde. Um nur als Gesicht erkannte Objekte zu übertragen und damit den zu übertragenden Zustand möglichst gering zu halten, verbleibt dabei die Komponente für die Objekterkennung (Erkennung der Gesichter) auf dem mobilen Gerät und es wird lediglich die Gesichtserkennung (Identifikation der Person) in die Infrastruktur verlagert. Aufgrund der begrenzten Komplexität der Beispielanwendung konnte auf eine weitergehende Partitionierung, die sich aus konzeptioneller Sicht an die Entwicklung der einzelnen Komponenten anschließt, verzichtet werden. Weitere Details zur Implementierung finden sich in [Kit17]. Die Anforderungen dieser Anwendung an die Latenz

7.2.1.2. Ergebnisse

In Bezug auf die Evaluationskriterien sollen die Erreichbarkeit der Adaptionsziele der Zeiteinsparung und der Erhöhung der Dienstqualität durch eine Verlagerung in die Infrastruktur untersucht werden. Insbesondere soll dabei untersucht werden, inwieweit durch die Auslagerung in die Infrastruktur eine Zeitverzögerung entsteht, die zu einer Einschränkung der Dienstqualität aus Nutzerwahrnehmung führen kann. Mit Bezug auf diese Evaluationskriterien sollen im Rahmen der Evaluation die folgenden zwei Szenarien untersucht werden:

- Zeiteinsparung und Verzögerung: Die Zeiteinsparung wird durch eine Übertragung der Gesichtserkennung in die Infrastruktur durchgeführt. Eine eventuelle Verzögerung würde durch eine Überschreitung der lokalen Ausführungszeit sichtbar.

- Erhöhung der Dienstqualität: Die Erhöhung der Dienstqualität wird in Form einer Fidelity-Adaptation durchgeführt. Hierzu wird für die Ausführung innerhalb der Infrastruktur eine von der lokalen Implementierung abweichende Implementierung der Gesichtserkennung verwendet. Dieses Kriterium wird ersatzweise statt der Anzahl erfolgreich ausgeführter Tasks herangezogen, die sich ebenso wie der tatsächliche Energieverbrauch nur sinnvoll mit einer über einen längeren Zeitraum durchgeführten Evaluation hätten ermitteln lassen.

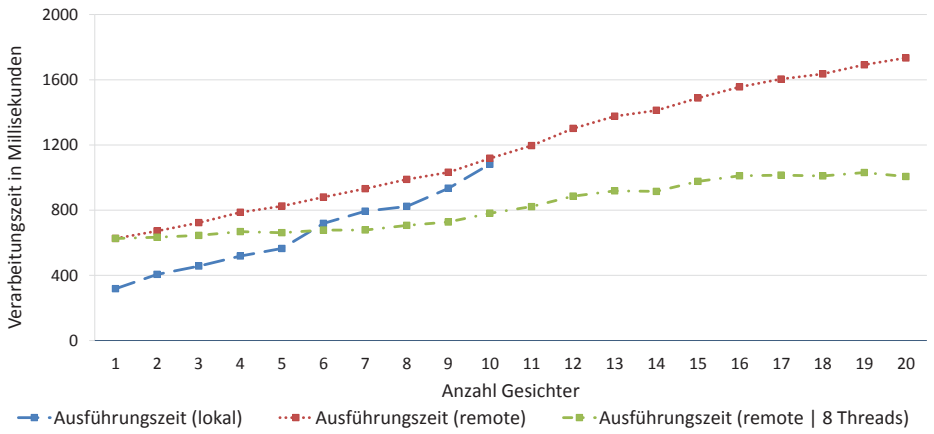

Abbildung 7.2.: Ausführungszeit der Objekt- und Gesichtserkennung

Ergebnisse der Evaluation Abbildung 7.2 fasst die Erkennungsleistungen für einen Datensatz mit 30 unterschiedlichen Gesichtern zusammen, für die jeweils zwei zu erkennende Originale in der Datenbasis der Anwendung hinterlegt waren.

In Bezug auf das Adaptionsziel der Zeiteinsparung zeigt sich, dass abhängig von der Anzahl erkannter Gesichter und dem damit zu übertragenden Zustand eine Kooperation mit der Infrastruktur sinnvoll realisiert werden kann. Sobald fünf oder mehr Gesichter zu identifizieren sind, kann dies mithilfe einer Parallelverarbeitung in der Infrastruktur durchgeführt werden und insgesamt zu einer Zeiteinsparung führen. Gleichzeitig ist zudem die Erkennungsleistung deutlich besser: Im Zusammenhang mit dem zweiten Adaptionsziel, der Erhöhung der Dienstqualität, konnte ebenso gezeigt werden, dass ein Austausch der Implementierung die Erkennungsleistung erhöht, ohne dass sich dies maßgeblich auf die benötigte Rechenzeit auswirkt. So wurden mithilfe der lokalen Implementierung 15 Gesichter korrekt identifiziert, mit der Implementierung in der Infrastruktur wurden 18 Gesichter korrekt identifiziert. Zusätzlich zeigt sich, dass ein aktuelles mobiles Gerät bei zunehmender Komplexität der zu analysierenden Szenen nicht alle vorhandenen Gesichter erfolgreich erkennen und identifizieren kann. Durch eine Kooperation mit der Infrastruktur kann dieser Restriktion ebenfalls begegnet und somit die Dienstqualität weiter erhöht werden.

7.2.2. Evaluationsprojekt ImageEffect

Im zweiten Evaluationsprojekt liegt der Fokus auf der Evaluation der Fähigkeit zur Kontextadaption in Bezug auf die einzelnen Ziele der Adaption. Um die Generalisierungsfähigkeit des entwickelten Ansatzes zu ermitteln, soll dieser Teil der Evaluation sowohl auf entsprechend aussagefähigen Nutzungszeiträumen als auch auf unterschiedlichen, repräsentativen Nutzungsprofilen durchgeführt werden. Aus diesem Grund wird für diesen Teil der Evaluation ein Simulationsmodell einer Infrastruktur entwickelt, welches insbesondere

die wechselnde Konnektivität eines mobilen Gerätes im Zusammenhang mit dessen wechselndem Kontext berücksichtigt.

Entsprechend wird im Folgenden zunächst die Implementierung einer Beispielanwendung beschrieben, bevor im Anschluss mit deren Hilfe ein entsprechender Datensatz mit Kontextdaten ausgewählt und ergänzt wird, um im Anschluss eine Anwendung zur Bildverarbeitung in diese Simulation einzubetten und hierdurch die Adaptionsziele der Zeit- und Energieeinsparung auf repräsentativen Nutzungsmustern unterschiedlicher Anwender zu untersuchen.

In Bezug auf die Evaluationskriterien sollen abweichend zum vorherigen Beispiel die Adaptionsformen der Verschiebung des Ausführungsortes und der Ausführungszeit genutzt werden. Zusätzlich wird die Adaptionsform der Fidelity-Adaptation im Zusammenhang mit der kontextadaptiven Ausführung betrachtet.

Implementierung der Beispielanwendung Neben dem Anwendungsgebiet der Augmented Reality, welches im vorhergehenden Anwendungsbeispiel gezeigt wurde, steht ebenso der Anwendungsbereich der Verarbeitung von Video- und Bilddaten (vergleiche Abschnitt 4.1) stellvertretend für ein wachsendes Segment mobiler Anwendungen, die ebenso von der Leistungssteigerung der Kameras mobiler Geräte profitieren. An die Aufnahme eines solchen Bildes schließen sich in der Regel eine Reihe von Verarbeitungsschritten an, ein prominentes Beispiel hierfür ist der Anbieter Instagram[2], der sich mit seinen mobilen Anwendungen in Kombination mit der zugehörigen Plattform darauf spezialisiert hat, die Bilder der Nutzer nach deren Wünschen zu bearbeiten und sie im Anschluss zu veröffentlichen.

Entsprechend wurde als Teil des Evaluationsprojekts zur Bildverarbeitung einem Studierenden als Aufgabe vorgegeben, eine Anwendung zur Bildverarbeitung zu entwickeln, die es erlaubt, verschiedene Kombinationen von Bildfiltern auf ein vorgegebenes Bild anzuwenden. Einzige Vorgabe war es hierbei, eine komponentenbasierte Realisierung in Form von Jadex-Mikroagenten vorzunehmen und, entsprechend dem in Kapitel 5 vorgestellten Komponentenmodell, diese gegebenenfalls mit den in Abschnitt 6.4 entwickelten Annotationen (wie beispielsweise der *local*-Annotation für Komponenten, die auf dem mobilen Gerät verbleiben sollen) zu ergänzen.

Das Ergebnis dieser Implementierung weist insgesamt fünf Komponenten auf und soll im Rahmen der Evaluation verschiedener Verarbeitungspfade und Varianten als hinreichend komplex hierfür betrachtet werden. Als relevant für die Verteilung soll an dieser Stelle hervorgehoben werden, dass für diese Form der Verarbeitung keine zustandsbehafteten Komponenten benötigt wurden. Lediglich die Komponente, die mit dem Android-Service interagiert, wurde wegen der Zugriffe auf die Ressourcen (wie das Dateisystem) des mobilen Geräts als „lokal“ und damit auf dem mobilen Gerät verbleibend annotiert.

[2] http://www.instagram.com

Die Interaktion mit der grafischen Benutzeroberfläche, die als Android Activity realisiert ist, läuft über die in Abschnitt 6.4 beschriebene Kopplung zwischen Activity und Service, wobei letzterer wiederum direkt mit den einzelnen Komponenten in der Jadex-Middleware interagiert. Die anschließende Anbindung der Anwendung an die prototypische Realisierung einer Systemunterstützung erfolgte durch das Einbinden des hierfür vorgesehenen Interceptors. Das Ergebnis stellt die in Abbildung 7.3 gezeigte Realisierung der Anwendung auf einem mobilen Gerät dar.

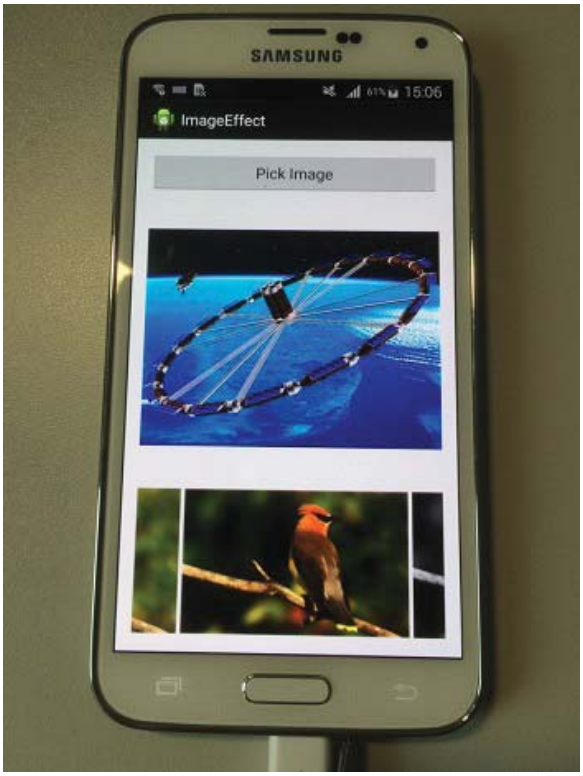

Abbildung 7.3.: Beispielanwendung: ImageEffect ausgeführt auf einem mobilen Gerät

7.2.2.1. Entwicklung des Simulationsmodells

Nachdem so weit die Entwicklung der Beispielanwendung erläutert wurde, soll im Folgenden das zugehörige Simulationsmodell beschrieben werden.

Profiling des mobilen Gerätes Das für die Simulation genutzte mobile Gerät ist ein Samsung Galaxy S5, welches Mitte 2014 als Smartphone im oberen Preissegment vorgestellt wurde und im Jahr 2016 noch als repräsentatives und gebräuchliches Smartphone angesehen werden kann. Hieraus leitet sich entsprechend die Eignung für die folgenden Messungen ab. Als Betriebssystem kommt das von Samsung ausgelieferte Android-Betriebssystem in der Version 5.0 zum Einsatz. Die relevanten Merkmale dieses Geräts, von dem mehrere Varianten existieren, können der Tabelle 7.2 entnommen werden.

Model	SM-G901F
Operating system	Android 5.0, Lollipop
CPU	2.5 GHz quad-core Krait 400
RAM	2 GB
Batterie	2800 mAh Li-ion
Funkstandard	LTE
Display	Full HD Super AMOLED

Tabelle 7.2.: Spezifikation des mobilen Gerätes

Die zugehörige Messung der Energie, die für die Ausführung benötigt wird, wurde im Rahmen einer Abschlussarbeit [Bes16] umgesetzt. Im Laufe dieser Arbeit wurden zunächst hardwarebasierte Messungen durchgeführt, um den tatsächlichen Energiebedarf der Methodenausführung zu ermitteln. Auf dieser Basis wurde im Anschluss eine Erweiterung des für die softwarebasierte Messung des Energiebedarfs existierenden Software „Power Tutor“ entwickelt. Dieses sogenannte *Power Model*, welches spezifisch für das verwendete mobile Gerät erstellt wurde, erlaubt es den tatsächlichen zusätzlichen Energieverbrauch, der einer Methodenausführung zuzuordnen ist, zu ermitteln, wie in Abbildung 7.4 veranschaulicht wird.

Abbildung 7.4.: Versuchsaufbau zur Energiemessung und der zugehörigen Messung, entnommen aus (Bes16)

In diesem Zusammenhang wurde ebenfalls die Messung der Ausführungszeit umgesetzt, ein exemplarischer Ausschnitt der ermittelten Werte für einzelne Methodenaufrufen findet sich in Tabelle 7.3.

Für die mit dem gezeigten Verfahren durchgeführte Messung der Energie der Datenübertragung mittels WLAN oder WWAN konnten hingegen keine zuverlässigen Messwerte erzeugt werden, wie in [Bes16] gezeigt wurde. Für die Energie, die für die Nutzung der Netzwerkschnittstellen WLAN und WWAN genutzt wird, wird daher auf die in [Spi15] ermittelten Werte zurückgegriffen.

Bildfilter	sheding_green	brightness	gamma	mean_remove
Laufzeit in s	0,52	2,11	2,33	7,223
CPU-Verbrauch in mJ	483	1710	2367	9849
CPU-Verbrauch in J für 1 s	0,929	0,810	1,016	1364
Gesamtverbrauch mJ	769	2735	3455	13042

Tabelle 7.3.: Messung des Energieverbrauches verschiedener Bildfilter

Profiling der Infrastruktur Für die Simulation eines Cloudlets wurde ein zeitgemäßer Arbeitsplatzrechner (i5-3570, 3,4 Ghz Quad-Core, 8 GB Arbeitsspeicher) herangezogen. Bei der Simulation des Cloud-Servers wurde eine vServer-Instanz eines Cloud-Service-Providers verwendet. In beiden Fällen wurde die entwickelte Beispielanwendung so modifiziert, dass sie eine Exploration durchführt und entsprechend unabhängig vom jeweiligen Ziel der Adaption sämtliche für das Profiling gestarteten Tasks jeweils auf allen verfügbaren Surrogates ausführt.

7.2.2.2. Auswahl eines Beispieldatensatzes

Dem gewählten explorativen Vorgehensmodell soll folgend ein Beispieldatensatz ausgewählt werden, welcher eine möglichst hohe Abdeckung der im vorhergehenden Abschnitt entwickelten Menge von Kontextattributen aufweist.

Im Hinblick auf die Auswahl eines Datensatzes, der einerseits der Untersuchung relevanter Kontextattribute dienen soll und auf dem idealerweise auch die Evaluation der kontextadaptiven Anwendungsarchitektur in Form einer Simulation erfolgen kann, ergeben sich eine Reihe von Anforderungen:

Zunächst sollte dieser Datensatz alle oder einen Großteil der Kontextattribute, die aktuelle mobile Geräte über ihre physischen und logischen Sensoren wahrnehmen, enthalten, die mit Blick auf die verwandten Arbeiten als relevant ermittelt wurden.

Mit Blick auf die Problemstellung des mobilen Cloud Computings und der Konnektivität soll dieser Datensatz dabei insbesondere Kontextdaten in Bezug auf die Nutzung der Schnittstellen zur Mobilkommunikation enthalten, welche sowohl die Konnektivität zu einzelnen Zugangspunkten als auch die jeweils erzielte oder erzielbare Bandbreite umfassen. Gleichzeitig ist es erforderlich, dass der Datensatz eine repräsentative Menge von Nutzungsmustern beinhaltet, die die typische Nutzung mobiler Geräte widerspiegelt. Ebenso ist es erforderlich, dass der Erhebungszeitraum ausreichend groß ist, um auch spezielle Ereignisse wie Urlaube oder den Wechsel von Verhaltensmustern mobiler Nutzer zu berücksichtigen.

Mit Blick auf die Untersuchung von Lane et al. in [LML+10], die sich auf Datensätze beziehen, in denen Sensordaten mobiler Geräte erfasst wurden, und die Ergebnisse des Projekts *Crawdad* [Cra16], welches sich zum Ziel gesetzt hat, Datensätze im Hinblick auf die Nutzung von drahtlosen Netzwerken

zu archivieren, zeigt sich, dass nur wenige Datensätze dieser Anforderung gerecht werden. Im Hinblick auf die Nutzungsdauer bieten sich lediglich vier Datensätze für die nähere Untersuchung an.

CenceMe So stellt der *CenceMe*-Datensatz [MPF+10, MLF+08] zwar eine ausreichend große Datenbasis mit den hochaufgelösten, via GPS ermittelten, Positionsdaten mobiler Nutzer dar, erfasst aber keine Informationen zur Konnektivität oder zu weiteren Kontextattributen.

MIT-Reality Im *MIT-Reality*-Datensatz [EP06] existieren mit einem Jahr Erfassungszeit und hundert Nutzern eine repräsentative Menge an Nutzungsmustern. Diese enthalten jedoch lediglich die über Bluetooth wahrgenommenen Kontakte mit anderen mobilen Geräten und ihren Nutzern. Da mit Blick auf den zweiten Teil der Anforderung, dass der Datensatz ebenfalls zur Evaluation genutzt werden soll, sich die Erfordernis ergibt, dass ebenso Nutzungsmuster der verwendeten mobilen Anwendungen vorhanden sind, bietet sich auch dieser Datensatz nicht für die Nutzung im Rahmen der Simulation an.

LiveLab Der *LiveLab*-Datensatz [SRT+11] beinhaltet die erforderlichen Nutzungsmuster und enthält für 25 Nutzer und jeweils ein Jahr eine Vielzahl von Kontextattributen in Verbindung mit der Konnektivität eines mobilen Geräts zu seiner Infrastruktur. Allerdings ist auch dieser Datensatz beschränkt im Hinblick auf das Messintervall und somit die Auflösung, beispielsweise wird der Beschleunigungssensor lediglich alle 15 Minuten gemessen.

LDDC Eine weitaus größere Datenbasis wurde im Rahmen des zweiten Datensatzes, der *Lausanne Data Collection Campaign (LDDC)* gesammelt. Die LDDC wurde von 2009 bis 2011 in Lausanne (Schweiz) durchgeführt. In diesem Zeitraum wurden die 185 Teilnehmer mit Smartphones ausgestattet, auf denen eine speziell entwickelte Anwendung zur Erfassung von Sensordaten installiert wurde, die die einzelnen Sensoren in unterschiedlichen Intervallen abgefragt hat (vergleiche [KBD+10] und [LGPA+13]). In Bezug auf die hier vorliegende Problemstellung umfasst dies die folgenden Informationen, welche zusätzlich in Anhang B dieser Arbeit detailliert sind:

- Den Status der einzelnen Schnittstellen zu Mobilkommunikation (GSM, WLAN und Bluetooth): die von diesen Geräten jeweils erkannten Funkzellen, Zugangspunkte oder im Fall von Blueooth erkannten Geräte, jeweils zusammen mit der erfassten Signalstärke.
- Den Ort und die Bewegung: Informationen über den Beschleunigungssensor, die GPS-Daten und die Information zur Sichtbarkeit der geolokalisierten WLAN-Zugangspunkte).
- Kalendereinträge und Kommunikation: Die Termine der Nutzer und die Information über Call-Log und SMS-Log.

- Nutzung und Zustand des Gerätes: vergangene Zeit seit der letzten Interaktion mit dem Nutzer und zum aktuell gesetzten Nutzungsprofil, zum Ladezustand, zum verbleibenden Energievorrat und zum verbleibenden freien Arbeitsspeicher.
- Genutzte Anwendungen: Die Anwendung, die der Nutzer genutzt und welche spezifischen Funktionen er daran aufgerufen hat.
- Aktivität: Ermittelt über den Beschleunigungssensor und zugeordnet zu Aktivitäten durch ein Erkennungsprogramm.

Da der LDDC-Datensatz damit die höchste Abdeckung der erforderlichen Kontextattribute besitzt, soll dieser zur Durchführung der Simulation ausgewählt werden.

7.2.2.3. Ergänzung des Datensatzes um fehlende Kontextattribute

Obwohl der LDDC-Datensatz eine vergleichsweise hohe Abdeckung der im Rahmen der in Abschnitt 5.5.3 ausgewählten Kontextattribute aufweist, fehlen jedoch relevante Attribute wie die Bandbreite und die Ausführungsstatistik mobiler Anwendungen. Diese Kontextdaten sollen im Folgenden ergänzt werden.

Bandbreite der WWAN-/WLAN-Schnittstelle Die in der LDDC erhobenen Daten weisen eine Vielzahl der als relevant definierten Kontextattribute auf; allerdings enthalten sie in Bezug auf die unterschiedlichen Schnittstellen zur Mobilkommunikation keine Information zur erzielbaren oder erzielten Bandbreite. Es ist anzunehmen, dass auf diese Messung wegen des hohen Energie- und Bandbreitenverbrauches verzichtet wurde.

Eine Information, die näherungsweise Aussagen im Hinblick auf die erzielbare Bandbreite zulässt, ist allerdings im LDDC-Datensatz vorhanden: die Signalstärke der einzelnen Gegenstellen, die in Reichweite der jeweiligen Schnittstellen zur Mobilkommunikation sind. Dieser Zusammenhang lässt jedoch nur grob eine Aussage auf die erzielbare Bandbreite zu, im Wesentlichen, da die Signalstärke auf einem *Shared-Medium* nur bedingt eine Schlussfolgerung bezüglich der für ein einzelnes Gerät verfügbaren Bandbreite ermöglicht.

Entsprechend wurde eine mobile Anwendung zur Erfassung der Signalstärke und aktiven Ermittlung der verfügbaren Bandbreite entwickelt. Für die Messung der Bandbreite auf der WWAN- und WLAN-Schnittstelle wurden zunächst Messwerte für den Download und Upload ermittelt[3]. Das konkrete Vorgehen wird im Folgenden beschrieben:

WWAN-Schnittstelle Zur Ermittlung der Bandbreite der WWAN-Schnittstelle wird der zu dem Zeitpunkt der LDDC genutzte UMTS-Standard auf das heutzutage übliche LTE Advanced aktualisiert, um den aktuellen

[3] Die Messungen wurden mit einem Samsung Galaxy S5 durchgeführt.

Stand der technischen Entwicklung widerzuspiegeln. In diesem Zusammenhang wurden an unterschiedlichen Orten mit LTE-Versorgung zu verschiedenen Tageszeiten Messwerte erhoben und gemittelt. Die resultierende Abhängigkeit zwischen Signalstärke und Bandbreite wird als orangefarbene Kurve in Abbildung 7.5 veranschaulicht.

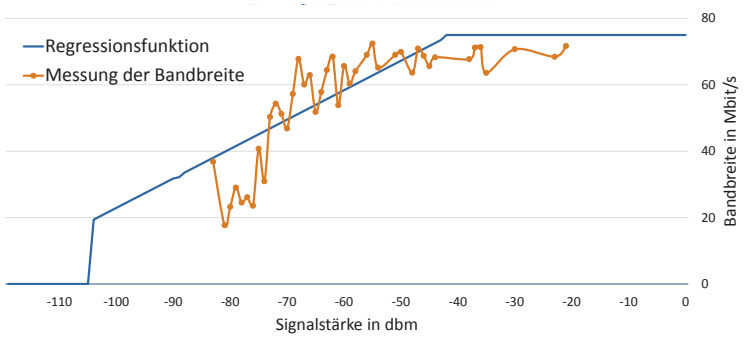

Abbildung 7.5.: Ermittlung der WWAN-Bandbreite

Anschließend wurde die *Arbitrary Strength Unit* (ASU), welche eine Form der Dienstqualität mit Bezug auf die erreichbare Signalstärke widerspiegelt [LTE16], abgeschätzt, um bei einer zu geringen Signalstärke dahingehend Annahmen zu treffen, dass das mobile Gerät in diesem Fall über weitere nutzbare Mobilfunkstandards wie EDGE verbunden war; als Datenquelle dienten ebenfalls die ASU-Werte aus [LTE16]. Bei mehreren empfangenen Signalen wurde das stärkste verfügbare verwendet, unter der Annahme, dass sich ein mobiles Gerät üblicherweise zu dieser Basisstation verbindet. Eine Einschränkung auf einen bestimmten Netzbetreiber konnte hierbei nicht vorgenommen werden, da hierzu im Datensatz keine Information vorlag. Zur Interpolation fehlender Messwerte wurde anschließend eine lineare Regression durchgeführt, um im Anschluss eine Korrektur im Hinblick auf die beim LTE-Standard minimal nutzbaren Signalstärken [LTE16] vorzunehmen - diese wird durch die blaue Kurve in Abbildung 7.5 dargestellt.

Zusätzlich zeigt Abbildung 7.6 die mit diesem Verfahren ermittelten Mittelwerte der verfügbaren Bandbreite für eingehenden Datenverkehr im Tagesverlauf und pro Wochentag.

In Bezug auf den ebenso zu ermittelnden ausgehenden Datenverkehr konnte nicht auf die Messwerte zurückgegriffen werden, da diese durch eine Beschränkung des verwendeten Mobilfunktarifs limitiert waren. Die aktuelle Version des Mobilfunkstandards LTE Advanced unterstützt jedoch typischerweise ausgehenden Datenverkehr bis zur Hälfte der maximalen Bandbreite des eingehenden Datenverkehrs; entsprechend soll hier die Annahme gelten, dass jeweils die Hälfte der eingehenden Datenrate realisiert werden kann. Zusätzlich wurde die verfügbare Bandbreite der WWAN-Schnittstelle tageszeitabhängig reduziert, um zu berücksich-

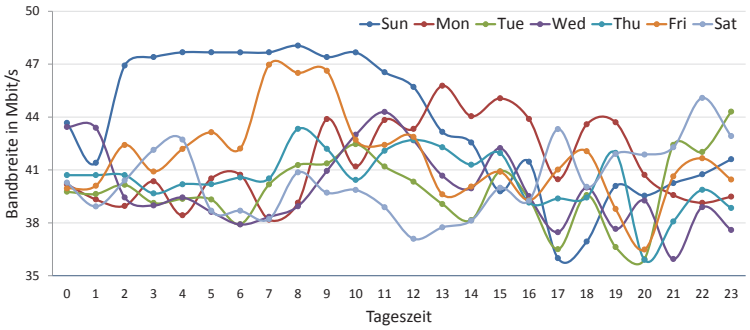

Abbildung 7.6.: Ermittlung der WWAN-Bandbreite

tigen, dass ein Netzbetreiber nicht notwendigerweise ausreichend Kapazität für Lastspitzen vorhält.

WLAN-Schnittstelle Zur Abbildung der Zusammenhänge zwischen Signalstärke und Bandbreite liefert der Datensatz für die WLAN-Schnittstelle ebenfalls keine direkte Information zur Bandbreite oder in Bezug darauf, mit welchem Zugangspunkt das mobile Gerät verbunden war. In diesem Fall erfolgt die Ermittlung analog dem für die WWAN-Schnittstelle beschriebenen Vorgehensmodell, unterstellt wurde hier der Standard 802.11ac.

Konnektivität Die hierdurch gewonnenen Zusammenhänge zwischen Signalstärke und realisierbarer Bandbreite werden im Weiteren als Ergänzung der in der LDDC hinterlegten Informationen genutzt. Für die WWAN-Schnittstelle wird im Datensatz nur die verbundene Funkzelle ausgewiesen.

Für die WLAN-Schnittstelle werden jedoch alle periodisch gefundenen drahtlosen Netzwerke herangezogen. Für die Antwort auf die Frage, ob ein mobiles Gerät über eine gültige Zugangsberechtigung zu einem Zugangspunkt verfügt, erfolgt zunächst die Ermittlung der drei am häufigsten beobachteten Zugangspunkte. Hierzu werden die gefundenen MAC-Adressen der letzten 90 Tage herangezogen und für diese Zugangspunkte wurde eine Verbindung zum Internet und eine Zugangsberechtigung des Nutzers unterstellt.

Dementsprechend gilt es, eine weitere Annahme zu treffen, ob das gefundene drahtlos Netzwerk ein bekanntes ist und der Nutzer einen gültigen Zugang besitzt. Da hierfür keine ergänzenden Informationen vorliegen, wird für die Generierung des Kontextattributs, das aussagt ob ein Nutzer mit einem WLAN tatsächlich verbunden ist, die Information herangezogen, ob es sich beim erkannten Zugangspunkt um eine der häufigsten beobachteten MAC-Adressen der letzten 90 Tage handelt.

Bandbreite Basierend auf den vorhergehenden Annahmen wird anschließend die Konnektivität zu unterschiedlichen Surrogates ermittelt, wobei in Be-

zug auf die WLAN Zugangspunkte die Limitierung des jeweils dazwischenliegenden Breitbandanschlusses berücksichtigt wurde, welche in Richtung des mobilen Gerätes auf 100 Mbit/s und in Gegenrichtung auf 40 Mbit/s limitiert wurde. Gleichzeitig wird die hierdurch angenommene Bandbreite bei einer erkannten Verbindung zum zweit- und dritthäufigsten erkannten WLAN-Zugangspunkt halbiert, um der unterschiedlichen Breitbandversorgung an unterschiedlichen Orten Rechnung zu tragen.

Auf diese Weise kann dem Datensatz der LDDC, der nur Signalstärken jedoch keine Informationen zur tatsächlich realisierbaren Bandbreite oder zum Verbindungsstatus eines mobilen Gerätes enthält, diese Informationen hinzugefügt werden. Stehen beide Verbindungen zur Verfügung, wird angenommen, dass stets vorrangig die WLAN-Schnittstelle verwendet wird, wie dies von mobilen Geräten üblicherweise umgesetzt wird.

Ausführungsstatistik mobiler Anwendungen Eine weitere Kontextinformation, welche im Datensatz nicht vorhanden ist und an dieser Stelle generiert werden soll, bezieht sich auf die Nutzung mobiler Anwendungen und die im Rahmen der Interaktion mit dem Nutzer initiierten Tasks dieser mobilen Anwendung. Als Grundlage zur Generierung dieser Ausführungsstatistik auf Ebene der Tasks werden die im Datensatz vorhandenen Informationen zur Nutzung zur jeweils angezeigten Maske unterschiedlicher Anwendungen herangezogen. Hierzu erfolgt eine Gruppierung der vorhandenen Daten in Kategorien wie unter anderem Bildverarbeitung, Navigation, Webbrowsing, Messaging so wie übrige Nutzerinteraktionen.

Für die exemplarisch herangezogene Kategorie der Bildverarbeitung wurden die benötigten Informationen zur Ausführungsstatistik der Tasks mithilfe der vorgestellten Beispielanwendung generiert. Das Ergebnis in Tabelle 7.4 zeigt exemplarisch die daraus resultierenden und für die Simulation der Nutzungsmuster herangezogene Nutzungsstatistik der simulierten Beispielanwendung zur Bildverarbeitung. Beispielsweise hat Nutzer 5479 am 21.6. um 14:30 Uhr in einer Anwendung zur Bildverarbeitung die Android-View *[0000004b]* aufgerufen, welche einen Dienstaufruf an den Dienst *service1* initiiert hat.

USERID	INVOCATIONTIME	TIME_LOCALIZED	USAGEPROFILE	CALLINGAPP	VIEWID	CALLEDSERVICE
5479	21.06.2020 14:30	1592749850	0	Photo	[0000004b]	service_1
5479	21.06.2020 14:17	1592749051	0	Photo	[0000004a]	service_1
5479	21.06.2020 12:25	1592742328	0	Photo	[0000004c]	service_1
5479	21.06.2020 11:56	1592740583	0	Photo	[0000000a]	service_2

Tabelle 7.4.: Ausschnitt der generierten Nutzungsstatistik

Die Annahmen zu den im Rahmen dieser Nutzungsstatistik übertragenen Datenmengen basieren auf den für die Kamera des verwendeten mobilen Geräts typischen Größen für ein JPEG-Bild.

Verfügbarkeit und Auslastung der Infrastruktur Da aktuell noch keine Cloudlet-Infrastruktur existiert, wird die Verfügbarkeit eines Cloudlets analog zum WLAN-Verbindungsstatus angenommen. Allerdings wird die Verfügbarkeit des Cloudlets als schwankend innerhalb des Tagesverlaufs betrachtet.

Abbildung 7.7 zeigt hierzu den prozentualen Anteil der in einer Stunde eines Tages auf den mobilen Geräten ausgeführten Anwendungen aus dem LDDC-Datensatz. Diese werden dafür herangezogen, die Auslastung der Cloudlet-Infrastruktur und hierdurch die tagezeitsabhängige Verarbeitungszeit sowie eventuelle Überlastsituationen zu simulieren.

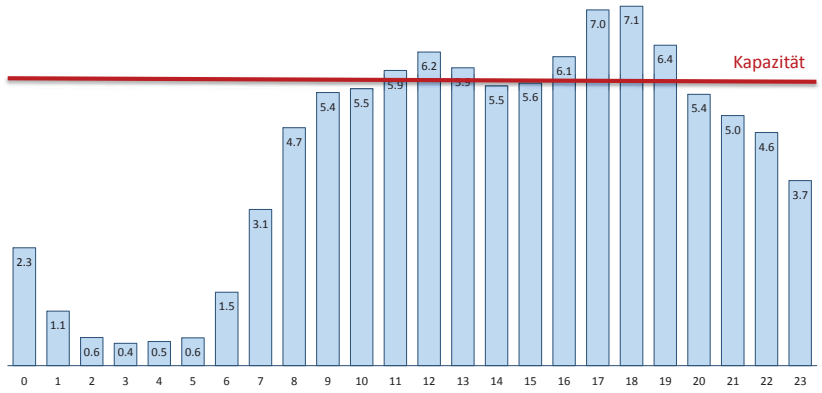

Abbildung 7.7.: Nutzungsstatistik zur Ermittlung der Auslastung der Cloudlet-Infrastruktur: prozentualer Anteil pro Stunde des Tages

7.2.2.4. Abgleich mit identifizierten Kontextattributen

Nachdem der Datensatz um relevante Kontextattribute ergänzt wurde, soll im Folgenden veranschaulicht werden, inwieweit die existierende Datenbasis die in Abschnitt 5.5 für das Konzept der generischen Kontextadaption ermittelten relevanten Kontextattribute beinhaltet. Tabelle 7.5 fasst hierzu die einzelnen Zielgrößen der Prognosen und die zuvor ermittelten relevanten Kontextattribute zusammen und gleicht deren Vorhandensein in Bezug auf den zuvor erweiterten Beispieldatensatz aus der LDDC ab.

Ein spezielles Kontextattribut stellt hierbei der verbleibende Energievorrat des mobilen Gerätes dar, da dieser sich, sofern das Gerät nicht mit dem Netz verbunden ist, sich durch die simulierte Ausführung der Bildverarbeitungsanwendung in darauffolgenden Kontextintervallen verändert.

Dies wiederum führt dazu, dass das mobile Gerät in bestimmten Zeiträumen als nicht verfügbar markiert wird, was durch den zusätzlichen Energiebedarf der simulierten Anwendungsnutzung begründet ist. Zusätzlich existieren weitere Zeiträume, in denen das mobile Gerät als nicht verfügbar angesehen wird. Diese werden daraus abgeleitet, dass im Datensatz keine Kontextinformationen für das jeweilige Zeitintervall verfügbar sind.

	Attribut(-Gruppen)	Beispiel	verfügbar im LDDC
Ort	DistanceFromTop[N=1..3]Location	325	nein (GPS-Daten unpräzise)
	IsAtTopPlace[N=1..3]	[Yes,No,No]	ja
Zeit	BankHoliday	No	ja (zu ermitteln)
	Month[N=1..12]	[No,..,Yes,No]	ja
	SchoolVacation	No	nein
	TimeHour[N=0..23]	[No,..,Yes,No]	ja
	TimeSlot[N=1..6]	[No,..,Yes,No]	ja
	TimeUntilLeave	17	ja
	Weekday[N=1..7]	[No,..,Yes,No]	ja
	Weekend	No	ja
Kalender	CalendarInEvent	No	ja
	CalEventEndingIn[N=10,20,..60]	No	ja
	CalEventIn[N=10,20,60]InTop[M=1..3]	[No,..,No]	ja
	CalLeaveTime	0	ja
Aktivität	ActivityEvent	[running, sitting, ..]	nein (nur RAW-Daten)
	BatteryCharging	No	ja
	BatteryLevel	100	nein
	BatteryLevel[N=10,20,70]	[Yes,Yes,Yes]	ja
	CallReceivedLast[N=10,20,60]	19	ja
	ChargingCompleted	Yes	ja
	FreeMemory	79321	ja
	InactiveTime	17	ja
	ScreenOn	No	ja
	SilentModeSwitch	On	ja
WPAN	BluetoothActivated	Yes	ja
	BluetoothMacCount	12	ja
WWAN	GSMAverageStrength	-97 dbm	ja
	ConnectionType	3G	nein
	GSMActivated	1	ja
	GSMBars	7	ja
	GSMConnected	Yes	ja
	GSMLeavingIn	15	ja (zu ermitteln)
	GSMSignalCount	3	ja
	GSMTop[N=1..3]Visible	[Yes,No,No]	ja (zu ermitteln)
	GSMTop3Visible	Yes	ja
WLAN	WiFiAverageStrength	-67 dbm	ja
	WiFiActivated	Yes	ja
	WiFiConnected	Yes	ja
	WiFiLeavingIn	38	ja (zu ermitteln)
	WiFiSignalCount	3	ja
	WiFiVisibleTop[N=1..3]	[Yes,No,No]	ja (zu ermitteln)
Band-breite	BandwidthDown	4332	Abschätzung aus Signalstärke
	BandwidthUp	1700	Abschätzung aus Signalstärke
	Ausführungsstatistik	-	Ermittelt aus LDDC-AppUsage

Tabelle 7.5.: Abgleich mit identifizierten Kontextattributen

7.2.2.5. Ergebnisse der Simulation

Nachdem das Simulationsmodell entwickelt und eine entsprechende Datenbasis für die Simulation aufbereitet wurden, sollen zur Beurteilung der Leistungsfähigkeit des entwickelten Ansatzes im Folgenden ausgewählte Parameter der Simulation variiert werden, um zu zeigen, in welchen Fällen die unterschiedlichen Ziele der Adaption zu welchem Grad erreicht werden können. Hierzu werden ausgehend von einem Basis-Szenario nacheinander die einzelnen Parameter der Simulation so variiert, dass sie die Beantwortung der in diesem Abschnitt betrachteten Fragestellungen unterstützen, die sich grob an den drei vorgestellten Formen der Adaption orientieren.

Die einzelnen Ergebnisse werden dabei sowohl für einen exemplarisch ausgewählten Nutzer *(UserID: 5479)*, der auch für die Beschreibungen herangezogen wird, als auch für die im Rahmen der Simulation ausgewählten 20 Nutzer gezeigt, um den Aspekt der Generalisierungsfähigkeit zu belegen. Hierzu bildet das in Tabelle 7.6 gezeigte Basisszenario die Grundlage für die unterschiedlichen Varianten der Simulation, die im Folgenden sukzessive untersucht werden.

Anwendung	Verarbeitungszeit	Übertragene Daten	Anzahl Threads	% der lokalen Laufzeit	Fidelity Adaptation
Bildverarbeitung	20 sec.	10 MB	1	100%	nein

Tabelle 7.6.: Basisszenario

Zu untersuchende Fragestellung Die wichtigste Fragestellung im Zusammenhang mit der Simulation der Architektur lautet: Ist die entwickelte Anwendungsarchitektur in der Lage, sinnvoll eine Kooperation mit der Infrastruktur zu realisieren:

Generell ist eine Auslagerung von Funktionalität nur dann sinnvoll, wenn sie im Hinblick auf das jeweilige Ziel der Adaption ein besseres Verhältnis zwischen der zusätzlichen Kosten der Nutzung von Ressourcen für die Synchronisation mit der Infrastruktur und dem erwarteten Nutzen im Hinblick auf das jeweilige Ziel der Adaption, üblicherweise eine Einsparung von Ressourcen auf einem mobilen Gerät, liefert. Entsprechend soll untersucht werden: Welchen Einfluss haben der zu synchronisierende Zustand und die verfügbaren Ressourcen wie die Bandbreite und der Energievorrat eines mobilen Gerätes auf die erfolgreiche Ausführung von Tasks - das den unterschiedlichen Zielen der Adaption vorgelagerte Ziel?

Basisszenario Anhand des in Tabelle 7.6 beschriebenen Basisszenarios soll zunächst die Struktur der Auswertung beschrieben werden, die sich durch den Großteil der folgenden Simulationen zieht. Im Hinblick auf die unterschiedlichen Adaptionsstrategien wurde das übergreifende Ziel definiert, die Anzahl der ausgeführten Tasks zu maximieren. Der entsprechende Parameter *success rate* bildet dieses Kriterium als Quotient der erfolgreich ausgeführten Tasks im

Abgleich zur Gesamtmenge der simulierten Tasks ab. Als weitere wesentliche Kennzahlen sind dabei jeweils zusätzlich die durchschnittliche Ausführungszeit der erfolgreich ausgeführten Tasks, die pro erfolgreich ausgeführtem Task aufgewendete Menge an Energie und Bandbreite dargestellt. Abbildung 7.8 zeigt diese, im Zusammenhang mit den Zielen der Adaption als relevant identifizierten Kennzahlen.

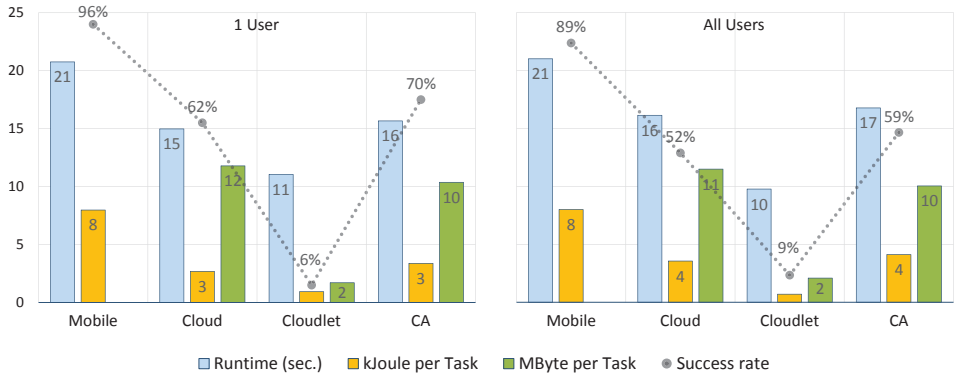

Abbildung 7.8.: Basisszenario der Simulation

Diese Darstellung ergänzend zeigt Abbildung 7.9 einen Boxplot für jede dieser vier Kennzahlen, um die Verteilung der Messergebnisse zu verdeutlichen. Für die Darstellung wird die von Tukey in [TUK77] vorgeschlagene Begrenzung der Whisker auf Basis des 1,5-fachen Interquartilsabstandes verwendet. Die Punkte markieren entsprechend einzelne Ausreißer, wobei deren Färbung der Zuordnung zum jeweiligen Nutzer entspricht. Dargestellt werden hierzu vier verschiedene Szenarien:

- *Mobile*: Sämtliche Tasks werden auf dem mobilen Gerät ausgeführt. Die Anzahl erfolgreich ausgeführter Tasks ist im Wesentlichen dadurch limitiert, dass die simulierte Anwendungsnutzung einen zusätzlichen Energieverbrauch verursacht. Dies wiederum kann dazu führen, dass neben den mit Hilfe der Kontextdaten simulierten Zustände weitere Zustände eintreten können, in denen der Energievorrat des mobilen Geräts erschöpft ist. Für den ausgewählten Nutzer liegt hierdurch der Anteil erfolgreich ausgeführter Tasks bei 96 %.

- *Cloud* und *Cloudlet*: Alle Tasks werden auf dem entsprechenden Surrogate ausgeführt. Der Anteil erfolgreich ausgeführter Tasks ist im Wesentlichen dadurch limitiert, dass die Verbindung zu den jeweiligen Surrogates nicht dauerhaft vorhanden ist, sondern nur dann, wenn das mobile Gerät mit der Infrastruktur verbunden ist. Als obere Schranke für die zulässige Verarbeitungszeit wurde für diese Szenarien die durchschnittliche Verarbeitungszeit dieses Tasks auf dem verwendeten mobilen Gerät herangezogen. Hierdurch soll sichergestellt werden, dass die Verarbeitung in der Infrastruktur nicht länger als auf dem mobilen Gerät dauert.

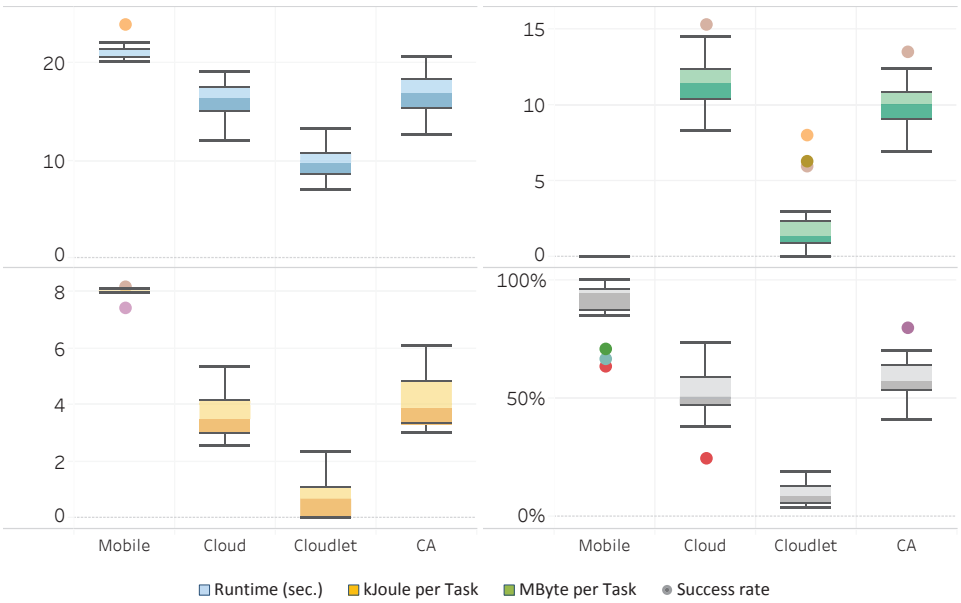

Abbildung 7.9.: Basisszenario der Simulation: Verteilung der Messergebnisse

- *CloudAware (CA)*: Zur Ausführung von Tasks werden das mobile Gerät sowie alle verfügbaren Surrogates verwendet und es wird anhand der durchschnittlichen Ausführungszeit und Bandbreite ermittelt, welche Alternative das im Hinblick auf das Ziel der Adaption gewählte Optimierungsziel bestmöglich umsetzen kann. Als obere Schranke der zulässigen Verarbeitungszeit wird hier ebenfalls die durchschnittliche Verarbeitungszeit des Tasks auf dem mobilen Gerät herangezogen.

Varianten des Basisszenarios Nachdem dieses Basisszenario erläutert wurde, soll nun zunächst der Parameter der Verarbeitungszeit variiert werden, um aufzuzeigen, wie sich für die einzelnen Fälle der Nutzen einer kooperativen Ausführung im Abgleich mit einer rein lokalen Verarbeitung der Tasks darstellt. Abbildung 7.10 zeigt hierzu unterschiedliche Verarbeitungszeiten, die gegenüber dem Basisszenario (20 Sekunden Laufzeit) jeweils halbiert und verdoppelt wurden. Zusätzlich wurde eine Variante mit 400 Sekunden Verarbeitungszeit simuliert.

Hierbei zeigt sich, dass bei einem zu übertragenden Zustand von 10 MB eine Kooperation mit der Infrastruktur ab einer Verarbeitungszeit von 20 Sekunden sinnvoll realisiert werden kann: So ergibt sich für das Szenario *CloudAware* gegenüber einer rein mobilen Verarbeitung (Szenario: Mobile) bereits eine deutliche Annäherung im Anteil ausgeführter Tasks bei annähernd gleicher Verarbeitungszeit aber weniger als der Hälfte der benötigten Energie im Vergleich zur lokalen Ausführung der Tasks. Längere Verarbeitungszeiten verstärken diesen Effekt, sodass der Nutzen der Auslagerung entsprechender Tasks zunimmt. Dies kann vor allem dadurch begründet werden, dass der Anteil der Zeit, der für die Synchronisation mit dem Surrogate aufgewendet wird, sich

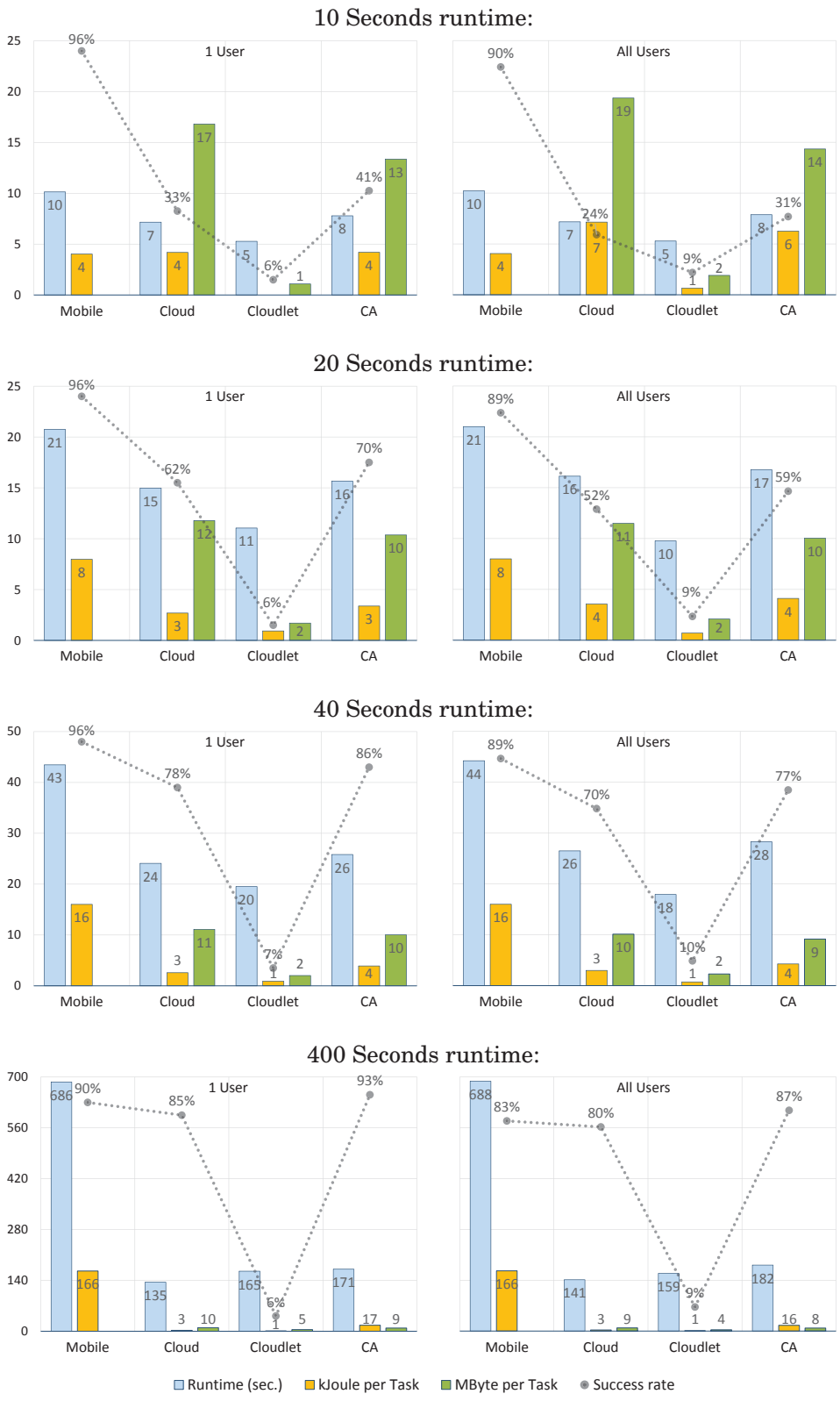

Abbildung 7.10.: Simulation der Abhängigkeit von der Verarbeitungszeit

bei höheren Verarbeitungszeiten verringert. Zusätzlich sorgt die höhere Verarbeitungsgeschwindigkeit der Surrogates dafür, dass das Abbruchkriterium, die Überschreitung der lokalen Verarbeitungszeit, seltener eintritt. Hierdurch ergibt sich bei längeren Verarbeitungszeiten für die kooperative Ausführung stets eine höhere Wahrscheinlichkeit dafür, dass Tasks erfolgreich ausgeführt werden. Dieser Effekt zeigt sich bei 400 Sekunden Verarbeitungszeit aus und sorgt für eine, in diesem Fall auch gegenüber der lokalen Ausführung, höhere Ausführungswahrscheinlichkeit. Dies liegt unter anderem darin begründet, dass der Energievorrat des mobilen Geräts bei Verarbeitungszeiten dieser Größenordnung entsprechend häufiger erschöpft ist.

Ein ähnliches Bild wie bei der Variation der Verarbeitungszeit zeigt sich in Abbildung 7.11, wenn der zu übertragende Zustand variiert wird. Hier ist zu erkennen, dass eine Halbierung der zu übertragende Datenmenge bereits zu deutlich höheren Ausführungswahrscheinlichkeiten bei der kooperativen Taskausführung führt. Ist die Datenmenge jedoch zu hoch, zeigt sich entsprechend ein gegenteiliger Effekt.

Wie bereits zuvor erwähnt ist die obere Schranke der zulässigen Verarbeitungszeit oft das entscheidende Kriterium dafür, dass Tasks nicht erfolgreich abgeschlossen werden. Wird dieser Parameter beispielsweise zugunsten einer energieeffizienten Ausführung von Tasks erhöht, so lassen sich auch für kurze Verarbeitungszeiten bzw. größere Mengen eines zu synchronisierenden Zustands kooperative Taskausführungen sinnvoll umsetzen. Abbildung 7.12 zeigt diesen Zusammenhang, bei dem sich bei der Erhöhung der maximal zulässigen Verarbeitungszeit eine Verbesserung der Anzahl erfolgreich ausgeführter Tasks zeigt.

Um weitergehend zu belegen, dass die entwickelte Anwendungsarchitektur in der Lage ist, sinnvoll eine Kooperation mit der Infrastruktur zu realisieren, soll abschließend gezeigt werden, inwiefern Anwendungen, die für eine Parallelverarbeitung entwickelt wurden, von einer kooperativen Ausführung profitieren. Obwohl das mobile Gerät einen Vierkernprozessor besitzt, konnte bei länger andauernder Nutzung aller vier Recheneinheiten lediglich eine Beschleunigung um den Faktor drei gegenüber dem in Tabelle 7.6 gezeigten Basisszenario[4] ermittelt werden (vergleiche [Bes16]).

Entsprechend zeigt sich in Abbildung 7.13, dass die Parallelverarbeitung ebenfalls ein wirksamer Hebel ist, um in Situationen, in denen eine kooperative Ausführung möglich ist, diese sinnvoll zu nutzen und den Anteil erfolgreich ausgeführter Tasks dabei auf dem Niveau der lokalen Ausführung zu halten.

Simulation der Fidelity Adaptation Nachdem die generelle Umsetzbarkeit der in dieser Arbeit vorgeschlagenen Architektur gezeigt wurde, soll untersucht werden, welche zusätzlichen Faktoren die Ausführungswahrscheinlichkeit von Tasks positiv beeinflussen können. In Abbildung 7.14 und 7.15 wird hierzu dargestellt, wie eine Anpassung der Implementierung die Ausführungs-

[4] Die Verarbeitungszeit wurde auf 40 Sekunden verdoppelt, um den Effekt besser zeigen zu können.

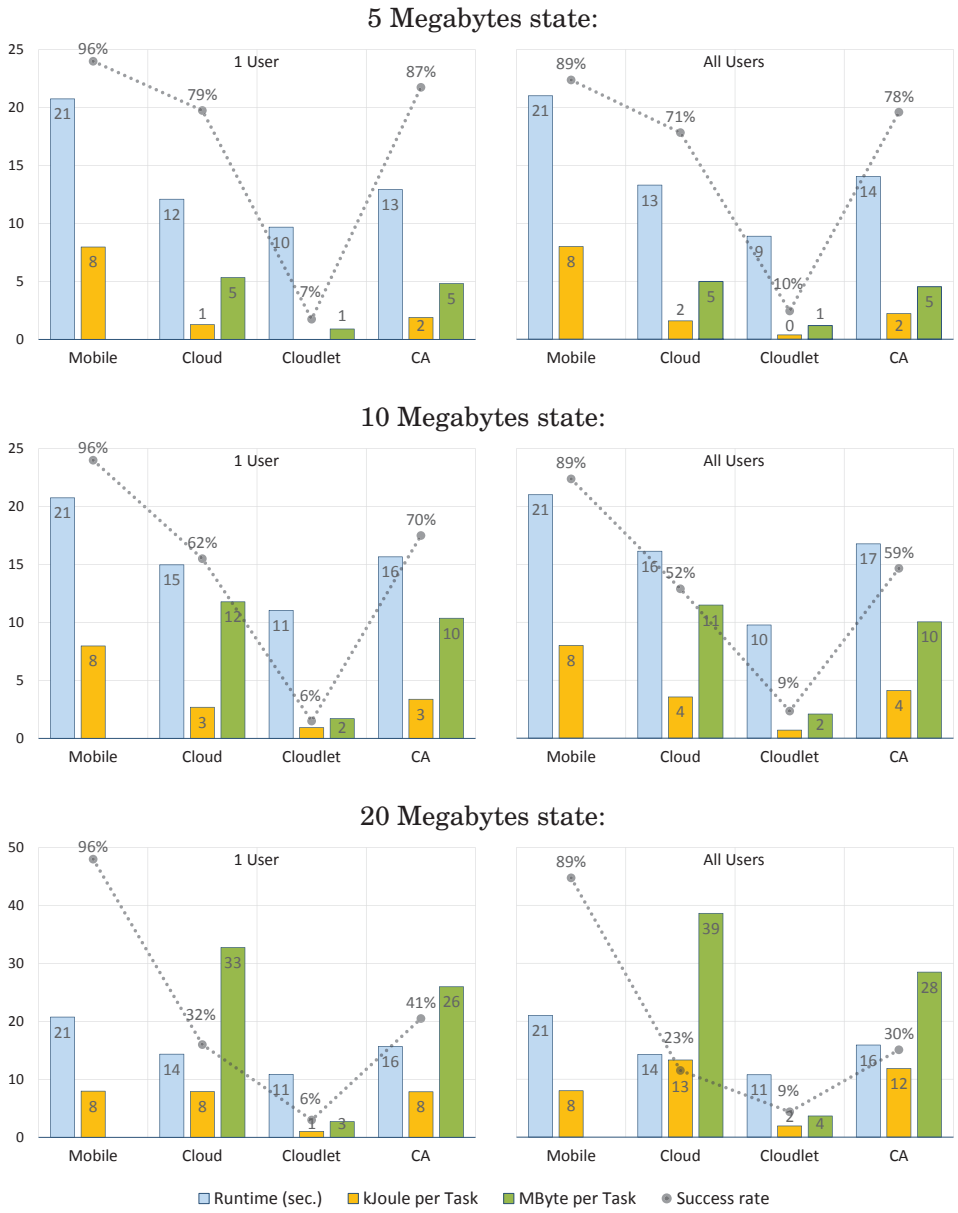

Abbildung 7.11.: Simulation der Abhängigkeit vom zu übertragenden Zustand

wahrscheinlichkeit von Tasks erhöhen kann. Der Boxplot zur Variante ohne Fidelity Adaptation findet sich in Abbildung 7.9.

Dabei zeigt sich, dass durch eine entsprechende Anpassung der Implementierung der Anteil erfolgreich ausgeführter Tasks um den Faktor 2,1 erhöht werden kann. Dies geht mit einer gleichzeitigen Reduktion des Energieverbrauchs um 24 % einher (Szenario: 1 User). Dies wiederum belegt, dass die entwickelte kontextadaptive Anwendungsarchitektur dazu genutzt werden kann, die Funktionalität eines mobilen Gerätes dahingehend anzupassen, dass es auch bei stark begrenzten Ressourcen noch eine sinnvolle Kooperation mit der Infrastruktur realisieren kann.

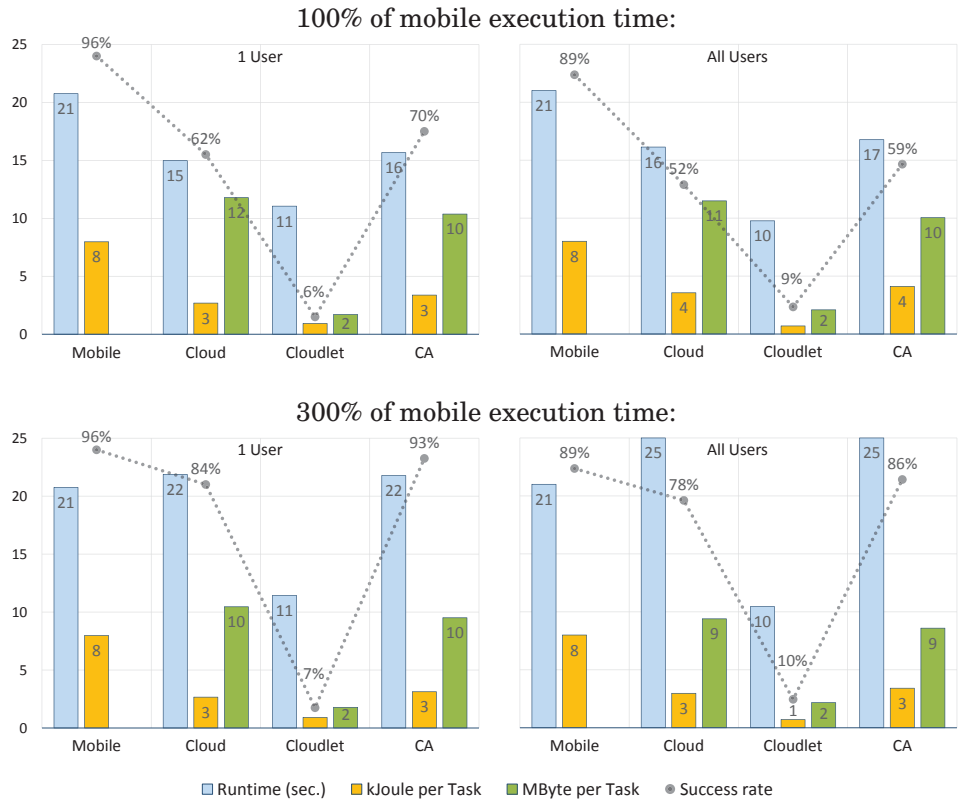

Abbildung 7.12.: Simulation der Abhängigkeit von der zulässigen Verzögerung

Simulation der Fähigkeit zur Kontextadaption In diesem Abschnitt soll gezeigt werden, wie gut ein zukünftiger Kontext vom entwickelten Ansatz antizipiert werden kann, da ein wesentlicher Aspekt des entwickelten Ansatzes in der Fähigkeit liegt, diese Information in die Adaptionsentscheidungen einzubeziehen. Entsprechend soll zunächst die Fähigkeit zur Prognose unterschiedlicher Kontextattribute untersucht werden, bevor im Anschluss gezeigt wird, wie gut hierdurch ein zukünftiger Kontext in Bezug auf die Taskausführung antizipiert werden kann. Abbildung 7.16 zeigt hierzu die Prognosen unterschiedlicher binärer (oberes Diagramm) und reellwertiger (unteres Diagramm) Kontextattribute im Zeitverlauf. Das Fehlermaß der Dabei ist zu erkennen, dass die Prognosequalität mit größer werdendem Zeithorizont zwar abnimmt, sich insgesamt aber in einem für den Anwendungsfall angemessenen Bereich befindet.

Da die Prognosequalität datengetriebener Verfahren allerdings stark von der bereits vorhandenen Datenmenge abhängig ist, soll an dieser Stelle noch einmal der Einsatz des in Abschnitt 6.4 beschriebenen Verfahrens zum Umgang mit der Cold-Start-Problematik gezeigt werden. In Abbildung 7.17 ist hierzu für ein einzelnes binäres Kontextattribut die isolierte Prognosequalität innerhalb einzelner Wochen gezeigt.

In diesem Fall wurde zur vierten Woche vom durschnittsbasiertem Verfahren auf den generischen Prozess zur Kontextadaption gewechselt. Der in

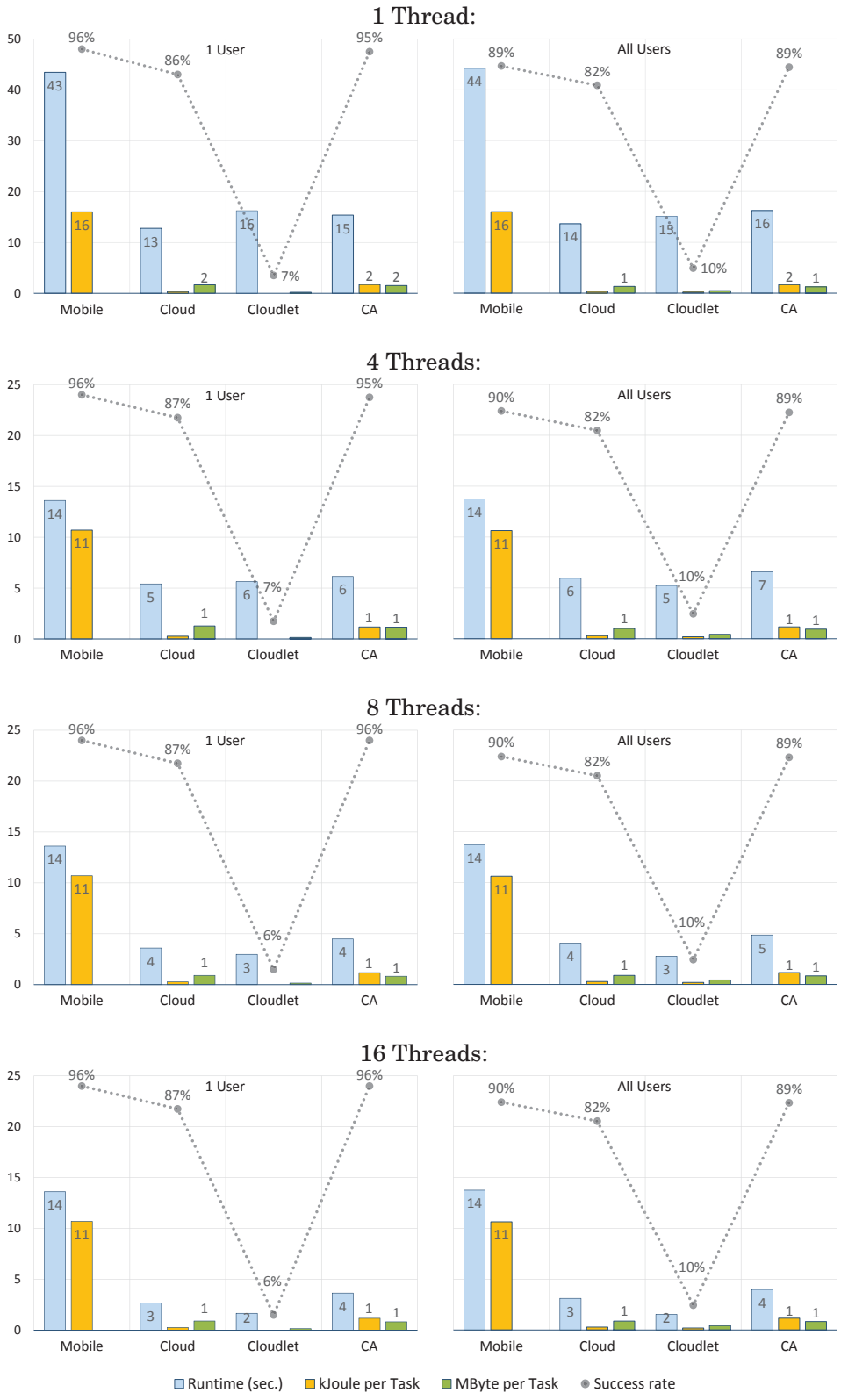

Abbildung 7.13.: Simulation der Parallelverarbeitung

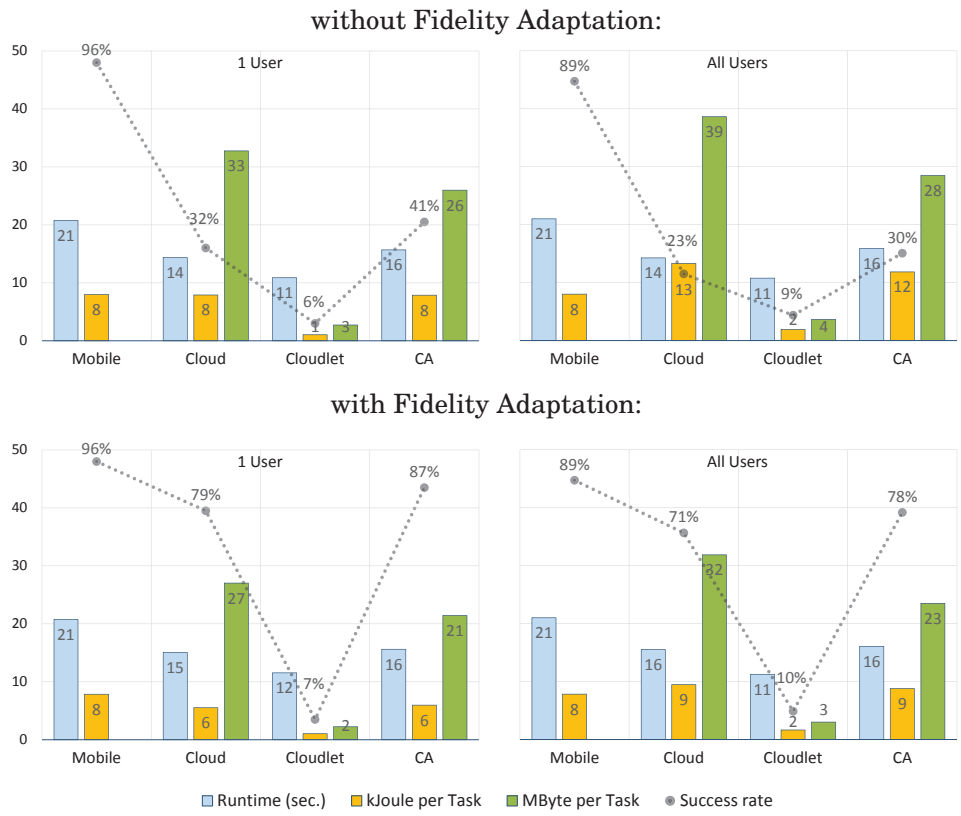

Abbildung 7.14.: Simulation der Fidelity-Adaptation

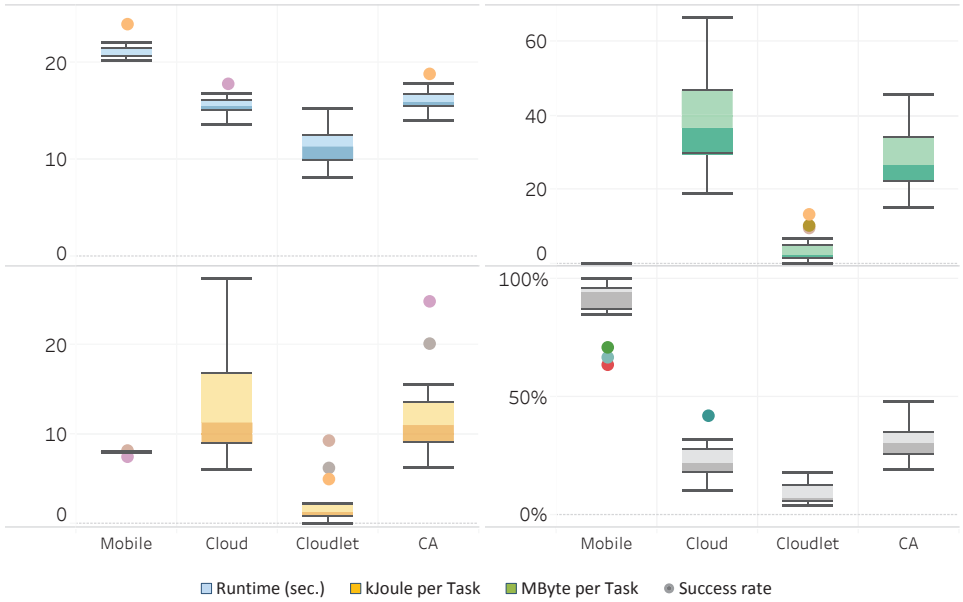

Abbildung 7.15.: Simulation der Fidelity-Adaptation: Verteilung der Messergebnisse für die untere Variante in Abbildung 7.14.

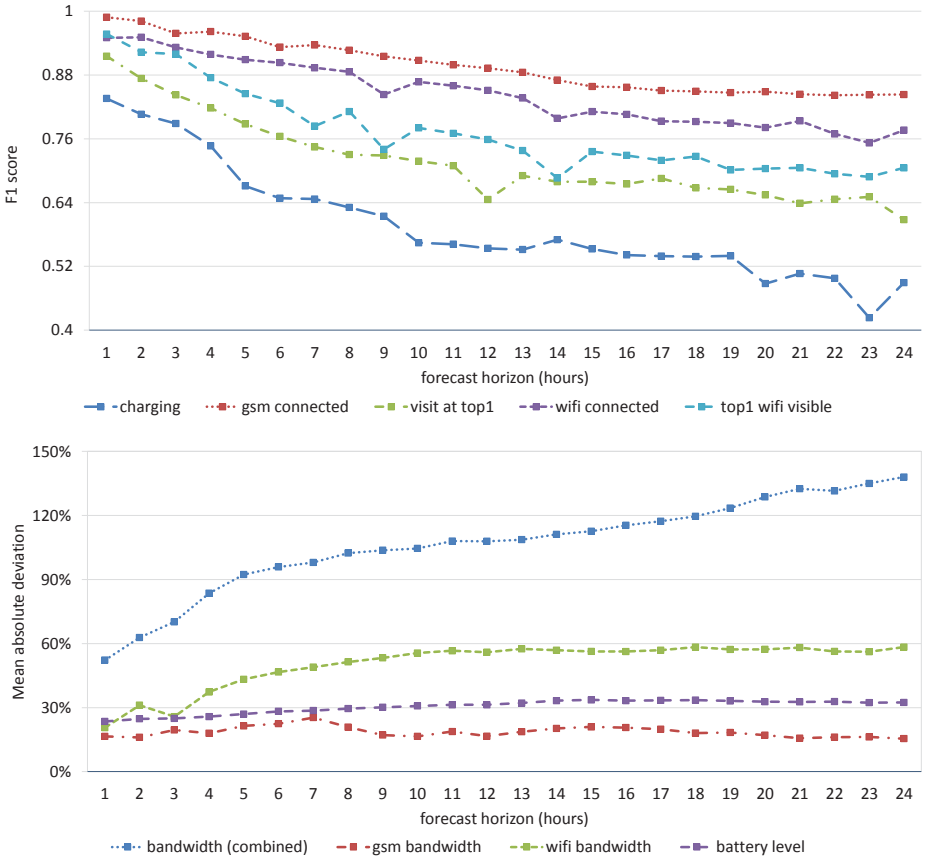

Abbildung 7.16.: Simulation der Prognosequalität einzelner Kontextattribute

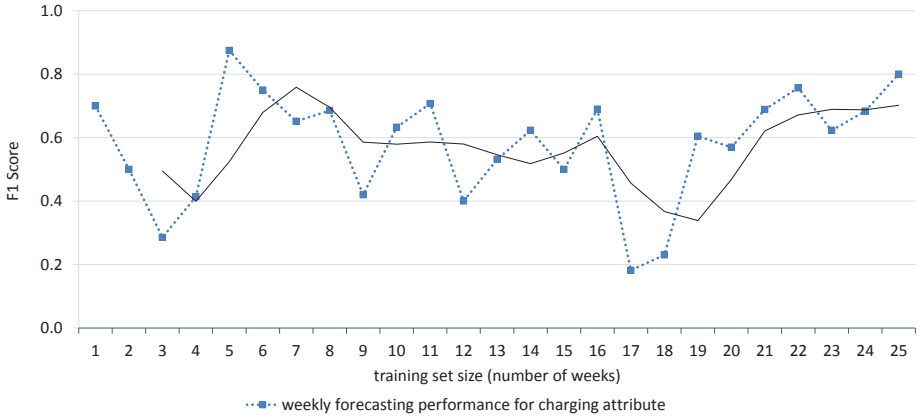

Abbildung 7.17.: Simulation der Prognosequalität einzelner Kontextattribute

der schwarzen Trendlinie gezeigte gleitende Durchschnitt der letzten drei Messwerte zeigt, dass das gewählte zweistufige Verfahren in der Lage ist, bereits bei wenigen Tagen aufgezeichneter Trainingsdaten eine durchschnittliche Prognosequalität zu erreichen. Diese Prognosequalität wird jedoch mit dem Wechsel auf den generischen Prozess zur Kontextadaption noch weiter gestei-

gert. In diesem Fall fand der Wechsel zur vierten Woche statt. Zur Interpretation der Daten ist weiterhin anzumerken, dass die für die Simulation genutzten Kontextdaten darauf hindeuten, dass das mobile Geräte in den ersten Tagen der Nutzung ständig mit einer externen Energiequelle verbunden war. Hieraus resultiert die in den ersten beiden Wochen erreichte, gute Prognosequalität des durschnittsbasierten Verfahrens.

Um hierauf aufbauend zu zeigen, inwieweit die Prognosen in Anpassung von Adaptionsentscheidungen im Zusammenhang mit der kooperativen Taskausführung genutzt werden können, soll zunächst das entsprechende (zusätzliche) Szenario näher beschrieben werden:

- CloudAware unterstützt durch generische Kontextadaption (CA + GCA): Aufbauend auf dem Szenario *CloudAware* wird die Ermittlung passender Adaptionsstrategien von unterschiedlichen Prognosen unterstützt, die mithilfe des Prozesses der generischen Kontextadaption ermittelt werden. Dies umfasst im Wesentlichen den Erfolg der Ausführung eines Tasks, die Ausführungszeit und auch die zu erwartende Konnektivität und Bandbreite.

Abbildung 7.18 zeigt hierzu das in Tabelle 7.6 definierte Basisszenario, ergänzt um die Variante der kooperativen Ausführung, die durch den Prozess der generischen Kontextadaption unterstützt wird. Hierbei zeigt sich, dass sich gegenüber einer lokalen Ausführung eine ähnliche Ausführungswahrscheinlichkeit von Tasks ergibt, jedoch der Energieverbrauch und die Verarbeitungszeit stets etwas, je nach Szenario aber auch deutlich, von dieser Variante der kooperativen Ausführung profitieren.

Abbildung 7.19 zeigt in der Variante „Multiple Applications“ ergänzend, dass das entwickelte Verfahren ebenso in der Lage ist, auch komplexere Zusammenhänge zu erkennen und zu adaptieren. In diesem Fall wurden die Nutzung dreier unterschiedlicher Anwendungen simuliert, welche jeweils fünf in Bezug auf den zu übertragenden Zustand und die Verarbeitungszeit deutlich unterschiedliche Typen von Tasks ausführen.

Abschließend soll differenziert werden, in wie weit das zuletzt eingeführte Szenario (CA + GCA) von einer Kontextadaption profitiert und welcher Teil der Verbesserung sich auf die, mit der vorgeschlagenen Architektur verbundene, Nutzung von Cloud-Ressourcen zurückführen lässt. Das Szenario Cloud und Cloudlet markiert hierzu den einen Extremfall, in dem keine Kontextadaption durchgeführt wird. Der andere Extremfall, eine Kontextadaption die stets korrekte Prognosen hinsichtlich des Erfolgs einer Taskausführung und der erwarteten Ausführungszeit liefert, ist durch das in Abbildung 7.20 gezeigte zusätzliche Szenario *CA Optimal* dargestellt. In diesem Szenario werden entsprechend lediglich die Tasks initiiert, die tatsächlich erfolgreich abgeschlossen werden. Dies bedeutet, dass beispielsweise auch Verbindungsabbrüche und fehlende Energievorräte vorhergesehen werden. So sind der Energie- und Bandbreitenverbrauch in diesem Fall nur halb so hoch und die Ausführungszeit ist ebenfalls leicht besser. Durch einen Vergleich mit dem ersten Extremfall, der rein

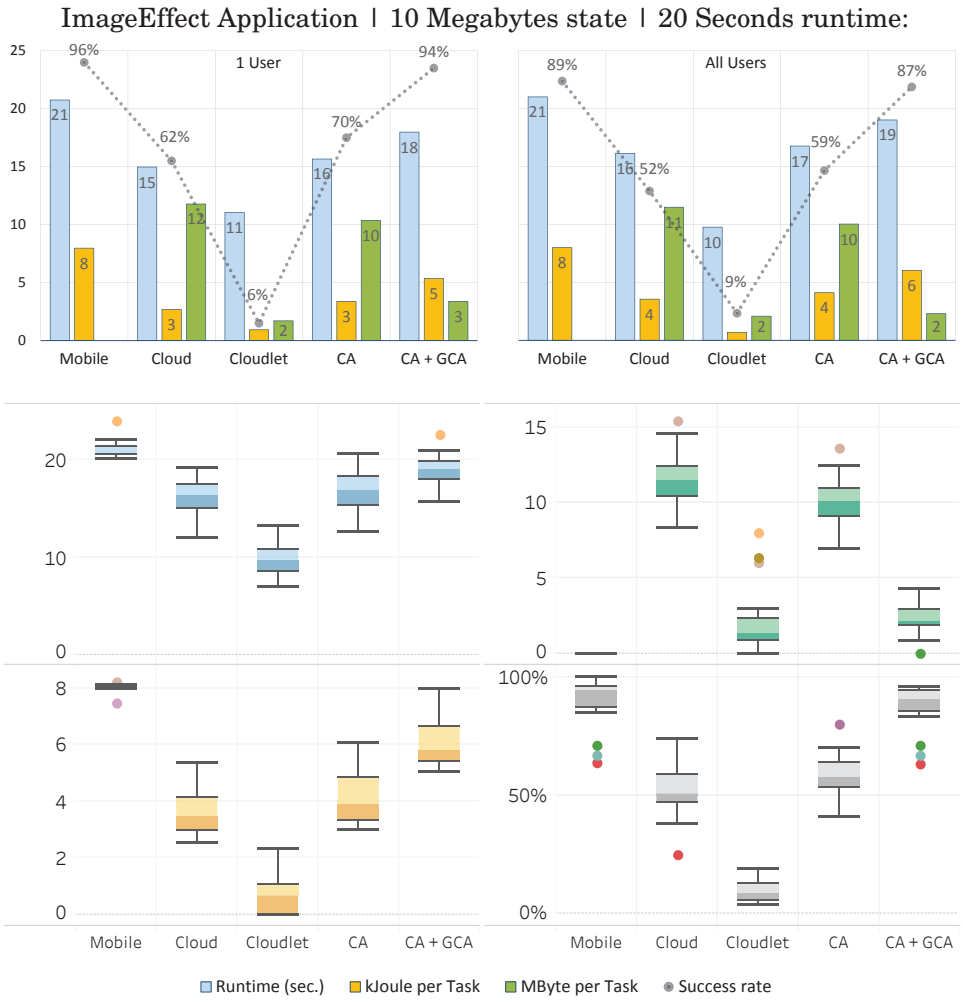

Abbildung 7.18.: Taskausführung einer Anwendung, unterstützt durch GCA mit Verteilung der Messergebnisse

cloudbasierten Ausführung, zeigt sich allerdings ein deutlich höherer Anteil erfolgreich ausgeführter Tasks, weswegen die Fähigkeit zur Kontextadaption insgesamt als sehr gut beurteilt werden kann.

Interpretation der Simulationen Aus den gezeigten Versuchen lässt sich ableiten, dass die Variante (CA + GCA) in der Lage ist, Anwendern eine vergleichbare Dienstverfügbarkeit zu liefern. Gleichzeitig erlaubt die zusätzliche Möglichkeit zur Fidelity-Adaptation und Parallelverarbeitung je nach Anwendungsfall eine deutliche Reduktion des Energieverbrauchs oder der Verarbeitungszeit realisieren zu können. Zu berücksichtgen ist dabei, das die in Abbildung 7.19 gezeigte Simulation mehrerer Anwendungen einen Worst Case darstellt, der durch die Kombination mit einer Fidelity Adaptation (vergleiche Abbildung 7.14) und einer parallelen Ausführung (vergleiche Abbildung 7.13) noch weitaus stärker vom vorgeschlagenen Ansatz profitieren kann.

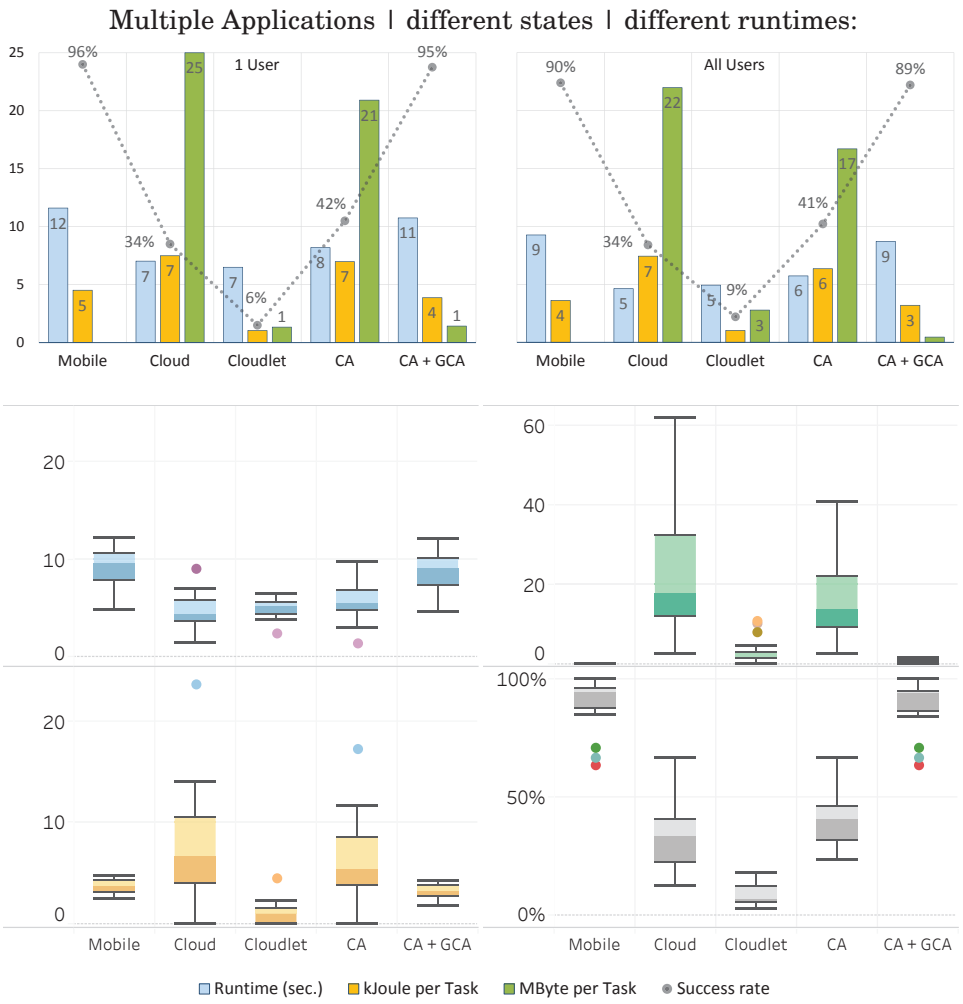

Abbildung 7.19.: Taskausführung mehrerer Anwendungen, unterstützt durch GCA und Verteilung der Messergebnisse

Durch die entwickelte Funktionalität kann ein mobiles Gerät ebenso ganz neue Aufgaben wahrnehmen, die ohne eine Einbeziehung der Infrastruktur nicht möglich wären. Gleichzeitig erlaubt der gewählte Ansatz, die ressourcenintensive aktive Messung der Bandbreite zu vermeiden und hierdurch die Effizienz der kooperativen Ausführung zu erhöhen. Allerdings ist auch zu erkennen, dass zwar viele, aber nicht alle Anwendungsfälle und nicht alle Arten von Tasks von einer kooperativen Ausführung profitieren. Entsprechend ist es beispielsweise notwendig, die Funktionalität einer mobilen Anwendung aufzuteilen, wie es im ersten Evaluationsszenario gezeigt wurde (vergleich Abschnitt 7.2.1).

7.3. Zusammenfassung

Im vorhergehenden Abschnitt wurde zunächst eine qualitative Evaluation des entwickelten Konzepts und der entsprechenden prototypischen Implementie-

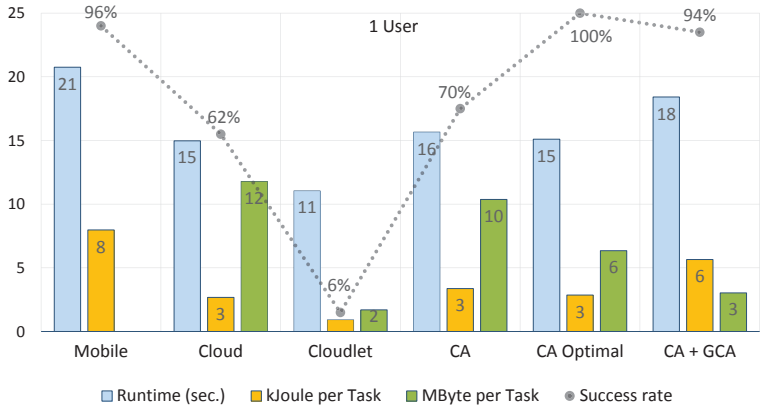

Abbildung 7.20.: Taskausführung zur Beurteilung der Qualität der Kontextadaption

rung vorgenommen. Hierbei kann festgehalten werden, dass das vorgeschlagene Konzept einen hohen Erfüllungsgrad der im Rahmen der Anforderungsanalyse erhobenen Kriterien aufweist. Die prototypische Implementierung dieses Konzepts wurde anschließend im Rahmen einer aus zwei Projekten bestehenden quantitativen Evaluation auf ihre praktische Einsetzbarkeit hin untersucht.

Das erste Evaluationsprojekt hat exemplarisch die Nutzbarkeit des entwickelten Konzepts im Zusammenhang mit der Entwicklung einer Beispielanwendung zur Objekt- und Gesichtserkennung auf mobilen Geräten demonstriert. Die dabei genutzte Beispielanwendung hat die Infrastruktur eines mobilen Gerätes verwendet, um die Erkennungsleistung gegenüber der lokalen Verarbeitung auf dem mobilen Gerät zu erhöhen und weitergehende Funktionalität in Bezug auf die Untersuchung der erkannten Objekte bereitzustellen. Hierdurch konnte die Effizienz des vorgeschlagenen Konzepts im Hinblick auf die Geschwindigkeit und Latenz gezeigt werden.

Das zweite Evaluationsprojekt fokussierte die Untersuchung der Fähigkeit des vorgeschlagenen Konzepts, mit wechselndem Kontext umzugehen und im Zusammenhang mit der Nutzung der generischen Kontextadaption einen zukünftigen Kontext zu antizipieren, um unterschiedliche Ziele der Adaption, beispielsweise die Einsparung von Energie oder die Beschleunigung der Ausführung von Geschäftslogik, zu realisieren. Hierzu wurde ein Simulationsmodell entwickelt, welches es erlaubt, den Kontext eines mobilen Gerätes im Zusammenhang mit der Ausführung einer entsprechenden Beispielanwendung zur Bildverarbeitung zu simulieren. Dabei wurden realistische Nutzungsmuster unterschiedlicher Anwender über einen Zeitraum von jeweils 18 Monaten simuliert, um anschließend die Umsetzbarkeit der unterschiedlichen Ziele der Adaption, beispielsweise die Einsparung von Energie oder die Beschleunigung der Ausführung von Geschäftslogik, beurteilen zu können. Gegenüber einer Evaluation in einer Laborsituation konnten so Messwerte mit einer deutlich höheren Generalisierungsfähigkeit ermittelt werden.

8. Zusammenfassung und Fazit

In dieser Arbeit wurden Konzepte für eine kontextadaptive Anwendungsarchitektur entwickelt, die es mobilen Anwendungen erlaubt, abhängig von ihrem Nutzungskontext mit ihrer umgebenden Infrastruktur zu kooperieren.

In der folgenden Schlussbetrachtung sollen die hierbei erzielten praktischen Ergebnisse zunächst zusammengefasst und anschließend in Bezug auf die Motivation und Zielsetzung dieser Arbeit kritisch beleuchtet werden. Abschließend werden die im Verlauf der Arbeit gewonnenen Erkenntnisse und der dadurch erzielte Beitrag zur Wissenschaft herausgestellt. Das Kapitel schließt mit einem Ausblick auf weitergehende Fragestellungen und mögliche Erweiterungen des Konzepts einer kontextadaptiven Anwendungsarchitektur ab. Hierzu werden zunächst mögliche funktionale Erweiterungen des vorgeschlagenen Lösungsansatzes identifiziert. Abschließend wird ein Ausblick auf generelle Anknüpfungspunkte gegeben.

8.1. Diskussion der Ergebnisse

Mobile Geräte haben sich für eine Vielzahl von Menschen zu selbstverständlichen Alltagsgegenständen entwickelt. Sie unterstützen sie intelligent bei einer Vielzahl täglicher Aufgaben. Belege für diese Relevanz finden sich unter anderem in der Nutzungsdauer dieser Geräte und in den weltweit zunehmenden Absatzzahlen (vergleiche Abschnitt 1). Mit Blick auf aktuelle mobile Anwendungen und Geräte zeichnen sich hierbei allerdings zwei gegensätzliche Trends ab:

Einerseits lassen sich wachsende Anforderungen der Nutzer in Hinblick auf die Leistungsfähigkeit und Verfügbarkeit dieser mobilen Geräte feststellen. Andererseits ist eine stetige Miniaturisierung mobiler Geräte zu beobachten, der nur in Teilen eine entsprechende Leistungssteigerung folgt. Dies hat dazu geführt, dass ein Großteil mobiler Geräte unterschiedlichen Restriktionen wie begrenzten Rechen- und Speicherressourcen oder einem begrenzten Energievorrat unterliegen. Diese Grenzen der ortsbezogenen Mobilität der Nutzer haben die Entwicklung von mobilen Anwendungen begünstigt, die eine Kooperation mit weiteren Ressourcen in ihrer direkten Umgebung oder ihrer Infrastruktur im Rahmen eines mobilen Cloud Computings nutzen, um den aufgezeigten Restriktionen mobiler Geräte zu begegnen.

Der Versuch, diese Kooperation mithilfe existierender Mechanismen zur verteilten Ausführung von Anwendungen zu realisieren, bringt jedoch eine Reihe zusätzlicher Herausforderungen mit sich. Diese resultieren unter anderem aus der wechselnden Konnektivität mobiler Geräte, die darauf zurückzuführen ist, dass sich der Nutzungskontext mobiler Geräte deutlich häufiger ändert als

dies bei stationären Geräten der Fall ist. Dies betrifft die Erkennung nutzbarer Ressourcen innerhalb eines laufend wechselnden Kontextes sowie die Entscheidung über die sinnvolle Aufteilung einer mobilen Anwendung zwischen mobilem Gerät und der Infrastruktur durch Nutzung entsprechender Adaptionsstrategien.

Motiviert durch diese Feststellung hat diese Arbeit vor dem Hintergrund aktueller und zukünftiger Anwendungsszenarien in Bezug auf die Nutzung mobiler Geräte untersucht, wie die verteilte Ausführung mobiler Anwendungen im mobilen Cloud Computing ermöglicht und unter Berücksichtigung des Nutzungskontextes effizient gestaltet werden kann. In diesem Zusammenhang wurden, aufbauend auf einer initialen Anforderungsanalyse, als Ergebnis und wesentlicher Beitrag dieser Arbeit Konzepte für eine kontextadaptive Anwendungsarchitektur entwickelt, die es mobilen Anwendungen ermöglicht, verteilt und kontextadaptiv in der sie umgebenden Infrastruktur ausgeführt zu werden.

Diese Anwendungsarchitektur erlaubt es damit, unterschiedliche Adaptionsziele umzusetzen und beispielsweise die Fähigkeiten mobiler Geräte zu verbessern oder sie um gänzlich neue Fähigkeiten zu erweitern. Konkret umfassen diese Ziele die Entwicklung von Adaptionsstrategien, die der Einsparung von Ressourcen oder der Erhöhung der Dienstqualität und Dienstverfügbarkeit dienen. Im Gegensatz zu vergleichbaren Lösungsansätzen für das mobile Cloud Computing, in denen Entwickler mobiler Anwendung aufgefordert sind, entsprechende Adaptionsstrategien mithilfe regelbasierter Systeme zu implementieren, wurde im Rahmen der vorliegenden Arbeit das Konzept der generischen Kontextadaption entwickelt, welches die entsprechenden Adaptionsstrategien weitgehend eigenständig lernt und hierzu den zukünftigen Kontext eines mobilen Gerätes antizipiert, um die Effektivität der Anpassung und die Effizienz der jeweiligen Adaptionsstrategie weiter zu erhöhen. Im Rahmen der Umsetzung der vorgeschlagenen Architektur in Form einer entsprechenden Systemunterstützung konnten hierbei alle wesentlichen konzeptionellen Elemente im Rahmen der prototypischen Implementierung umgesetzt werden.

Die Evaluation dieses Lösungsansatzes wurde im Rahmen zweier Beispielanwendungen durchgeführt: Das erste Evaluationsprojekt hat die Nutzbarkeit des vorgeschlagenen Konzepts im Zusammenhang mit der Entwicklung einer Beispielanwendung zur Gesichtserkennung auf mobilen Geräten untersucht. Diese Beispielanwendung verwendet die Infrastruktur eines mobilen Gerätes, um die Erkennungsleistung gegenüber der lokalen Verarbeitung zu erhöhen und weitergehende Funktionalität im Hinblick auf die Untersuchung der erkannten Elemente bereitzustellen. In diesem Zusammenhang wurden die im Konzept entwickelten Adaptionsformen der Anpassung des Ausführungsortes und der Anpassung der Geschäftslogik auf ihre praktische Anwendbarkeit hin untersucht. Als Ergebnis der Evaluation konnte dabei einerseits die Nutzbarkeit der in der vorliegenden Arbeit entwickelten kontextadaptiven Anwendungsarchitektur durch typische Entwickler mobiler Anwendungen belegt werden. Andererseits konnte die Effizienz des Ansatzes in

Bezug auf Geschwindigkeit und Latenz der entwickelten Beispielanwendung exemplarisch demonstriert werden.

Das zweite Evaluationsprojekt fokussierte die Untersuchung der Fähigkeit des vorgeschlagenen Konzepts, mit einem wechselnden Kontext umzugehen und im Zusammenhang mit der Nutzung des Konzepts der generischen Kontextadaption einen zukünftigen Kontext zu antizipieren. Es zielt darauf ab, unterschiedliche Ziele der Adaption, zum Beispiel die Einsparung von Energie oder die Beschleunigung der Ausführung von Geschäftslogik, zu realisieren. Hierzu wurde eine Anwendung zur Bildverarbeitung entwickelt, die eine Anpassung des Ausführungsortes, der Ausführungszeit und der Implementierung nutzt, um eine kooperative Ausführung innerhalb der Infrastruktur eines mobilen Gerätes zu realisieren. Die Laufzeiteigenschaften dieser Anwendung wurden anschließend anhand von Ausführungszeiten und Energieverbrauch einzelner Task-Ausführungen ermittelt, um sie im Rahmen des wechselnden Kontextes mobiler Geräte zu simulieren. In diesem Zusammenhang wurde eine Simulationsumgebung vorgeschlagen, die es erlaubt, sowohl die relevanten Kontextattribute eines mobilen Gerätes als auch die das Gerät umgebenden Infrastruktur zu simulieren, um darauf aufbauend die Effizienz der Ausführung unterschiedlicher Tasks der entwickelten Beispielanwendung demonstrieren zu können.

Aufbauend auf einem entsprechenden Datensatz, der die Kontextdaten unterschiedlicher Nutzer mobiler Geräte widerspiegelt, wurden anschließend die Kontextzustände und die im Datensatz enthaltenen Nutzungsmuster mobiler Anwendungen verwendet, um mithilfe des Simulationsmodells die Nutzung der exemplarischen mobilen Anwendung innerhalb eines wechselnden Kontextes zu simulieren. Hierbei wurde ein Nutzungszeitraum von 18 Monaten für 20 unterschiedliche Nutzer zugrunde gelegt, um repräsentative Aussagen hinsichtlich der allgemeinen Anwendbarkeit und den zu erwartenden Nutzen im praktischen Einsatz des vorgeschlagenen Konzepts zu belegen. Hierbei konnte im Rahmen der Simulation gezeigt werden, dass die im Konzept verfolgten Ziele, insbesondere die Einsparung von Energie und die Beschleunigung der Ausführung, nicht nur konzeptionell, sondern auch innerhalb realistischer Nutzungsszenarien erreicht werden können. Gleichzeitig wurde demonstriert, dass der in der vorliegenden Arbeit entwickelte Lösungsansatz in der Lage ist, den häufig wechselnden Kontext mobiler Geräte zu berücksichtigen und entsprechend die verteilte Ausführung von Anwendungen im mobilen Cloud Computing proaktiv anpassen kann.

Im Rahmen der Evaluation wurden jedoch gleichzeitig die Grenzen des hier vorgeschlagenen Ansatzes sichtbar: So lassen sich nur bestimmte Klassen mobiler Anwendungen sinnvoll im Rahmen eines mobilen Cloud Computings nutzen. Bei diesen Anwendungen gilt es zusätzlich, die Teile der Geschäftslogik im Rahmen einer Partitionierung zu identifizieren, die von einer Verschiebung in die Infrastruktur, einer Verzögerung der Ausführung oder einer Anpassung der Implementierung profitieren. Hierbei kann festgehalten werden, dass diese Einheiten der Geschäftslogik weder so klein zugeschnitten sein sollten, dass

sie im Rahmen einer taskbasierten Adaption einer häufigen Interaktion mit weiteren an der Ausführung beteiligten Teilnehmern einer mobilen Cloud bedürfen. Noch sollten diese Einheiten nicht so groß sein, sodass eine Verlagerung vor dem Hintergrund des schnell und häufig wechselnden Kontextes mobiler Geräte beeinträchtigt und somit in vielen Fällen eine kontextabhängige Anpassung der Adaptionsstrategie verhindert wird.

Obwohl die Nutzbarkeit des hierzu entwickelten Prozesses der generischen Kontextadaption im Rahmen der Evaluation belegt wurde, gilt es, die ständige Erfassung der hierzu benötigten Kontextattribute möglichst effizient und insbesondere energiesparend zu realisieren, um insgesamt trotzdem eine Einsparung von Ressourcen zu realisieren. Entsprechend wurde beispielsweise die Ermittlung der verfügbaren Bandbreite aus Gründen der Effizienz nicht in Form einer aktiven Messung sondern als Abschätzung in Form der Prognose eines Kontextattributes umgesetzt. Durch den Einsatz dieser datengetriebenen Verfahren zur Ermittlung entsprechender Adaptionsstrategien gilt es zudem, zunächst eine entsprechende Historie erfolgreicher als auch nicht erfolgreicher Adaptionen zu generieren und für das Training der entsprechenden Vorhersagemodelle vorzuhalten. Dieser Bedarf zur Exploration sorgt dafür, dass, sofern nicht andere Nutzer einer mobilen Anwendung diese Daten bereitstellen, diese zunächst durch die Nutzung der Anwendung zu ermitteln sind.

Insgesamt kann jedoch festgehalten werden, dass es gelungen ist, die angestrebte Erweiterung bestehender Konzepte für die verteilte Ausführung im mobilen Cloud Computing zu realisieren und hierbei den Nutzungskontext der Geräte in die für eine effiziente Ressourcennutzung optimierten Adaptionsprozesse zu integrieren.

8.2. Wissenschaftlicher Beitrag

Das konkrete Ziel der vorliegenden Dissertation war es, Konzepte bereitzustellen, die die zusätzliche Komplexität des mobilen Cloud Computings beherrschbar machen und eine angemessene Unterstützung für die Entwickler bereitstellen, um die verteilte und kontextadaptive Ausführung von Anwendungen zwischen mobilen und stationären Geräten zu ermöglichen. Der zentrale Forschungsaspekt der Arbeit lag entsprechend auf der gemeinsamen Betrachtung zweier miteinander verbundenen Problemstellungen: der effizienten verteilten Ausführung mobiler Anwendungen durch mobiles Cloud Computing und der Berücksichtigung des Nutzungskontextes dieser Anwendungen sowie der beteiligten Geräte, um eine effiziente Kooperation zu ermöglichen.

In diesem Zusammenhang wurden die spezifischen Problemstellungen im Bereich des mobilen Cloud Computings herausgestellt und entsprechende Lösungsansätze erarbeitet, um Forscher und Softwareentwickler gleichermaßen bei der Konzeption eigener Lösungen zu unterstützen.

Vor diesem Hintergrund hat diese Arbeit mit ihrer ersten Forschungsfrage zunächst die generelle Eignung existierender Konzepte dahingehend untersucht, ob sie generell für das mobile Cloud Computing genutzt und im Hinblick

auf ihre Fähigkeit zur Berücksichtigung des Nutzungskontextes erweitert werden können. Als konkretes Forschungsziel wurden hierzu zunächst bestehende Konzepte zur verteilten Ausführung von Anwendungen in Bezug auf ihren Einsatz im mobilen Cloud Computing untersucht. In diesem Zusammenhang wurden Anwendungsdomänen ausgewählt, die von einer kooperativen Ausführung durch mobiles Cloud Computing profitieren, um die bei der Nutzung auftretenden Problemstellungen der Verteilung und Kontextadaption zu identifizieren.

Diese Problemstellungen wurden im Rahmen eines Anforderungskataloges für eine kontextadaptive Anwendungsarchitektur festgehalten. Anschließend wurde eine repräsentative Zahl existierender Ansätze aus verschiedenen Forschungsrichtungen im Umfeld des mobilen Cloud Computings hinsichtlich ihrer Nutzbarkeit und Eignung für die hier vorliegende Problemstellung untersucht. Hierbei konnte festgestellt werden, dass aktuell keine Lösung existiert, die alle wesentlichen im Rahmen der Arbeit identifizierten Anforderungen ausreichend erfüllt. Als Konsequenz wurden anschließend erfolgsversprechende Lösungsansätze tiefergehend gegen die identifizierten Anforderungen analysiert, um im Anschluss einen abstrakten Entwurf eines Lösungsansatzes in Form einer Basisarchitektur für mobiles Cloud Computing bereitzustellen.

Der erste wissenschaftliche Beitrag liegt somit im hierdurch entstandenen Erkenntnisgewinn für spezifische Problemstellungen und Lösungsansätze im mobilen Cloud Computing und zielt somit gleichermaßen darauf ab, die Eignung bestehender Konzepte zu belegen sowie die Entwicklung neuer Konzepte zu unterstützen.

Um die verteilte Ausführung einer mobilen Anwendung nicht nur effizient, sondern im Hinblick auf die wechselnde Konnektivität und den wechselnden Kontext mobiler Geräte auch möglichst spontan zu ermöglichen, wurde als zweites Forschungsziel untersucht, wie Kontextinformationen dazu verwendet werden können, die verteilte Ausführung mobiler Anwendungen im mobilen Cloud Computing effizienter zu gestalten. Konkret wurde hierzu untersucht, wie die Berücksichtigung der umliegenden Infrastruktur mobiler Geräte und deren kontextabhängiger Verfügbarkeit die Benutzbarkeit mobiler Anwendungen positiv beeinflussen kann. Hierzu wurden zunächst Adaptionsziele identifiziert und anschließend Adaptionsstrategien entwickelt, die den aktuellen und zukünftigen Kontext eines mobilen Gerätes berücksichtigen. Anschließend wurde eine konkrete Unterstützung für Softwareentwickler vorgeschlagen, die sowohl die hierfür konkret benötigten Kontextattribute als auch weitere Kontextattribute vorhersagen kann, um eine proaktive Anpassung zu ermöglichen. Hierzu wurde ein generischer Prozess zur Kontextadaption entwickelt, der speziell auf die beschränkten Ressourcen mobiler Geräte ausgerichtet wurde und der durch seine Fähigkeit mit verschiedenen Datenquellen, wechselnder Datenqualität und einer begrenzten Historie in der Lage ist, den Kontext eines mobilen Gerätes weitgehend vorherzusagen.

Hierauf aufbauend wurde das Konzept einer kontextadaptiven Anwendungsarchitektur entwickelt, die in der Lage ist, den Nutzungskontext in ihre Verteilungsstrategie und ihr Verhalten mit einzubeziehen. Als wesentlicher

Beitrag zur Wissenschaft wurden in diesem Zusammenhang Adaptionsstrategien entwickelt, die den zukünftigen Kontext eines mobilen Geräts vorhersagen können und die Verteilungsstrategie sowie das Verhalten der mobilen Anwendung proaktiv anpassen. Die entwickelten Lösungen erlauben es mobilen Anwendungen, einen Adaptionsbedarf eigenständig zu erkennen und neue Adaptionsstrategien auf Basis existierenden Wissens und gelernter Zusammenhänge selbstständig zu erschließen. Durch diesen Ansatz sind die Entwickler mobiler Geräte nicht länger in der Verantwortung, sich selbst mit den Details der Verteilung oder Kontextadaption auseinanderzusetzen und entsprechende Adaptionsregeln bereitstellen zu müssen.

Um die Umsetzbarkeit der vorgeschlagenen Konzepte zu demonstrieren, erfolgte anschließend eine exemplarische Implementierung einer kontextadaptiven Anwendungsarchitektur in Form des CloudAware-Frameworks, dessen praktische Nutzbarkeit im Rahmen einer Simulation und einer Beispielanwendung belegt wurde. An dieser Stelle konnte die erfolgreiche Kombination von Konzepten zur verteilten Ausführung und Kontextadaption im Hinblick auf die effiziente Nutzung der Infrastruktur mobiler Geräte gezeigt werden.

Der zweite wissenschaftliche Beitrag liegt entsprechend in den entwickelten Lösungsansätzen für die zuvor beschriebenen Problemfelder: den Adaptionsstrategien zur Kooperation mit der Infrastruktur und dem Konzept der generischen Kontextadaption.

Die allgemeine Anwendbarkeit der vorgestellten Lösungsansätze ist dabei nicht auf das Problemfeld des mobilen Cloud Computings beschränkt: So kann beispielsweise das Konzept der generischen Kontextadaption ebenso dazu genutzt werden, im Allgemeinen die Proaktivität kontextbewusster Systeme im Bereich des Mobile Computings zu verbessern.

Zusammenfassend leistet diese Arbeit somit einen Beitrag in Bezug auf die Vision des Ubiquitous Computings, die darauf abzielt, intelligente Gegenstände in unseren Alltag einzubetten, um uns möglichst unauffällig in unserem Alltag zu unterstützen.

8.3. Ausblick

Das in der vorliegenden Arbeit entwickelte Konzept einer kontextadaptiven Anwendungsarchitektur für das mobile Cloud Computing bietet eine Reihe von Erweiterungsmöglichkeiten für zukünftige Forschungs- und Entwicklungsaktivitäten.

8.3.1. Erweiterungen des Ansatzes

Allem voran gilt es, die Benutzbarkeit des entwickelten Ansatzes sowohl für die Entwickler mobiler Anwendungen als auch für die Betreiber einer entsprechenden Surrogate-Infrastruktur zu optimieren, um die Einstiegshürden für die Nutzung möglichst gering zu halten. In diesem Zusammenhang sollte beispielsweise bereits innerhalb des Entwicklungsprozesses das zukünftige Ver-

halten der mobilen Anwendung innerhalb einer verteilten Infrastruktur untersucht werden können, um dem Entwickler Hinweise in Bezug auf mögliche Optimierungen seines Entwurfs zu geben. Dies könnte beispielsweise in Form der Erweiterung einer entsprechenden Entwicklungsumgebung realisiert werden, die hierzu die im Rahmen der Evaluation entwickelte Simulationsumgebung nutzt.

Im Hinblick auf die Unterstützung der Anbieter und Betreiber erforderlicher Surrogate-Infrastrukturen sollte das Deployment Software-Infrastrukturen und Anwendungen möglichst im Einklang mit den durch die Ökosysteme der App-Stores vorgegebenen Deployment-Prozessen für mobile Anwendungen erfolgen. Auf der Seite der Infrastruktur können hierzu eine leichtgewichtige Betriebssystemvirtualisierung mittels entsprechender Container-Lösungen[1] genutzt werden, für welche eine stets größer werdende Anzahl mobiler und stationärer Geräte entsprechende Ausführungsumgebungen bereitstellen. Alternativ könnte die Bereitstellung einer entsprechenden Surrogate-Infrastruktur im Sinne eines sogenannten Mobile Backend as a Service (MBaaS) [Flo15, Kin12] erfolgen, um es einer Vielzahl von Entwicklern zu ermöglichen, sich mit der Entwicklung entsprechender Anwendungen auseinanderzusetzen.

Hinsichtlich der Verfügbarkeit entsprechender Infrastrukturen gilt es ebenso, stets den Kompromiss zwischen einer möglichst spontanen Interaktion mit der Infrastruktur und gleichzeitig einem möglichst geringen Overhead durch die hierfür benötigte aktive Suche nach verfügbaren Surrogates zu realisieren. Eine entsprechende Umsetzung leichtgewichtiger Verfahren zur Dienstsuche innerhalb der verantwortlichen Komponente der verwendeten Middleware würde die Effizienz der realisierten Implementierung weiter erhöhen und möglicherweise gleichzeitig die Fähigkeit zur spontanen Interaktion optimieren.

Eine weitere generelle Herausforderung des mobilen Cloud Computings besteht in der Aufrechterhaltung der Informationssicherheit. Die verarbeiteten Daten stellen mitunter vertrauliche Informationen dar, die in einer gemeinsam genutzten und teils öffentlich zugänglichen Infrastruktur verarbeitet werden. Obwohl im Rahmen dieser Arbeit Mechanismen zum Schutz der Vertraulichkeit dieser Informationen entwickelt wurden, kann dabei kein umfassender Schutz vor unberechtigtem Zugriff sichergestellt werden, was es beim praktischen Einsatz des entwickelten Konzepts zu berücksichtigen gilt.

8.3.2. Weitere Anwendungen

Neben diesen konkreten Erweiterungen des in dieser Arbeit vorgeschlagenen Lösungsansatzes existieren ebenso eine Reihe genereller Anknüpfungspunkte, die abschließend erläutert werden sollen.

Mit Blick auf die in dieser Arbeit entwickelten Koordinationsstrategien kann festgehalten werden, dass diese im Hinblick auf die Einsparung von Ressour-

[1] Betriebssystem-Container stellen eine leichtgewichtige Form der Virtualisierung dar, die dazu verwendet werden kann, Anwendungen mithilfe von Betriebssystemvirtualisierung in Containern zu isolieren.

cen mobiler Geräte und zur Erhöhung der Dienstqualität mobiler Anwendungen entwickelt wurden. Entsprechend kann ein Anknüpfungspunkt darin bestehen, weitergehende Kooperationsformen zwischen Anbietern und Nutzern von Ressourcen innerhalb des mobilen Cloud Computings zu untersuchen und hierfür entsprechende Konzepte bereitzustellen. In diesem Zusammenhang könnten insbesondere existierende Ansätze zur dezentralen Koordination und Problemlösung Berücksichtigung finden. Beispielsweise ließe sich eine Realisierung mithilfe von *Multiagentensystemen* [Woo09] dazu nutzen, den auftretenden Koordinationsaufwand bei der konkurrierenden Nutzung von Ressourcen effizient abzubilden. Darüber hinaus kann dieser Ansatz weiteren Kooperationsszenarien im mobilen Cloud Computing dienen, die sich durch die Koordination und Erfüllung gemeinsamer Ziele ergeben.

Ein weiterer Anknüpfungspunkt kann sich aus der Anwendung des in der vorliegenden Dissertation entwickelten Prozesses zur generischen Kontextadaption ergeben: So kann dessen Anwendbarkeit auf gänzlich neue Problemfelder hin untersucht werden, in denen es gilt, datengetriebene Entscheidungen mithilfe automatisierter Wissensextraktion zu unterstützen. Ein konkretes Beispiel hierfür lässt sich beispielsweise im Anwendungsgebiet der automatisierten Produktionsanlagen finden. Hier gilt es auf Basis des beobachteten und damit bekannten Verhaltens dieser Anlagen die Auswirkungen der Anpassung von Betriebsparametern abzuschätzen. Diese Vorhersage erlaubt es Empfehlungen abzuleiten, die Zielgrößen wie die Ausschussrate oder den Wirkungsgrad einer solchen Produktionsanlage zu verbessern. In diesem Zusammenhang wird abschließend auf die aktuell im DFG-Projekt *Fypac*[2] untersuchte Unterstützung der automatisierten Evolution von Softwaresystemen verwiesen, die von den Ergebnissen dieser Arbeit profitieren könnte.

[2] http://www.dfg-spp1593.de/index.php?id=40

A. Kommentare zur qualitativen Evaluation

	Anforderung		Begründung
	Verfügbarkeit		
A1	Fehlerbehandlung	++	Bei Verbindundgsabbrüchen wird nach einem Timeout eine neue Ausführung initiiert.
A2	Erweitertes Fehlermanagement	++	Die Neuausführung beeinflusst wann immer sinnvoll möglich (stateless components) den Kontrollfluss nicht, andernfalls liegt die Kontrolle beim Entwickler.
A3	Prognose der Konnektivität	++	Kontextdienst nutzt historische Daten um Prognosen abzuleiten.
A4	Effizienz	+	Der Service-Aufruf wird, wenn möglich, durch das mobile Gerät abgebrochen.
	Portierbarkeit		
P1	Generelle Auslagerungsfähigkeit	+	Vorhanden, der verwendete Dienst muss jedoch bereits in der Infrastruktur laufen.
P2	Grundlegende Adaptionsfähigkeit	++	Kontextmonitor prüft Kontext und signalisiert Änderungen an den Coordinator.
P3	Erweiterte Adaptionsfähigkeit	++	Zukünftige Kontextzustände werden in der Adaption / kooperativen Ausführung berücksichtigt.
P4	Koexistenz	-	Macht keine exklusive Nutzung von Ressourcen, ist jedoch nicht in das Betriebssystem des mobilen Geräts integriert und nutzt somit dessen Scheduler.
P5	Deployment	+	Das Deployment auf einem mobilen Gerät unterscheidet sich nicht von Deployment übriger mobiler Anwendungen. In der Infrrastruktur ist ein Deployment der Dienste der Anwendung nötig.
P6	Offenheit	++	Setzt lediglich die Java Virtual Machine voraus.
	Skalierbarkeit		
S1	Overhead	+	Der Overhead zur Laufzeit ist abhängig vom Intervall der Kontextadaption (Messung, Prognose, Anpassung der kooperativen Ausführung).
S2	Dienstsuche und Integration	+	Neue Ressourcen können auf Kosten des Overheads aktiv gesucht werden.
S3	Parallelisierung	+	Ist durch den Entwickler zu implementieren, aber wird durch das Aktormodell einfach unterstützt.
S4	Effizienz	-	nicht explizit adressiert
	Nutzbarkeit		
U1	Einfache Nutzbarkeit	++	Es werden nur Kenntnisse des typischen Entwicklers mobiler Anwendungen vorausgesetzt (OOP).
U2	Angemessene Abstraktion	++	MCC-spezifische Problemstellungen werden für den Entwickler transparent gelöst.
U3	Unterstützung von Standards	++	Die Lösung integriert sich in den Entwicklungsprozess und Werkzeuge für mobile Anwendungen.
U4	Automatische Konfiguration	+	Adaptionsziele können in der mobilen Anwendung durch den Entwickler definiert werden.
	Wartbarkeit		
M1	Determiniertheit	+	Sofern der Entwickler auch im Rahmen der Fidelity-Adaptation gleiche Dienste bereitstellt.
M2	Debugging	++	Die Lösung integriert sich in den Entwicklungsprozess und Werkzeuge für mobile Anwendungen.
M3	Offene Standards	+	Nutzt nur frei verfügbare Software als Basistechnologien.
	Sicherheit		
SE1	Schutz der Vertraulichkeit	+	Ermöglicht es, bestimmte Teile der Geschäftslogik nur lokal auszuführen.
SE2	Isolation	-	nicht adressiert

++ vollständig erfüllt + teilweise erfüllt - nicht erfüllt 0 nicht adressiert

Tabelle A.1.: Kommentare zur Evaluation des entwickelten Lösungsansatzes

B. Struktur des LDDC-Datensatzes

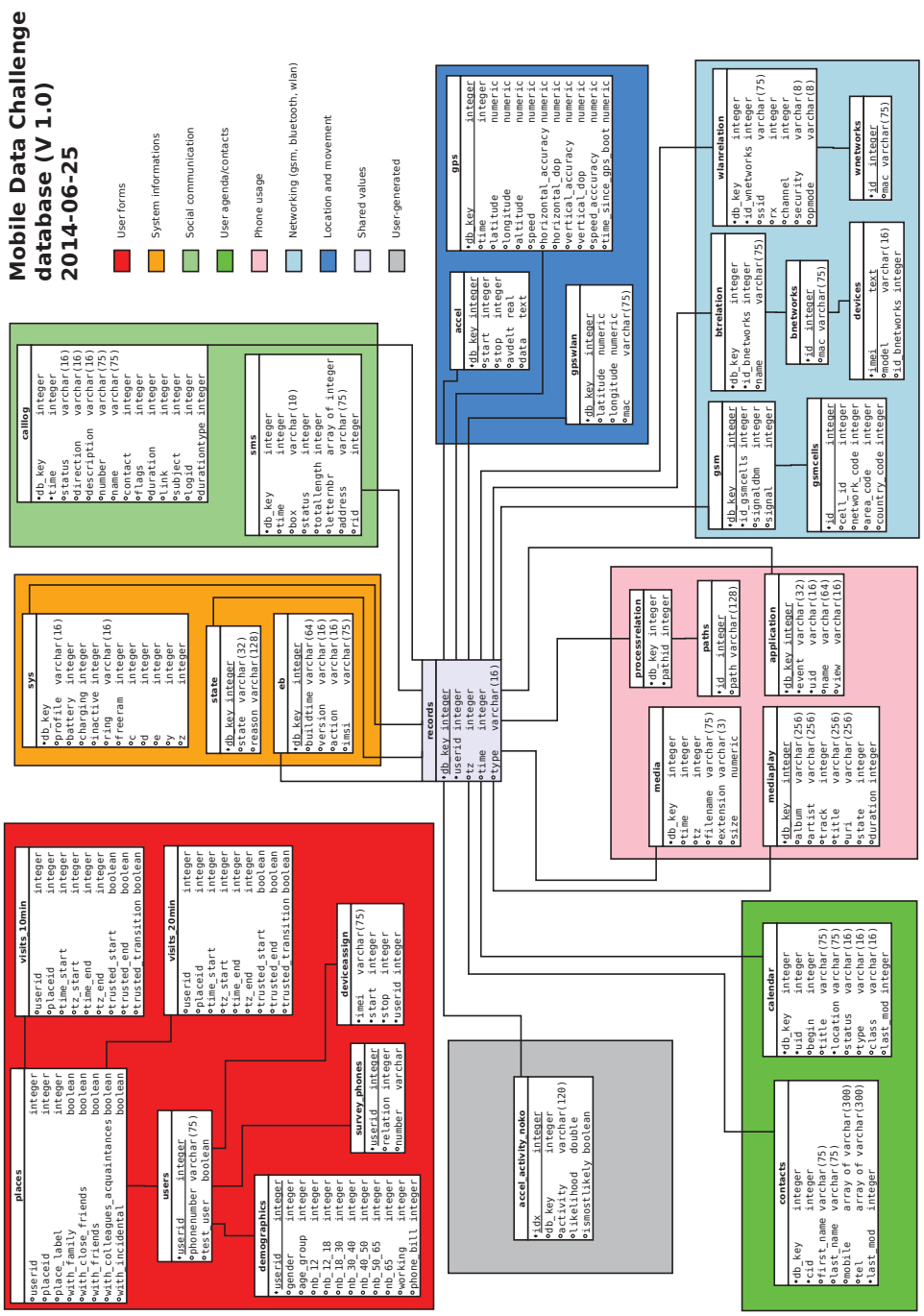

Abbildung B.1.: Struktur des LDDC-Datensatzes, entnommen aus (ldi16)

Eigene Veröffentlichungen

Folgende Veröffentlichungen sind aus den Ergebnissen im Umfeld dieses Dissertationsprojekts hervorgegangen:

G. Orsini, D. Bade und W. Lamersdorf CloudAware: Towards Context-adaptive Mobile Cloud Computing In: Badonnel, R., J. Xiao, S. Ata, F. de Turck und V. Groza (Herausgeber): *IFIP/IEEE International Symposium on Integrated Network Management, IM 2015, Ottawa, ON, Canada, 11-15 May, 2015*, Seiten 1190–1195. IFIP, IEEE Explore Washington/DC, USA, 5 2015.

G. Orsini, D. Bade und W. Lamersdorf Context-Aware Computation Offloading for Mobile Cloud Computing: Requirements Analysis, Survey and Design Guideline In: *The 12th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2015) / Affiliated Workshops, August 17-20, 2015, Belfort, France*, Procedia Computer Science, Seiten 10–17. Elsevier Science, 8 2015.

G. Orsini, D. Bade und W. Lamersdorf Computing at the Mobile Edge: Designing Elastic Android Applications for Computation Offloading In: *8th IFIP Wireless and Mobile Networking Conference, WMNC 2015, Munich, Germany, October 5-7, 2015*, Seiten 112–119. IEEE Explore Washington/DC, USA, 2015.

G. Orsini, D. Bade und W. Lamersdorf Generic Context Adaptation for Mobile Cloud Computing Environments In: *The 13th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2016) / Affiliated Workshops, August 15-18, 2016, Montreal, Quebec, Canada*, Procedia Computer Science, Seiten 17–24. Elsevier Science, 8 2016.

G. Orsini, D. Bade und W. Lamersdorf CloudAware: A Context-adaptive Middleware for Mobile Edge and Cloud Computing Applications In: *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, Seiten 216–221. IEEE Explore Washington/DC, USA, 9 2016.

G. Orsini, D. Bade und W. Lamersdorf (angenommen zur Veröffentl.) Generic Context Adaptation for Mobile Cloud Computing Environments (Extended Version) In: *Journal of Ambient Intelligence and Humanized Computing (JAIHC)*. Springer, 2017.

G. Orsini, D. Bade und W. Lamersdorf (angenommen zur Veröffentl.) CloudAware: Empowering Context-Aware Self-Adaptation for Mobile Applications In: *Transactions on Emerging Telecommunications Technologies (ETT)*. Wiley, 2017.

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Literaturverzeichnis

[3GP13] 3GPP: Lte-advanced. http://www.3gpp.org/technologies/keywords-acronyms/97-lte-advanced, 2013. aufgerufen am 31.12.2016.

[AA16] AHMED, ARIF und EJAZ AHMED: A survey on mobile edge computing. In: *the Proceedings of the 10th IEEE International Conference on Intelligent Systems and Control (ISCO 2016), Coimbatore, India*, 2016.

[AAH+09] ANAGNOSTOPOULOS, THEODOROS, CHRISTOS ANAGNOSTOPOULOS, STATHES HADJIEFTHYMIADES, MILTOS KYRIAKAKOS und ALEXANDROS KALOUSIS: Predicting the location of mobile users: a machine learning approach. In: *Proceedings of the 2009 international conference on Pervasive services*, Seiten 65–72. ACM, 2009.

[AAH11] ANAGNOSTOPOULOS, THEODOROS, CHRISTOS ANAGNOSTOPOULOS und STATHES HADJIEFTHYMIADES: Mobility prediction based on machine learning. In: *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*, Band 2, Seiten 27–30. IEEE, 2011.

[Act16] ACTORON GMBH: Android - jadex active components documentation. https://download.actoron.com/docs/releases/latest/jadex-mkdocs/android/android/, 2016. aufgerufen am 31.12.2016.

[AFG+10] ARMBRUST, MICHAEL, ARMANDO FOX, REAN GRIFFITH, ANTHONY D JOSEPH, RANDY KATZ, ANDY KONWINSKI, GUNHO LEE, DAVID PATTERSON, ARIEL RABKIN, ION STOICA und OTHERS: A view of cloud computing. Communications of the ACM, 53(4):50–58, 2010.

[AGK+15] AHMED, EJAZ, ABDULLAH GANI, MUHAMMAD KHURRAM KHAN, RAJKUMAR BUYYA und SAMEE U KHAN: Seamless application execution in mobile cloud computing: Motivation, ta-

xonomy, and open challenges. Journal of Network and Computer Applications, 52:154–172, 2015.

[AGRS05] ADELSTEIN, FRANK, SANDEEP KS GUPTA, GOLDEN RICHARD und LOREN SCHWIEBERT: *Fundamentals of mobile and pervasive computing*, Band 1. McGraw-Hill New York, 2005.

[AGS+15] AHMED, EJAZ, ABDULLAH GANI, MEHDI SOOKHAK, SITI HAFIZAH AB HAMID und FENG XIA: Application optimization in mobile cloud computing: Motivation, taxonomies, and open challenges. Journal of Network and Computer Applications, 52:52–68, 2015.

[All16] ALLIANCE, OPEN HANDSET: Application fundamentals | android developers. https://developer.android.com/guide/components/fundamentals.html, 2016. aufgerufen am 31.12.2016.

[Ama13] AMAZON.COM, INC.: EC2 Instance Types. http://aws.amazon.com/de/ec2/instance-types/, 2013. aufgerufen am 31.12.2016.

[Ama16] AMAZON.COM, INC.: Cloud-Produkte und -Services – Amazon Web Services (AWS). https://aws.amazon.com/de/products/, 2016. aufgerufen am 31.12.2016.

[App13] APPLE INC.: Apple – iOS – Fragen und Antworten zu Siri. http://www.apple.com/de/ios/siri/siri-faq/, 2013. aufgerufen am 31.12.2016.

[ASA+14] ABOLFAZLI, SAEID, ZOHREH SANAEI, EJAZ AHMED, ABDULLAH GANI und RAJKUMAR BUYYA: Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges. IEEE Communications Surveys & Tutorials, 16(1):337–368, 2014.

[ASG+14] ABOLFAZLI, SAEID, ZOHREH SANAEI, ABDULLAH GANI, FENG XIA und WEI-MING LIN: Rmcc: Restful mobile cloud computing framework for exploiting adjacent service-based mobile cloudlets. In: *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, Seiten 793–798. IEEE, 2014.

[ATBB08] AYED, DHOUHA, CHANTAL TACONET, GUY BERNARD und

YOLANDE BERBERS: Cadecomp: Context-aware deployment of component-based applications. Journal of Network and Computer Applications, 31(3):224–257, 2008.

[Bak14] BAKKER, ALEXANDER: Comparing energy profilers for android. In: *21st Twente Student Conference on IT*, Band 21, 2014.

[BBC97] BROWN, PETER J, JOHN D BOVEY und XIAN CHEN: Context-aware applications: from the laboratory to the marketplace. IEEE personal communications, 4(5):58–64, 1997.

[BBH+10] BETTINI, CLAUDIO, OLIVER BRDICZKA, KAREN HENRICKSEN, JADWIGA INDULSKA, DANIELA NICKLAS, ANAND RANGANATHAN und DANIELE RIBONI: A survey of context modelling and reasoning techniques. Pervasive and Mobile Computing, 6(2):161–180, 2010.

[BC12] BARRY, PETER und PATRICK CROWLEY: *Modern embedded computing: designing connected, pervasive, media-rich systems*. Elsevier, 2012.

[BC16] BELLAVISTA, PAOLO und ANTONIO CORRADI: *The handbook of mobile middleware*. CRC press, 2016.

[BCGS04] BASAGNI, STEFANO, MARCO CONTI, SILVIA GIORDANO und IVAN STOJMENOVIC: *Mobile ad hoc networking*. John Wiley & Sons, 2004.

[BDR07] BALDAUF, MATTHIAS, SCHAHRAM DUSTDAR und FLORIAN ROSENBERG: A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing, 2(4):263–277, 2007.

[BDR14a] BERG, FLORIAN, FRANK DÜRR und KURT ROTHERMEL: Increasing the efficiency and responsiveness of mobile applications with preemptable code offloading. In: *Mobile Services (MS), 2014 IEEE International Conference on*, Seiten 76–83. IEEE, 2014.

[BDR14b] BERG, FLORIAN, FRANK DÜRR und KURT ROTHERMEL: Optimal predictive code offloading. In: *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, Seiten 1–10. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014.

[Bes16] BESNER, FABIAN: Analyse des energieverbrauchs mobiler anwendungen. Bachelorarbeit, Universität Hamburg, Fachbereich Informatik, Vogt-Kölln-Str. 30, 22527 Hamburg, Germany, 7 2016.

[BGHM10] BEACH, AARON, MIKE GARTRELL, RICHARD HAN und SHIVAKANT MISHRA: Cawbweb: Towards a standardized programming framework to enable a context-aware web. Technischer Bericht, Technical Report CU-CS-1063-10, 2010.

[BGSH07] BALAN, RAJESH KRISHNA, DARREN GERGLE, MAHADEV SATYANARAYANAN und JAMES HERBSLEB: Simplifying cyber foraging for mobile devices. In: *Proceedings of the 5th international conference on Mobile systems, applications and services*, Seiten 272–285. ACM, 2007.

[BH04] BAI, FAN und AHMED HELMY: A survey of mobility models. Wireless Adhoc Networks. University of Southern California, USA, 206:147, 2004.

[BKS03] BUCHHOLZ, T, A KÜPPER und M SCHIFFERS: Quality of context information: What it is and why wee need it. In: *Proceedings of the 10th International Workshop of the HP OpenView University Association (HPOVUA'01)*, Band 2003, 2003.

[Blu16] BLUETOOTH SIG, INC.: Bluetooth technology website. `https://www.bluetooth.com/`, 2016. aufgerufen am 31.12.2016.

[BMV10] BALASUBRAMANIAN, ARUNA, RATUL MAHAJAN und ARUN VENKATARAMANI: Augmenting mobile 3g using wifi. In: *Proceedings of the 8th international conference on Mobile systems, applications, and services*, Seiten 209–222. ACM, 2010.

[BMZA12] BONOMI, FLAVIO, RODOLFO MILITO, JIANG ZHU und SATEESH ADDEPALLI: Fog computing and its role in the internet of things. In: *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, Seiten 13–16. ACM, 2012.

[BP11] BRAUBACH, LARS und ALEXANDER POKAHR: Addressing challenges of distributed systems using active components. In: *Intelligent Distributed Computing V*, Seiten 141–151. Springer, 2011.

[BR00] BETTSTETTER, CHRISTIAN und CHRISTOPH RENNER: A com-

parison of service discovery protocols and implementation of the service location protocol. In: *Proceedings of the 6th EUNICE Open European Summer School: Innovative Internet Applications*, 2000.

[BRS06] BAUER, HANS H, TINA REICHARDT und ANJA SCHÜLE: Was will der mobile nutzer?: Forschungsergebnisse zu den anforderungen von nutzern an kontextsensitive dienste. university of mannheim (2006). In: *Aktuelle Trends in der Softwareforschung : Tagungsband zum doIT Software-Forschungstag 2006*, Seiten 179–191. dpunkt, 2006.

[BSPO03] BALAN, RAJESH KRISHNA, MAHADEV SATYANARAYANAN, SO YOUNG PARK und TADASHI OKOSHI: Tactics-based remote execution for mobile computing. In: *Proceedings of the 1st international conference on Mobile systems, applications and services*, Seiten 273–286. ACM, 2003.

[BVS13] BUYYA, RAJKUMAR, CHRISTIAN VECCHIOLA und S THAMARAI SELVI: *Mastering cloud computing: foundations and applications programming*. Newnes, 2013.

[CAMCM05] CHETAN, SHIVA, JALAL AL-MUHTADI, ROY CAMPBELL und M DENNIS MICKUNAS: Mobile gaia: a middleware for ad-hoc pervasive computing. In: *Second IEEE Consumer Communications and Networking Conference, 2005. CCNC. 2005*, Seiten 223–228. IEEE, 2005.

[Can16a] CANONICAL LTD.: Ubuntu for android | supported devices - ubuntu wiki. `https://wiki.ubuntu.com/Touch/Devices`, 2016. aufgerufen am 31.12.2016.

[Can16b] CANONICAL LTD.: Ubuntu for phones | ubuntu. `http://www.ubuntu.com/phone`, 2016. aufgerufen am 31.12.2016.

[CBC+10] CUERVO, EDUARDO, ARUNA BALASUBRAMANIAN, DAE-KI CHO, ALEC WOLMAN, STEFAN SAROIU, RANVEER CHANDRA und PARAMVIR BAHL: Maui: making smartphones last longer with code offload. In: *Proceedings of the 8th international conference on Mobile systems, applications, and services*, Seiten 49–62. ACM, 2010.

[CCT+11] CHEN, KUAN-TA, YU-CHUN CHANG, PO-HAN TSENG, CHUN-YING HUANG und CHIN-LAUNG LEI: Measuring the latency of cloud gaming systems. In: *Proceedings of the 19th ACM inter-*

national conference on Multimedia, Seiten 1269–1272. ACM, 2011.

[CDC10] CHEN, CHAO, BARNAN DAS und DIANE J COOK: A data mining framework for activity recognition in smart environments. In: *Intelligent Environments (IE), 2010 Sixth International Conference on*, Seiten 80–83. IEEE, 2010.

[CDK12] COULOURIS, GEORGE F, JEAN DOLLIMORE und TIM KINDBERG: *Distributed systems: concepts and design*. pearson education, 2012.

[Cen16] CENTER FOR UBIQUITOUS COMPUTING AT THE UNIVERSITY OF OULU: Aware android mobile context instrumentation framework). `https://www.awareframework.com`, 2016. aufgerufen am 31.12.2016.

[CFH+05] CLARK, CHRISTOPHER, KEIR FRASER, STEVEN HAND, JACOB GORM HANSEN, ERIC JUL, CHRISTIAN LIMPACH, IAN PRATT und ANDREW WARFIELD: Live migration of virtual machines. In: *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation-Volume 2*, Seiten 273–286. USENIX Association, 2005.

[CIM+11] CHUN, BYUNG-GON, SUNGHWAN IHM, PETROS MANIATIS, MAYUR NAIK und ASHWIN PATTI: Clonecloud: elastic execution between mobile device and cloud. In: *Proceedings of the sixth conference on Computer systems*, Seiten 301–314. ACM, 2011.

[Cin12] CINAR, ONUR: *Android apps with Eclipse*. Apress and Distributed to the book trade worldwide by Springer, 2012.

[Cis15] CISCO SYSTEMS INC.: Cisco iox. `https://developer.cisco.com/site/iox/`, 2015. aufgerufen am 31.12.2016.

[Cis16] CISCO SYSTEMS, INC.: Cisco visual networking index: Global mobile data traffic forecast update, 2015–2020 white paper. `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html`, 2016. aufgerufen am 31.12.2016.

[CLK+11] CIDON, ASAF, TOMER M LONDON, SACHIN KATTI, CHRISTOS

KOZYRAKIS und MENDEL ROSENBLUM: Mars: adaptive remote execution for multi-threaded mobile devices. In: *Proceedings of the 3rd ACM SOSP Workshop on Networking, Systems, and Applications on Mobile Handhelds*, Seite 1. ACM, 2011.

[CM10] CHUN, BYUNG-GON und PETROS MANIATIS: Dynamically partitioning applications between weak devices and clouds. In: *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, Seite 7. ACM, 2010.

[CoD03] CODAMOS PROJECT: CoDAMoS: Context-driven adaptation of mobile services. `http://www.cs.kuleuven.be/~distrinet/projects/CoDAMoS/`, 2003. aufgerufen am 31.12.2016.

[COH12] CHEN, ERIC, SATOSHI OGATA und KEITARO HORIKAWA: Offloading android applications to the cloud without customizing android. In: *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*, Seiten 788–793. IEEE, 2012.

[Cra16] CRAWDAD PROJECT: Crawdad a community resource for archiving wireless data at dartmouth. `http://crawdad.org/all-byname.html`, 2016. aufgerufen am 31.12.2016.

[CTSC11] CHON, YOHAN, ELMUROD TALIPOV, HYOJEONG SHIN und HOJUNG CHA: Mobility prediction-based smartphone energy optimization for everyday location monitoring. In: *Proceedings of the 9th ACM conference on embedded networked sensor systems*, Seiten 82–95. ACM, 2011.

[CWSR14] CHOY, SHARON, BERNARD WONG, GWENDAL SIMON und CATHERINE ROSENBERG: A hybrid edge-cloud architecture for reducing on-demand gaming latency. Multimedia Systems, 20(5):503–519, 2014.

[DA99] DEY, ANIND K und GREGORY D ABOWD: Towards a better understanding of context and context-awareness. Technischer Bericht, Georgia Institute of Technology, Atlanta, 1999.

[DDMGP15] DO, TRINH MINH TRI, OLIVIER DOUSSE, MARKUS MIETTINEN und DANIEL GATICA-PEREZ: A probabilistic kernel method for human mobility prediction with smartphones. Pervasive and Mobile Computing, 20:13–28, 2015.

[De16] DE, DEBASHIS: *Mobile Cloud Computing: Architectures, Algorithms and Applications*. CRC Press, 2016.

[Del16] DELL TECHNOLOGIES INC.: T7600 workstation technical guidebook. `http://partnerdirect.dell.com/sites/channel/Documents/Dell_Precision_T7600_Workstation_Technical_Guidebook.pdf`, 2016. aufgerufen am 31.12.2016.

[Dey00] DEY, ANIND K: *Providing architectural support for building context-aware applications*. Doktorarbeit, Georgia Institute of Technology, 2000.

[DGP10] DO, TRINH-MINH-TRI und DANIEL GATICA-PEREZ: By their apps you shall understand them: mining large-scale patterns of mobile phone usage. In: *Proceedings of the 9th international conference on mobile and ubiquitous multimedia*, Seite 27. ACM, 2010.

[DLNW13] DINH, HOANG T, CHONHO LEE, DUSIT NIYATO und PING WANG: A survey of mobile cloud computing: architecture, applications, and approaches. Wireless communications and mobile computing, 13(18):1587–1611, 2013.

[DPS13] DAHLMAN, ERIK, STEFAN PARKVALL und JOHAN SKOLD: *4G: LTE/LTE-advanced for mobile broadband*. Academic press, 2013.

[EKK+13] ETTER, VINCENT, MOHAMED KAFSI, EHSAN KAZEMI, MATTHIAS GROSSGLAUSER und PATRICK THIRAN: Where to go from here? mobility prediction from instantaneous information. Pervasive and Mobile Computing, 9(6):784–797, 2013.

[EP06] EAGLE, NATHAN und ALEX SANDY PENTLAND: Reality mining: sensing complex social systems. Personal and ubiquitous computing, 10(4):255–268, 2006.

[Erl05] ERL, THOMAS: *Service-oriented architecture (SOA): concepts, technology, and design*. Prentice Hall, 2005.

[ESM14] ELGENDY, MOSTAFA A, AHMED SHAWISH und MAHMOUD I MOUSSA: An enhanced version of the mcacc to augment the computing capabilities of mobile devices using cloud computing. In: *International Journal of Advanced Computer Science*

and Applications (IJACSA), Special Issue on Extended Papers from Science and Information Conference. Citeseer, 2014.

[ETS99] ETSI: Virtual home environment. http://www.etsi.org/deliver/etsi_eg/201700_201799/201717/01.04.02_60/eg_201717v010402p.pdf, 1999. aufgerufen am 31.12.2016.

[ETS14] ETSI: Mobile-edge computing- introductory technical white paper. https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_paper_v1%2018-09-14.pdf, 2014. aufgerufen am 31.12.2016.

[ETS15] ETSI: Mobile edge computing. http://www.etsi.org/technologies-clusters/technologies/mobile-edge-computing, 2015. aufgerufen am 31.12.2016.

[FC04] FAHY, PATRICK und SIOBHAN CLARKE: Cass–a middleware for mobile context-aware applications. In: *Workshop on context awareness, MobiSys*. Citeseer, 2004.

[Fer13] FERREIRA, D: *AWARE: A mobile context instrumentation middleware to collaboratively understand human behavior*. Doktorarbeit, University of Oulu, 2013.

[FHLM01] FLOCH, JACQUELINE, SVEIN HALLSTEINSEN, ARNE LIE und HANS I MYRHAUG: A reference model for context-aware mobile services. In: *Proceedings of NIK*, 2001.

[FK13] FITZEK, FRANK HP und MARCOS D KATZ: *Mobile clouds: Exploiting distributed resources in wireless, mobile and social networks*. John Wiley & Sons, 2013.

[Fli12] FLINN, JASON: *Cyber foraging: Bridging mobile and cloud computing*. Morgan & Claypool Publishers, 2012.

[Flo15] FLORES, HUBER: *Service-oriented and evidence-aware mobile cloud computing*. Doktorarbeit, Universitatis Tartuensis, 2015.

[FLR13] FERNANDO, NIROSHINIE, SENG W LOKE und WENNY RAHAYU: Mobile cloud computing: A survey. Future Generation Computer Systems, 29(1):84–106, 2013.

[For16] FORWARDADGROUP GMBH: Mobile Effects 2016 - Wie wir Smartphone und Tablet im Alltag nutzen. http://www.burda-forward.de/uploads/tx_mjstudien/BF_Digitalmarkt_MobileEffects2016.pdf, 2016. aufgerufen am 31.12.2016.

[FPSS96] FAYYAD, USAMA, GREGORY PIATETSKY-SHAPIRO und PADHRAIC SMYTH: From data mining to knowledge discovery in databases. AI magazine, 17(3):37, 1996.

[FPV98] FUGGETTA, ALFONSO, GIAN PIETRO PICCO und GIOVANNI VIGNA: Understanding code mobility. IEEE Transactions on software engineering, 24(5):342–361, 1998.

[FRL05] FOK, C-L, G-C ROMAN und CHENYANG LU: Mobile agent middleware for sensor networks: An application case study. In: *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, Seiten 382–387. IEEE, 2005.

[Fru16] FRUMUSANU, ANDREI: The samsung exynos 7420 deep dive - inside a modern 14nm soc. http://www.anandtech.com/show/9330/exynos-7420-deep-dive/5, 2016. aufgerufen am 31.12.2016.

[FSB14] FLORES, HUBER, SATISH NARAYANA SRIRAMA und RAJKUMAR BUYYA: Computational offloading or data binding? bridging the cloud infrastructure to the proximity of the mobile user. In: *Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference on*, Seiten 10–18. IEEE, 2014.

[Fuc09] FUCHSS, THOMAS: *Mobile Computing*. Hanser, 2009.

[Fur12] FURTHMÜLLER, JOCHEN: *Verfahren und Protokolle für energiebewusste, gemeinsame Ressourcenverwendung mit mobilen Geräten*. KIT Scientific Publishing, 2012.

[FZRL08] FOSTER, IAN, YONG ZHAO, IOAN RAICU und SHIYONG LU: Cloud computing and grid computing 360-degree compared. In: *2008 Grid Computing Environments Workshop*, Seiten 1–10. Ieee, 2008.

[Gar15] GARTNER, INC.: Gartner says 6.4 billion connected "things" will be in use in 2016, up 30 percent from 2015. http://www.

gartner.com/newsroom/id/3165317, 2015. aufgerufen am 31.12.2016.

[Gar16] GARTNER, INC.: Gartner says five of top 10 worldwide mobile phone vendors increased sales in second quarter of 2016. http://www.gartner.com/newsroom/id/3415117, 2016. aufgerufen am 31.12.2016.

[GBE+09] GEIHS, KURT, PAOLO BARONE, FRANK ELIASSEN, JACQUELINE FLOCH, ROLF FRICKE, ELI GJORVEN, SVEIN HALLSTEINSEN, GEIR HORN, MOHAMMAD ULLAH KHAN, ALESSANDRO MAMELLI und OTHERS: A comprehensive solution for application-level adaptation. Software: Practice and Experience, 39(4):385–422, 2009.

[GC04] GOYAL, SACHIN und JOHN CARTER: A lightweight secure cyber foraging infrastructure for resource-constrained devices. In: *Mobile Computing Systems and Applications, 2004. WMCSA 2004. Sixth IEEE Workshop on*, Seiten 186–195. IEEE, 2004.

[GCGBDRD16] GUERRERO-CONTRERAS, GABRIEL, JOSE LUIS GARRIDO, SARA BALDERAS-DIAZ und CARLOS RODRIGUEZ-DOMINGUEZ: A context-aware architecture supporting service availability in mobile cloud computing. IEEE Transactions on Services Computing, 13(9):1939–1374, 2016.

[GDB+13] GOLUMBIC, ELANA M ZION, NAI DING, STEPHAN BICKEL, PETER LAKATOS, CATHERINE A SCHEVON, GUY M MCKHANN, ROBERT R GOODMAN, RONALD EMERSON, ASHESH D MEHTA, JONATHAN Z SIMON und OTHERS: Mechanisms underlying selective neuronal tracking of attended speech at a cocktail party. Neuron, 77(5):980–991, 2013.

[GE03] GUYON, ISABELLE und ANDRÉ ELISSEEFF: An introduction to variable and feature selection. Journal of machine learning research, 3(Mar):1157–1182, 2003.

[Gei08] GEIHS, KURT: Selbst-adaptive software. Informatik-Spektrum, 31(2):133–145, 2008.

[GGG05] GUMMADI, RAMAKRISHNA, OMPRAKASH GNAWALI und RAMESH GOVINDAN: Macro-programming wireless sensor networks using kairos. In: *International Conference on Distribu-*

ted Computing in Sensor Systems, Seiten 126–140. Springer, 2005.

[GJM+12] GORDON, MARK S, D ANOUSHE JAMSHIDI, SCOTT MAHLKE, Z MORLEY MAO und XU CHEN: Comet: code offload by migrating execution transparently. In: *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, Seiten 93–106, 2012.

[GNM+03] GU, XIAOHUI, KLARA NAHRSTEDT, ALAN MESSER, IRA GREENBERG und DEJAN MILOJICIC: Adaptive offloading inference for delivering applications in pervasive computing environments. In: *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, PERCOM '03, Washington, DC, USA, 2003. IEEE Computer Society.

[GNS+14] GANI, ABDULLAH, GOLAM MOKATDER NAYEEM, MUHAMMAD SHIRAZ, MEHDI SOOKHAK, MD WHAIDUZZAMAN und SULEMAN KHAN: A review on interworking and mobility techniques for seamless connectivity in mobile cloud computing. Journal of Network and Computer Applications, 43:84–102, 2014.

[Goo16] GOOGLE, INC.: App engine - platform as a service. `https://cloud.google.com/appengine/`, 2016. aufgerufen am 31.12.2016.

[GPK13] GOMES, JOAO BÁRTOLO, CLIFTON PHUA und SHONALI KRISHNASWAMY: Where will you go? mobile data mining for next place prediction. In: *International Conference on Data Warehousing and Knowledge Discovery*, Seiten 146–158. Springer, 2013.

[GPP+12] GHIANI, GIUSEPPE, FABIO PATERNÒ, JUSSI POLET, VILLE ANTILA und JANI MÄNTYJÄRVI: A context-dependent environment for web application migration. In: *Proceedings of the International Working Conference on Advanced Visual Interfaces*, Seiten 815–817, 2012.

[GPSV10] GERBER, ALEXANDRE, JEFFREY PANG, OLIVER SPATSCHECK und SHOBHA VENKATARAMAN: Speed testing without speed tests: estimating achievable download speed from passive measurements. In: *Proceedings of the 10th ACM SIGCOMM*

conference on Internet measurement, Seiten 424–430. ACM, 2010.

[GPZ05] GU, TAO, HUNG KENG PUNG und DA QING ZHANG: A service-oriented middleware for building context-aware services. Journal of Network and computer applications, 28(1):1–18, 2005.

[GPZA13] GRAZIOLI, ALESSANDRO, MARCO PICONE, FRANCESCO ZANICHELLI und MICHELE AMORETTI: Code migration in mobile clouds with the nam4j middleware. In: *2013 IEEE 14th International Conference on Mobile Data Management*, Band 2, Seiten 194–199. IEEE, 2013.

[GRA12] GIURGIU, IOANA, ORIANA RIVA und GUSTAVO ALONSO: Dynamic software deployment from clouds to mobile devices. In: *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, Seiten 394–414. Springer, 2012.

[GRJ+09] GIURGIU, IOANA, ORIANA RIVA, DEJAN JURIC, IVAN KRIVULEV und GUSTAVO ALONSO: Calling the cloud: enabling mobile phones as interfaces to cloud applications. In: *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, Seiten 83–102. Springer, 2009.

[GRWK09] GEIHS, KURT, ROLAND REICHLE, MICHAEL WAGNER und MOHAMMAD ULLAH KHAN: Modeling of context-aware self-adaptive applications in ubiquitous and service-oriented environments. In: *Software engineering for self-adaptive systems*, Seiten 146–163. Springer, 2009.

[Gut99] GUTTMAN, ERIK: Service location protocol: Automatic discovery of ip network services. IEEE Internet Computing, 3(4):71–80, 1999.

[GZK+11] GUO, YAO, LIN ZHANG, JUNJUN KONG, JIAN SUN, TAO FENG und XIANGQUN CHEN: Jupiter: transparent augmentation of smartphone capabilities through cloud computing. In: *Proceedings of the 3rd ACM SOSP Workshop on Networking, Systems, and Applications on Mobile Handhelds*, Seite 2. ACM, 2011.

[HAHZ15] HABAK, KARIM, MOSTAFA AMMAR, KHALED A HARRAS und ELLEN ZEGURA: Femto clouds: Leveraging mobile devices to provide cloud service at the edge. In: *2015 IEEE 8th Inter-*

national Conference on Cloud Computing, Seiten 9–16. IEEE, 2015.

[Ham05] HAMMERSCHALL, ULRIKE: *Verteilte Systeme und Anwendungen*. Informatik Verteilte Systeme. Pearson Studium, 2005.

[HBC13] HASSAN, MOHAMMED A, KSHITIZ BHATTARAI und SONGQING CHEN: vups: Virtually unifying personal storage for fast and pervasive data accesses. In: *Mobile Computing, Applications, and Services: Fourth International Conference, MobiCASE 2012, Seattle, WA, USA, October 2012. Revised Selected Papers*, Band 110, Seite 186. Springer, 2013.

[HBS73] HEWITT, CARL, PETER BISHOP und RICHARD STEIGER: Session 8 formalisms for artificial intelligence a universal modular actor formalism for artificial intelligence. In: *Advance Papers of the Conference*, Band 3, Seite 235. Stanford Research Institute, 1973.

[HCL10] HUERTA-CANEPA, GONZALO und DONGMAN LEE: A virtual cloud computing provider for mobile devices. In: *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, Seite 6. ACM, 2010.

[HCN08] HIRSCHFELD, ROBERT, PASCAL COSTANZA und OSCAR NIERSTRASZ: Context-oriented programming. Journal of Object Technology, 7(3):125–151, 2008.

[HCS13] HUNG, SHIH-HAO, YONG-WEI CHEN und JENG-PENG SHIEH: Creating pervasive, dynamic, scalable android applications. In: *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2013 Seventh International Conference on*, Seiten 43–50. IEEE, 2013.

[Hen03] HENRICKSEN, KAREN: *A framework for context-aware pervasive computing applications*. Doktorarbeit, University of Queensland Queensland, 2003.

[HHY15] HABAK, KARIM, KHALED A HARRAS und MOUSTAFA YOUSSEF: Bandwidth aggregation techniques in heterogeneous multi-homed devices: A survey. Computer Networks, 92:168–188, 2015.

[Him04] HIMBERG, JOHAN: *From insights to innovations: data mining, visualization, and user interfaces*. Helsinki University of Tech-

nology, 2004.

[HIMB05] HENRICKSEN, KAREN, JADWIGA INDULSKA, TED MCFADDEN und SASITHARAN BALASUBRAMANIAM: Middleware for distributed context-aware systems. In: *OTM Confederated International Conferences: On the Move to Meaningful Internet Systems*, Seiten 846–863. Springer, 2005.

[HIR03] HENRICKSEN, KAREN, JADWIGA INDULSKA und ANDRY RAKOTONIRAINY: Generating context management infrastructure from high-level context models. In: *4th International Conference on Mobile Data Management (MDM)-Industrial Track*. Citeseer, 2003.

[HL97] HALL, DAVID L und JAMES LLINAS: An introduction to multisensor data fusion. Proceedings of the IEEE, 85(1):6–23, 1997.

[HMNS03] HANSMANN, UWE, LOTHAR MERK, MARTIN S NICKLOUS und THOMAS STOBER: *Pervasive computing: The mobile world*. Springer Science & Business Media, 2003.

[HMNS13] HANSMANN, UWE, LOTHAR MERK, MARTIN S NICKLOUS und THOMAS STOBER: *Pervasive computing handbook*. Springer Science & Business Media, 2013.

[HQG+13] HUANG, JUNXIAN, FENG QIAN, YIHUA GUO, YUANYUAN ZHOU, QIANG XU, Z MORLEY MAO, SUBHABRATA SEN und OLIVER SPATSCHECK: An in-depth study of lte: effect of network protocol and application behavior on performance. In: *ACM SIGCOMM Computer Communication Review*, Band 43.4, Seiten 363–374. ACM, 2013.

[HRH09] HYNES, GEAROID, VINNY REYNOLDS und MANFRED HAUSWIRTH: A context lifecycle for web-based context management services. In: *Proceedings of the 4th European conference on Smart sensing and context*, Seiten 51–65. Springer-Verlag, 2009.

[HS05] HUHNS, MICHAEL N und MUNINDAR P SINGH: Service-oriented computing: Key concepts and principles. IEEE Internet computing, 9(1):75–81, 2005.

[HS08] HAUBERG, SØREN und JAKOB SLOTH: An efficient algorithm for modelling duration inÂ hiddenÂ markovÂ models, with a

dramatic application. Journal of Mathematical Imaging and Vision, 31(2):165–170, Jul 2008.

[HSC12] HUNG, SHIH-HAO, JENG-PENG SHIEH und YONG-WEI CHEN: A profile-driven dynamic application offloading scheme for android systems. In: *Consumer Electronics (GCCE), 2012 IEEE 1st Global Conference on*, Seiten 540–541. IEEE, 2012.

[HSMK09] HEAVEN, WILLIAM, DANIEL SYKES, JEFF MAGEE und JEFF KRAMER: A case study in goal-driven architectural adaptation. In: *Software Engineering for Self-Adaptive Systems*, Seiten 109–127. Springer, 2009.

[HZKL10] HUANG, DIJIANG, XINWEN ZHANG, MYONG KANG und JIM LUO: Mobicloud: building secure cloud framework for mobile computing and communication. In: *Service Oriented System Engineering (SOSE), 2010 Fifth IEEE International Symposium on*, Seiten 27–34. Ieee, 2010.

[IDC16a] IDC RESEARCH, INC.: Apple, huawei, and xiaomi finish 2015 with above average year-over-year growth, as worldwide smartphone shipments surpass 1.4 billion for the year, according to idc. `https://www.idc.com/getdoc.jsp?containerId=prUS40980416`, 2016. aufgerufen am 31.12.2016.

[IDC16b] IDC RESEARCH, INC.: Idc forecasts worldwide shipments of wearables to surpass 200 million in 2019, driven by strong smartwatch growth and the emergence of smarter watches. `http://www.idc.com/getdoc.jsp?containerId=prUS41100116`, 2016. aufgerufen am 31.12.2016.

[Idi16] IDIAP RESEARCH INSTITUTE: Mobile data challenge (mdc) dataset. `https://www.idiap.ch/dataset/mdc`, 2016. aufgerufen am 31.12.2016.

[IEE10] IEEE STANDARDS ASSOCIATION: Ieee sa - 802.11n-2009 - ieee standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless lan medium access control (mac)and physical layer (phy) specifications. `http://standards.ieee.org/findstds/standard/802.11n-2009.html`, 2010. aufgerufen am 31.12.2016.

[IEE13] IEEE STANDARDS ASSOCIATION: Ieee std 802.11ac™-

2013, ieee standard for information technology—telecommunications and information exchange between systems—local and metropolitan area networks—specific requirements—part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications—amendment 4: Enhancements for very high throughput for operation in bands below 6 ghz. http://standards.ieee.org/findstds/standard/802.11ac-2013.html, 2013. aufgerufen am 31.12.2016.

[Ind16] INDIEGOGO, INC.: Meet the pilot: Smart earpiece language translator. https://www.indiegogo.com/projects/meet-the-pilot-smart-earpiece-language-translator\-headphones-travel, 2016. aufgerufen am 31.12.2016.

[Int11] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: ISO/IEC 25010:2011 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. http://www.iso.org/iso/catalogue_detail.htm?csnumber=35733, 2011. aufgerufen am 31.12.2016.

[JCY14] JIA, MIKE, JIANNONG CAO und LEI YANG: Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing. In: *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, Seiten 352–357. IEEE, 2014.

[Jer14] JERONIMO, FRANCISCO: Idc survey finds "top 10 smartphone purchase drivers" (source: Idc's consumerscape 360). https://twitter.com/fjeronimo/status/465896917298577409, 2014. aufgerufen am 31.12.2016.

[JK11] JEONG, J. und H. KIM: Cloud systems and their applications for mobile devices. In: *MOBILITY 2011: The First International Conference on Mobile Services, Resources, and Users*, Seiten 135–138, 2011.

[JL13] JANDER, KAI und WINFRIED LAMERSDORF: Compact and efficient agent messaging. In: M. DASTANI, J.F. HÜBNER, B. LOGAN (Herausgeber): *Tenth International Workshop on Programming Multi-Agent Systems (ProMAS'12)*, Band Programming Multiagent Systems der Reihe *Lecture Notes in Computer Science*, Seiten 108–122. Springer Berlin / Heidelberg, 7 2013. ISBN: 978-1-4673-1807-5.

[JLHB88] JUL, ERIC, HENRY LEVY, NORMAN HUTCHINSON und ANDREW BLACK: Fine-grained mobility in the emerald system. ACM Transactions on Computer Systems (TOCS), 6(1):109–133, 1988.

[JLJ+10] JENSEN, BJØRN SAND, JAKOB EG LARSEN, KRISTIAN JENSEN, JAN LARSEN und LARS KAI HANSEN: Estimating human predictability from mobile sensor data. In: *Machine Learning for Signal Processing (MLSP), 2010 IEEE International Workshop on*, Seiten 196–201. IEEE, 2010.

[JW03] JERONIMO, MICHAEL und JACK WEAST: *UPnP design by example: a software developer's guide to universal plug and play*. Intel Press, 2003.

[KAH+12] KOSTA, SOKOL, ANDRIUS AUCINAS, PAN HUI, RICHARD MORTIER und XINWEN ZHANG: Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In: *INFOCOM, 2012 Proceedings IEEE*, Seiten 945–953. IEEE, 2012.

[Kal14] KALINOWSKI, JULIAN: Analyse und integration anwendungsorientierter middleware auf mobilen geräten im kontext des android-betriebssystems. Diplomarbeit, Universität Hamburg, Fachbereich Informatik, 2014.

[KBD+10] KIUKKONEN, NIKO, JAN BLOM, OLIVIER DOUSSE, DANIEL GATICA-PEREZ und JUHA LAURILA: Towards rich mobile phone datasets: Lausanne data collection campaign. In: *In Proceedings of the 7th International Conference on Pervasive Services*, 2010.

[KC03] KEPHART, JEFFREY O und DAVID M CHESS: The vision of autonomic computing. Computer, 36(1):41–50, 2003.

[KCK11] KOVACHEV, DEJAN, YIWEI CAO und RALF KLAMMA: Mobile cloud computing: a comparison of application models. In: *arXiv preprint arXiv:1107.4940*, 2011.

[KCRG15] KIM, KEUNSOO, BENJAMIN Y CHO, WON WOO RO und JEAN-LUC GAUDIOT: Network variation and fault tolerant performance acceleration in mobile devices with simultaneous remote execution. IEEE Transactions on Computers, 64(10):2862–2874, 2015.

[KD07] KEPHART, JEFFREY O und RAJARSHI DAS: Achieving self-management via utility functions. IEEE Internet Computing, 11(1):40–48, 2007.

[Kem11] KEMPPAINEN, MATTI: Mobile computation offloading: A context-driven approach. In: *Aalto University T-110.5190 Seminar on Internetworking*, 2011.

[KGMG07] KOTHARI, NUPUR, RAMAKRISHNA GUMMADI, TODD MILLSTEIN und RAMESH GOVINDAN: Reliable and efficient programming abstractions for wireless sensor networks. In: *ACM SIGPLAN Notices*, Band 42.6, Seiten 200–210. ACM, 2007.

[KH07] KREUTZER, MICHAEL und MANUEL HARTL: A toolbox of mechanisms for robust and scalable service discovery in mobile ad-hoc networks. In: *Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on*, Seiten 487–496. IEEE, 2007.

[Kin12] KIN LANE: Rise of mobile backend as a service (mbaas) api stacks. `http://apievangelist.com/2012/06/03/rise-of-mobile-backend-as-a-service-mbaas-api-\stacks/`, 2012. aufgerufen am 31.12.2016.

[Kit17] KITZKY, YANNIC: Kontextbezogenes mobiles cloud computing: Entwicklung einer komponentenbasierten anwendung zur gesichtserkennung. Bachelorarbeit, Universität Hamburg, Fachbereich Informatik, Vogt-Kölln-Str. 30, 22527 Hamburg, Germany, 7 2017.

[KKS13] KÜHN, ROMINA, CHRISTINE KELLER und THOMAS SCHLEGEL: A context taxonomy supporting public system design. In: *Proc 1st int workshop model-based interact ubiquitous syst. Pisa (Italy)*. Citeseer, 2013.

[KLLB13] KUMAR, KARTHIK, JIBANG LIU, YUNG-HSIANG LU und BHARAT BHARGAVA: A survey of computation offloading for mobile systems. Mobile Networks and Applications, 18(1):129–140, 2013.

[KLM+97] KICZALES, GREGOR, JOHN LAMPING, ANURAG MENDHEKAR, CHRIS MAEDA, CRISTINA LOPES, JEAN-MARC LOINGTIER und JOHN IRWIN: Aspect-oriented programming. In: *European conference on object-oriented programming*, Seiten 220–242. Springer, 1997.

[KM85] KRAMER, JEFF und JEFF MAGEE: Dynamic configuration for distributed systems. IEEE Transactions on Software Engineering, 4:424–436, 1985.

[KMS+10] KÖNÖNEN, VILLE, JANI MÄNTYJÄRVI, HEIDI SIMILÄ, JUHA PÄRKKÄ und MIIKKA ERMES: Automatic feature selection for context recognition in mobile devices. Pervasive and Mobile Computing, 6(2):181–197, 2010.

[KMYI09] KAMISAKA, DAISUKE, SHIGEKI MURAMATSU, HIROYUKI YOKOYAMA und TAKESHI IWAMOTO: Operation prediction for context-aware user interfaces of mobile phones. In: *Applications and the Internet, 2009. SAINT'09. Ninth Annual International Symposium on*, Seiten 16–22. IEEE, 2009.

[KNP15] KRAWCZYK, HENRYK, MICHAŁ NYKIEL und JERZY PROFICZ: Mobile offloading framework: Solution for optimizing mobile applications using cloud computingg. In: *International Conference on Computer Networks*, Seiten 293–305. Springer, 2015.

[KPKB10] KEMP, ROELOF, NICHOLAS PALMER, THILO KIELMANN und HENRI BAL: Cuckoo: a computation offloading framework for smartphones. In: *International Conference on Mobile Computing, Applications, and Services*, Seiten 59–79. Springer, 2010.

[KPP10] KAKOUSIS, KONSTANTINOS, NEARCHOS PASPALLIS und GEORGE ANGELOS PAPADOPOULOS: A survey of software adaptation in mobile and ubiquitous computing. Enterprise Information Systems, 4(4):355–389, 2010.

[Kri10] KRISTENSEN, MADS DARØ: Scavenger: Transparent development of efficient cyber foraging applications. In: *Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on*, Seiten 217–226. IEEE, 2010.

[KYK+16] KWON, YONGIN, HAYOON YI, DONGHYUN KWON, SEUNGJUN YANG, YEONGPIL CHO und YUNHEUNG PAEK: Precise execution offloading for applications with dynamic behavior in mobile cloud computing. Pervasive and Mobile Computing, 27:58–74, 2016.

[KZ08] KANSAL, AMAN und FENG ZHAO: Fine-grained energy profiling for power-aware application design. ACM SIGMETRICS Performance Evaluation Review, 36(2):26–31, 2008.

[LAS+15] Liu, Jieyao, Ejaz Ahmed, Muhammad Shiraz, Abdullah Gani, Rajkumar Buyya und Ahsan Qureshi: Application partitioning algorithms in mobile cloud computing: Taxonomy, review and future directions. Journal of Network and Computer Applications, 48:99–117, 2015.

[LBLX13] Li, Jiwei, Kai Bu, Xuan Liu und Bin Xiao: Enda: Embracing network inconsistency for dynamic application offloading in mobile cloud computing. In: *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*, Seiten 39–44. ACM, 2013.

[LCTB+06] Lagar-Cavilla, Andres, Niraj Tolia, Rajesh Balan, Eyal De Lara, Mahadev Satyanarayanan und David O'Hallaron: Dimorphic computing. Technischer Bericht, School of Computer Science-Carnegie Mellon University, Pittsburgh, 2006.

[LD10] Lim, Brian Y und Anind K Dey: Toolkit to support intelligibility in context-aware applications. In: *Proceedings of the 12th ACM international conference on Ubiquitous computing*, Seiten 13–22. ACM, 2010.

[LEMR15] Li, Xin, Martina Eckert, José-Fernán Martinez und Gregorio Rubio: Context aware middleware architectures: survey and challenges. Sensors, 15(8):20570–20607, 2015.

[LGPA+13] Laurila, Juha K, Daniel Gatica-Perez, Imad Aad, Jan Blom, Olivier Bornet, Trinh Minh Tri Do, Olivier Dousse, Julien Eberle und Markus Miettinen: From big smartphone data to worldwide research: The mobile data challenge. Pervasive and Mobile Computing, 9(6):752–771, 2013.

[LHB11] Lukowicz, Paul, Dirk Helbing und Steven Bishop: From context awareness to socially interactive computing. Pervasive Adaptation, 38(1):32–41, 2011.

[LHL12] Lin, Miao, Wen-Jing Hsu und Zhuo Qi Lee: Predictability of individuals' mobility with high-resolution positioning data. In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, Seiten 381–390. ACM, 2012.

[LLS11] Lu, Yan, Shipeng Li und Huifeng Shen: Virtualized screen: A third element for cloud-mobile convergence. IEEE

MultiMedia, 18(2):4–11, 2011.

[LLY+10] LEE, KYUNGHAN, JOOHYUN LEE, YUNG YI, INJONG RHEE und SONG CHONG: Mobile data offloading: how much can wifi deliver? In: *ACM SIGCOMM Computer Communication Review*, Band 40.4, Seiten 425–426. ACM, 2010.

[LML+10] LANE, NICHOLAS D, EMILIANO MILUZZO, HONG LU, DANIEL PEEBLES, TANZEEM CHOUDHURY und ANDREW T CAMPBELL: A survey of mobile phone sensing. IEEE Communications magazine, 48(9):140–150, 2010.

[LQLL10] LIU, ZHAOBIN, WENYU QU, WEIJIANG LIU und KEQIU LI: Xen live migration with slowdown scheduling algorithm. In: *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2010 International Conference on*, Seiten 215–221. IEEE, 2010.

[LS13] LEE, KILHO und INSIK SHIN: User mobility-aware decision making for mobile computation offloading. In: *Cyber-Physical Systems, Networks, and Applications (CPSNA), 2013 IEEE 1st International Conference on*, Seiten 116–119. IEEE, 2013.

[LSJ+13] LIU, FANGMING, PENG SHU, HAI JIN, LINJIE DING, JIE YU, DI NIU und BO LI: Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications. IEEE Wireless communications, 20(3):14–22, 2013.

[LT11] LAGERSPETZ, EEMIL und SASU TARKOMA: Mobile search and the cloud: The benefits of offloading. In: *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, Seiten 117–122. IEEE, 2011.

[LTE16] LTE-ANBIETER.INFO: Asu wert - signalstärke messen und interpretieren. `http://www.lte-anbieter.info/technik/asu.php`, 2016. aufgerufen am 31.12.2016.

[MAE14] MAEMO.ORG: Pys60 documentation. `http://pys60.garage.maemo.org/doc/s60/module-sysinfo.html`, 2014. aufgerufen am 31.12.2016.

[Mar09] MARINELLI, EUGENE E: Hyrax: cloud computing on mobile devices using mapreduce. Technischer Bericht, School of Computer Science-Carnegie Mellon University, Pittsburgh, 2009.

[May04] MAYRHOFER, RENE: *An Architecture for Context Prediction*. Doktorarbeit, Schriften der Johannes-Kepler-Universität Linz, 2004.

[MBB09] MIAN, ADNAN NOOR, ROBERTO BALDONI und ROBERTO BERALDI: A survey of service discovery protocols in multihop mobile ad hoc networks. IEEE Pervasive computing, 8(1):66–74, 2009.

[MBF+15] MICHELINAKIS, FOIVOS, NICOLA BUI, GUIDO FIORAVANTTI, JOERG WIDMER, FABIAN KAUP und DAVID HAUSHEER: Lightweight mobile bandwidth availability measurement. In: *IFIP Networking Conference (IFIP Networking), 2015*, Seiten 1–9. IEEE, 2015.

[MDP+00] MILOJIČIĆ, DEJAN S, FRED DOUGLIS, YVES PAINDAVEINE, RICHARD WHEELER und SONGNIAN ZHOU: Process migration. ACM Computing Surveys (CSUR), 32(3):241–299, 2000.

[Mer16] MERRIAM-WEBSTER, INC.: Definition of context. `https://www.merriam-webster.com/dictionary/context`, 2016. aufgerufen am 31.12.2016.

[Mey16] MEYER, ANNE-VICTORIA: Kontextbezogene energiegewinnung mobiler geräte. Bachelorarbeit, Universität Hamburg, Fachbereich Informatik, Vogt-Kölln-Str. 30, 22527 Hamburg, Germany, 8 2016.

[MG10] MELL, PETER und TIM GRANCE: The nist definition of cloud computing. Communications of the ACM, 53(6):50, 2010.

[MGB+02] MESSER, ALAN, IRA GREENBERG, PHILIPPE BERNADAT, DEJAN MILOJICIC, DEQING CHEN, THOMAS J GIULI und XIAOHUI GU: Towards a distributed platform for resource-constrained devices. In: *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on*, Seiten 43–51. IEEE, 2002.

[MGC13] MARCU, MARIUS, NADIA GHIATA und VLADIMIR CRETU: Extracting high-level user context from low-level mobile sensors data. In: *Applied Computational Intelligence and Informatics (SACI), 2013 IEEE 8th International Symposium on*, Seiten 449–454. IEEE, 2013.

[MGG+14] MEILÄNDER, DOMINIK, FRANK GLINKA, SERGEI GORLATCH,

LI LIN, WEI ZHANG und XIAOFEI LIAO: Using mobile cloud computing for real-time online applications. In: *Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference on*, Seiten 48–56. IEEE, 2014.

[MGL+11] MARCH, VERDI, YAN GU, ERWIN LEONARDI, GEORGE GOH, MARKUS KIRCHBERG und BU SUNG LEE: μcloud: towards a new paradigm of rich mobile applications. Procedia Computer Science, 5:618–624, 2011.

[MHAU15] MATTHIES, DENYS JC, MARIAN HAESCHER, REBEKKA ALM und BODO URBAN: Properties of a peripheral head-mounted display (phmd). In: *International Conference on Human-Computer Interaction*, Seiten 208–213. Springer, 2015.

[Mic14] MICROSOFT CORPORATION: Satya nadella: Mobile first, cloud first press briefing | news center. https://news.microsoft.com/2014/03/27/satya-nadella-mobile-first-cloud-first-press\-briefing/, 2014. aufgerufen am 31.12.2016.

[Mic15] MICROSOFT CORPORATION: Irides: Attaining quality, responsiveness and mobility for virtual reality head-mounted displays - microsoft research. http://research.microsoft.com/apps/video/default.aspx?id=246323, 2015. aufgerufen am 31.12.2016.

[Mic16a] MICROSOFT CORPORATION: Office 365 home: Bürosoftware für die private nutzung kaufen. https://products.office.com/de-de/office-365-home, 2016. aufgerufen am 31.12.2016.

[Mic16b] MICROSOFT CORPORATION: Skype translator. https://www.skype.com/de/features/skype-translator/, 2016. aufgerufen am 31.12.2016.

[Min16] MINDTREE LTD.: Ipv6 stack over bluetooth smart or 802.15.4. http://www.mindtree.com/solutions/bluetooth-technology/ipv6stack, 2016. aufgerufen am 31.12.2016.

[MLF+08] MILUZZO, EMILIANO, NICHOLAS D LANE, KRISTÓF FODOR, RONALD PETERSON, HONG LU, MIRCO MUSOLESI, SHANE B EISENMAN, XIAO ZHENG und ANDREW T CAMPBELL: Sen-

sing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In: *Proceedings of the 6th ACM conference on Embedded network sensor systems*, Seiten 337–350. ACM, 2008.

[MLW11] MA, RICKY KK, KING TIN LAM und CHO-LI WANG: excloud: Transparent runtime support for scaling mobile applications in cloud. In: *Cloud and Service Computing (CSC), 2011 International Conference on*, Seiten 103–110. IEEE, 2011.

[MM09] MURARASU, ALIN FLORINDOR und THOMAS MAGEDANZ: Mobile middleware solution for automatic reconfiguration of applications. In: *Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference on*, Seiten 1049–1055. IEEE, 2009.

[MMAM+14] MIZOUNI, RABEB, MOHAMMAD ABU MATAR, ZAID AL MAHMOUD, SALWA ALZAHMI und AZIZ SALAH: A framework for context-aware self-adaptive mobile applications spl. Expert Systems with applications, 41(16):7549–7564, 2014.

[MMR+13] MISRA, PRASANT KUMAR, LUCA MOTTOLA, SHAHID RAZA, SIMON DUQUENNOY, NICOLAS TSIFTES, JOEL HOGLUND und THIEMO VOIGT: Supporting cyber-physical systems with wireless sensor networks: An outlook of software and services. Journal of the Indian Institute of Science, 93(3):463–486, 2013.

[MN10] MIETTINEN, ANTTI P und JUKKA K NURMINEN: Energy efficiency of mobile clients in cloud computing. HotCloud, 10:4–4, 2010.

[MPF+10] MUSOLESI, MIRCO, MATTIA PIRACCINI, KRISTOF FODOR, ANTONIO CORRADI und ANDREW T CAMPBELL: Supporting energy-efficient uploading strategies for continuous sensing applications on mobile phones. In: *International Conference on Pervasive Computing*, Seiten 355–372. Springer, 2010.

[MRF03] MAYRHOFER, RENE, HARALD RADI und ALOIS FERSCHA: Recognizing and predicting context by learning from user behavior. In: SCHREINER, W., G. KOTSIS, A. FERSCHA und K. IBRAHIM (Herausgeber): *The International Conference On Advances in Mobile Multimedia (MoMM2003)*, Band 171, Seiten 25–35. Austrian Computer Society (OCG), 2003.

[MRST10] MANJUNATHA, ASHWIN, AJITH RANABAHU, AMIT SHETH

und KRISHNAPRASAD THIRUNARAYAN: Power of clouds in your pocket: An efficient approach for cloud mobile hybrid application development. In: *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, Seiten 496–503. IEEE, 2010.

[MS14] MEHMETI, FIDAN und THRASYVOULOS SPYROPOULOS: Is it worth to be patient? analysis and optimization of delayed mobile data offloading. In: *INFOCOM, 2014 Proceedings IEEE*, Seiten 2364–2372. IEEE, 2014.

[MSCN13] MANWEILER, JUSTIN, NAVEEN SANTHAPURI, ROMIT ROY CHOUDHURY und SRIHARI NELAKUDITI: Predicting length of stay at wifi hotspots. In: *INFOCOM, 2013 Proceedings IEEE*, Seiten 3102–3110. IEEE, 2013.

[MSKC04] MCKINLEY, PHILIP K, SEYED MASOUD SADJADI, ERIC P KASTEN und BETTY HC CHENG: Composing adaptive software. COMPUTER, 37(7):0056–64, 2004.

[MSP14] MAGGIORE, G, C SANTOS und A PLAAT: Smarter smartphones: understanding and predicting user habits from gps sensor data. Procedia Computer Science, 34:297–304, 2014.

[Mül10] MÜLLER, FELIX: *Modellierung und Repräsentation des Kontextes von mobilen Nutzungsszenarien-Ein Rahmenwerk für mobile kontextsensitive Applikationen*. Doktorarbeit, Bibliothek der Fachhochschule Köln, 2010.

[MYHZ14] MAGURAWALAGE, CHATHURA M SARATHCHANDRA, KUN YANG, LIANG HU und JIANMING ZHANG: Energy-efficient and network-aware offloading algorithm for mobile cloud computing. Computer Networks, 74:22–33, 2014.

[NCW+12] NGUYEN, L, HENG-TZE CHENG, PANG WU, SENAKA BUTHPITIYA und YING ZHANG: Pnlum: System for prediction of next location for users with mobility. In: *Nokia Mobile Data Challenge 2012 Workshop. p. Dedicated challenge*, Band 2, 2012.

[NKI+15] NUNNA, SWAROOP, APOSTOLOS KOUSARIDAS, MOHAMED IBRAHIM, MARKUS DILLINGER, CHRISTOPH THUEMMLER, HUBERTUS FEUSSNER und ARMIN SCHNEIDER: Enabling real-time context-aware collaboration through 5g and mobile edge computing. In: *Information Technology-New Generations*

(ITNG), 2015 12th International Conference on, Seiten 601–605. IEEE, 2015.

[NMF+05] NURMI, PETTERI, MIQUEL MARTIN, JOHN A FLANAGAN und OTHERS: Enabling proactiveness through context prediction. In: *Proceedings of the Workshop on Context Awareness for Proactive Systems, Helsinki*, Band 53. Citeseer, 2005.

[NMSH14] NIR, MANJINDER, ASHRAF MATRAWY und MARC ST-HILAIRE: An energy optimizing scheduler for mobile cloud computing environments. In: *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, Seiten 404–409. IEEE, 2014.

[NN08] NICHOLSON, ANTHONY J und BRIAN D NOBLE: Bread-crumbs: forecasting mobile connectivity. In: *Proceedings of the 14th ACM international conference on Mobile computing and networking*, Seiten 46–57. ACM, 2008.

[NNH+14] NIRJON, SHAHRIAR, ANGELA NICOARA, CHENG-HSIN HSU, JATINDER PAL SINGH und JOHN A STANKOVIC: Multinets: A system for real-time switching between multiple network interfaces on mobile devices. ACM Transactions on Embedded Computing Systems (TECS), 13(4s):121, 2014.

[Nok15] NOKIA SIEMENS NETWORKS: Intelligent base stations. `http://networks.nokia.com/sites/default/files/document/liquid_applications_wp.pdf`, 2015. aufgerufen am 31.12.2016.

[NSA14] NIU, JIANWEI, WENFANG SONG und MOHAMMED ATIQUZZAMAN: Bandwidth-adaptive partitioning for distributed execution optimization of mobile applications. Journal of Network and Computer Applications, 37:334–347, 2014.

[NSI+15] NIEMINEN, J., T. SAVOLAINEN, M. ISOMAKI, B. PATIL, Z. SHELBY und C. GOMEZ: Request for comments: 7668 ipv6 over bluetooth(r) low energy. `https://tools.ietf.org/html/rfc7668`, 2015. aufgerufen am 31.12.2016.

[NSN+97] NOBLE, BRIAN D, MAHADEV SATYANARAYANAN, DUSHYANTH NARAYANAN, JAMES ERIC TILTON, JASON FLINN und KEVIN R WALKER: Agile application-aware adaptation for mobility. ACM SIGOPS Operating Systems Review, 31(5):276–287, 1997.

[NTG+09] NEWTON, RYAN, SIVAN TOLEDO, LEWIS GIROD, HARI BALAKRISHNAN und SAMUEL MADDEN: Wishbone: profile-based partitioning for sensornet applications. In: *Proceedings of the 6th USENIX symposium on Networked systems design and implementation. Boston, Massachusetts: USENIX Association*, Seiten 395–408. USENIX Association, 2009.

[Nvi15] NVIDIA CORPORATION: Nvidia shield game streaming dienst geforce now. `http://shield.nvidia.de/game-streaming-with-geforce-now`, 2015. aufgerufen am 31.12.2016.

[OBL15a] ORSINI, GABRIEL, DIRK BADE und WINFRIED LAMERSDORF: Cloudaware: Towards context-adaptive mobile cloud computing. In: BADONNEL, R., J. XIAO, S. ATA, F. DE TURCK und V. GROZA (Herausgeber): *IFIP/IEEE International Symposium on Integrated Network Management, IM 2015, Ottawa, ON, Canada, 11-15 May, 2015*, Seiten 1190–1195. IFIP, IEEE Explore Washington/DC, USA, 5 2015.

[OBL15b] ORSINI, GABRIEL, DIRK BADE und WINFRIED LAMERSDORF: Computing at the mobile edge: Designing elastic android applications for computation offloading. In: *8th IFIP Wireless and Mobile Networking Conference, WMNC 2015, Munich, Germany, October 5-7, 2015*, Seiten 112–119. IEEE Explore Washington/DC, USA, 2015.

[OBL15c] ORSINI, GABRIEL, DIRK BADE und WINFRIED LAMERSDORF: Context-aware computation offloading for mobile cloud computing: Requirements analysis, survey and design guideline. In: *The 10th International Conference on Future Networks and Communications (FNC 2015) / The 12th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2015) / Affiliated Workshops, August 17-20, 2015, Belfort, France*, Procedia Computer Science, Seiten 10–17. Elsevier Science, 8 2015.

[OBL16a] ORSINI, GABRIEL, DIRK BADE und WINFRIED LAMERSDORF: Cloudaware: A context-adaptive middleware for mobile edge and cloud computing applications. In: *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, Seiten 216–221. IEEE Explore Washington/DC, USA, 9 2016.

[OBL16b] ORSINI, GABRIEL, DIRK BADE und WINFRIED LAMERSDORF:

Generic context adaptation for mobile cloud computing environments. In: *The 11th International Conference on Future Networks and Communications (FNC 2016) / The 13th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2016) / Affiliated Workshops, August 15-18, 2016, Montreal, Quebec, Canada*, Procedia Computer Science, Seiten 17–24. Elsevier Science, 8 2016.

[OBL17a] ORSINI, GABRIEL, DIRK BADE und WINFRIED LAMERSDORF: Cloudaware: Empowering context-aware self-adaptation for mobile applications. Transactions on Emerging Telecommunications Technologies, 2017. zur Veröffentlichung angenommen.

[OBL17b] ORSINI, GABRIEL, DIRK BADE und WINFRIED LAMERSDORF: Generic context adaptation for mobile cloud computing environments (extended version). Journal of Ambient Intelligence and Humanized Computing, 2017. zur Veröffentlichung angenommen, vorab Online-Publikation: `http://www.springer.com/-/3/AVzcNhYhvMFoNfTGtdeY`.

[Oh15] OH, YOOSOO: Activity context integration in mobile computing environments. In: *International Conference on Distributed, Ambient, and Pervasive Interactions*, Seiten 527–535. Springer, 2015.

[OKA+15] OTHMAN, MAZLIZA, ABDUL NASIR KHAN, SHAHBAZ AKHTAR ABID, SAJJAD AHMAD MADANI und OTHERS: Mobibyte: an application development model for mobile cloud computing. Journal of Grid Computing, 13(4):605–628, 2015.

[OMK+14] OTHMAN, MAZLIZA, SAJJAD AHMAD MADANI, SAMEE ULLAH KHAN und OTHERS: A survey of mobile cloud computing application models. IEEE Communications Surveys & Tutorials, 16(1):393–413, 2014.

[Ope16] OPENCV.ORG: OpenCV4Android SDK — OpenCV 2.4.13.0 documentation. `http://docs.opencv.org/2.4/doc/tutorials/introduction/android_binary_package/O4A_SDK.html`, 2016. aufgerufen am 31.12.2016.

[OXK+15] OTHMAN, MAZLIZA, FENG XIA, ABDUL NASIR KHAN und OTHERS: Context-aware mobile cloud computing and its challenges. IEEE Cloud Computing, 2(3):42–49, 2015.

[OYH07] OU, SHUMAO, KUN YANG und LIANG HU: Cross: a combined routing and surrogate selection algorithm for pervasive service offloading in mobile ad hoc environments. In: *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*, Seiten 720–725. IEEE, 2007.

[OYL06] OU, SHUMAO, KUN YANG und ANTONIO LIOTTA: An adaptive multi-constraint partitioning algorithm for offloading in pervasive systems. In: *Pervasive Computing and Communications, 2006. PerCom 2006. Fourth Annual IEEE International Conference on*, Seiten 10–pp. IEEE, 2006.

[OYZ07] OU, SHUMAO, KUN YANG und JIE ZHANG: An effective offloading middleware for pervasive services on mobile devices. Pervasive and Mobile Computing, 3(4):362–385, 2007.

[Pan04] PANDYA, RAJ: *Mobile and personal communication systems and services*, Band 13. John Wiley & Sons, 2004.

[Pas99] PASCOE, BOB: Salutation architectures and the newly defined service discovery protocols from microsoft and sun. Technischer Bericht, IBM, 1999.

[PB05] PREUVENEERS, DAVY und YOLANDE BERBERS: Automated context-driven composition of pervasive services to alleviate non-functional concerns. International Journal of Computing and Information Sciences, 3(2):19–28, 2005.

[PB13] POKAHR, ALEXANDER und LARS BRAUBACH: The active components approach for distributed systems development. International Journal of Parallel, Emergent and Distributed Systems, 28(4):321–369, 2013.

[PEN10] PEDDEMORS, ARJAN, HENK EERTINK und IGNAS NIEMEGEERS: Predicting mobility events on personal devices. Pervasive and Mobile Computing, 6(4):401–423, 2010.

[PJZ+14] PERERA, CHARITH, PREM PRAKASH JAYARAMAN, ARKADY ZASLAVSKY, PETER CHRISTEN und DIMITRIOS GEORGAKOPOULOS: Mosden: An internet of things middleware for resource constrained mobile devices. In: *2014 47th Hawaii International Conference on System Sciences*, Seiten 1053–1062. IEEE, 2014.

[PM01] PITT, ESMOND und KATHY MCNIFF: *Java. rmi: The Remote*

Method Invocation Guide. Addison-Wesley Longman Publishing Co., Inc., 2001.

[Pow16] POWERTUTOR.ORG: Powertutor - a power monitor for android-based mobile platforms. http://ziyang.eecs.umich.edu/projects/powertutor/index.html/, 2016. aufgerufen am 31.12.2016.

[PP14] PASPALLIS, NEARCHOS und GEORGE A PAPADOPOULOS: A pluggable middleware architecture for developing context-aware mobile applications. Personal and Ubiquitous Computing, 18(5):1099–1116, 2014.

[PPRB06] PAUTY, JULIEN, DAVY PREUVENEERS, PETER RIGOLE und YOLANDE BERBERS: Research challenges in mobile and context-aware service development. In: *Future Research Challenges for Software and Services Conference*, Seiten 141–148. Citeseer, 2006.

[PRK09] PORRAS, JARI, ORIANA RIVA und MADS DARØ KRISTENSEN: Dynamic resource management and cyber foraging. In: *Middleware for Network Eccentric and Mobile Applications*, Seiten 349–368. Springer, 2009.

[PZCG14] PERERA, CHARITH, ARKADY ZASLAVSKY, PETER CHRISTEN und DIMITRIOS GEORGAKOPOULOS: Context aware computing for the internet of things: A survey. IEEE Communications Surveys & Tutorials, 16(1):414–454, 2014.

[RBD+09] ROUVOY, ROMAIN, PAOLO BARONE, YUN DING, FRANK ELIASSEN, SVEIN HALLSTEINSEN, JORGE LORENZO, ALESSANDRO MAMELLI und ULRICH SCHOLZ: Music: Middleware support for self-adaptation in ubiquitous and service-oriented environments. In: *Software engineering for self-adaptive systems*, Seiten 164–182. Springer, 2009.

[RCKZ13] RAYCHOUDHURY, VASKAR, JIANNONG CAO, MOHAN KUMAR und DAQIANG ZHANG: Middleware for pervasive computing: A survey. Pervasive and Mobile Computing, 9(2):177–200, 2013.

[RGO06] ROTEM-GAL-OZ, ARNON: Fallacies of distributed computing explained. http://www.rgoarchitects.com/Files/fallacies.pdf, 2006. aufgerufen am 31.12.2016.

[RHC+02] ROMÁN, MANUEL, CHRISTOPHER HESS, RENATO CERQUEI-

RA, ANAND RANGANATHAN, ROY H CAMPBELL und KLARA NAHRSTEDT: Gaia: a middleware platform for active spaces. ACM SIGMOBILE Mobile Computing and Communications Review, 6(4):65–67, 2002.

[Rob13] ROBINSON, STUART: Cellphone energy gap: Desperately seeking solutions. https://www.strategyanalytics.com/default.aspx?mod=reportabstractviewer&a0=4645, 2013. aufgerufen am 31.12.2013.

[Rot02] ROTH, JÖRG: *Mobile Computing*. dpunkt.verlag, 2002.

[RRA08] RELLERMEYER, JAN S, ORIANA RIVA und GUSTAVO ALONSO: Alfredo: an architecture for flexible interaction with electronic devices. In: *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, Seiten 22–41. Springer-Verlag New York, Inc., 2008.

[RSM+11] RA, MOO-RYONG, ANMOL SHETH, LILY MUMMERT, PADMANABHAN PILLAI, DAVID WETHERALL und RAMESH GOVINDAN: Odessa: enabling interactive perception applications on mobile devices. In: *Proceedings of the 9th international conference on Mobile systems, applications, and services*, Seiten 43–56. ACM, 2011.

[RSW+16] REHMAN, MUHAMMAD HABIB UR, CHEE SUN, TEH YING WAH, AHSAN IQBAL und PREM PRAKASH JAYARAMAN: Opportunistic computation offloading in mobile edge cloud computing environments. In: *Mobile Data Management (MDM), 2016 17th IEEE International Conference on*, Band 1, Seiten 208–213. IEEE, 2016.

[RVGH11] RIISER, HAAKON, PAUL VIGMOSTAD, CARSTEN GRIWODZ und PÅL HALVORSEN: Bitrate and video quality planning for mobile streaming scenarios using a gps-based bandwidth lookup service. In: *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, Seiten 1–6. IEEE, 2011.

[SA13] SIRIS, VASILIOS A und MARIA ANAGNOSTOPOULOU: Performance and energy efficiency of mobile data offloading with mobility prediction and prefetching. In: *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*, Seiten 1–6. IEEE, 2013.

[SAGB14] SANAEI, ZOHREH, SAEID ABOLFAZLI, ABDULLAH GANI und RAJKUMAR BUYYA: Heterogeneity in mobile cloud computing: taxonomy and open challenges. IEEE Communications Surveys & Tutorials, 16(1):369–392, 2014.

[SAH11] SCHUSTER, CHRISTOPHER, MALTE APPELTAUER und ROBERT HIRSCHFELD: Context-oriented programming for mobile devices: Jcop on android. In: *Proceedings of the 3rd International Workshop on Context-Oriented Programming*, Seite 5. ACM, 2011.

[Sat01] SATYANARAYANAN, MAHADEV: Pervasive computing: Vision and challenges. IEEE Personal communications, 8(4):10–17, 2001.

[Sat17a] SATYANARAYANAN, MAHADEV: Edge computing for situational awareness. In: *Local and Metropolitan Area Networks (LANMAN), 2017 IEEE International Symposium on*, Seiten 1–6. IEEE, 2017.

[Sat17b] SATYANARAYANAN, MAHADEV: The emergence of edge computing. Computer, 50(1):30–39, 2017.

[SAZN12] SHI, CONG, MOSTAFA H AMMAR, ELLEN W ZEGURA und MAYUR NAIK: Computing in cirrus clouds: the challenge of intermittent connectivity. In: *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, Seiten 23–28. ACM, 2012.

[SBCD09] SATYANARAYANAN, MAHADEV, PARAMVIR BAHL, RAMÓN CACERES und NIGEL DAVIES: The case for vm-based cloudlets in mobile computing. IEEE pervasive Computing, 8(4):14–23, 2009.

[SBI+15] SHOAIB, MUHAMMAD, STEPHAN BOSCH, OZLEM DURMAZ INCEL, HANS SCHOLTEN und PAUL JM HAVINGA: A survey of online activity recognition using mobile phones. Sensors, 15(1):2059–2085, 2015.

[Sch95] SCHILIT, WILLIAM NOAH: *A system architecture for context-aware mobile computing*. Doktorarbeit, Columbia University, 1995.

[Sch07] SCHIELE, GREGOR: *System support for spontaneous pervasive computing environments*. Doktorarbeit, Universität Stuttgart,

2007.

[SDA99] SALBER, DANIEL, ANIND K DEY und GREGORY D ABOWD: The context toolkit: aiding the development of context-enabled applications. In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, Seiten 434–441. ACM, 1999.

[SF05] SU, YA-YUNN und JASON FLINN: Slingshot: deploying stateful services in wireless hotspots. In: *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, Seiten 79–92. ACM, 2005.

[SG14] SHIRAZ, MUHAMMAD und ABDULLAH GANI: A lightweight active service migration framework for computational offloading in mobile cloud computing. The Journal of Supercomputing, 68(2):978–995, 2014.

[SGKB13] SHIRAZ, MUHAMMAD, ABDULLAH GANI, RASHID HAFEEZ KHOKHAR und RAJKUMAR BUYYA: A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing. IEEE Communications Surveys & Tutorials, 15(3):1294–1313, 2013.

[SGM02] SZYPERSKI, C., D. GRUNTZ und S. MURER: *Component Software: Beyond Object-Oriented Programming*. Component Software Series. Prentice Hall, 2002.

[SHD12] SHIN, CHOONSUNG, JIN-HYUK HONG und ANIND K DEY: Understanding and prediction of mobile application usage for smart phones. In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, Seiten 173–182. ACM, 2012.

[She00] SHEARER, COLIN: The crisp-dm model: the new blueprint for data mining. Journal of data warehousing, 5(4):13–22, 2000.

[Shi14] SHI, CONG: *Addressing connectivity challenges for mobile computing and communication*. Doktorarbeit, Georgia Institute of Technology, 2014.

[SHP+14] SHI, CONG, KARIM HABAK, PRANESH PANDURANGAN, MOSTAFA AMMAR, MAYUR NAIK und ELLEN ZEGURA: Cosmos: computation offloading as a service for mobile devices. In: *Proceedings of the 15th ACM international symposium on Mobi-*

le ad hoc networking and computing, Seiten 287–296. ACM, 2014.

[Sig08] SIGG, STEPHAN: *Development of a novel context prediction algorithm and analysis of context prediction schemes*. Kassel University Press GmbH, 2008.

[SKK+90] SATYANARAYANAN, MAHADEV, JAMES J. KISTLER, PUNEET KUMAR, MARIA E. OKASAKI, ELLEN H. SIEGEL und DAVID C. STEERE: Coda: A highly available file system for a distributed workstation environment. IEEE Transactions on computers, 39(4):447–459, 1990.

[SKK+14] SONG, WOOK, YESEONG KIM, HAKBONG KIM, JEHUN LIM und JIHONG KIM: Personalized optimization for android smartphones. ACM Transactions on Embedded Computing Systems (TECS), 13(2s):60, 2014.

[SLAZ12] SHI, CONG, VASILEIOS LAKAFOSIS, MOSTAFA H AMMAR und ELLEN W ZEGURA: Serendipity: enabling remote computing among intermittently connected mobile devices. In: *Proceedings of the thirteenth ACM international symposium on Mobile Ad Hoc Networking and Computing*, Seiten 145–154. ACM, 2012.

[SLP04] STRANG, THOMAS und CLAUDIA LINNHOFF-POPIEN: A context modeling survey. In: *Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004-The Sixth International Conference on Ubiquitous Computing, Nottingham/England*, 2004.

[SMF+12] SOYATA, TOLGA, RAJANI MURALEEDHARAN, COLIN FUNAI, MINSEOK KWON und WENDI HEINZELMAN: Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In: *Computers and Communications (ISCC), 2012 IEEE Symposium on*, Seiten 000059–000066. IEEE, 2012.

[SMM+11] SCELLATO, SALVATORE, MIRCO MUSOLESI, CECILIA MASCOLO, VITO LATORA und ANDREW T CAMPBELL: Nextplace: a spatio-temporal prediction framework for pervasive systems. In: *International Conference on Pervasive Computing*, Seiten 152–169. Springer, 2011.

[SMM15] SILVA, FRANCISCO AIRTON, PAULO MACIEL und RUBENS

MATOS: Smartrank: a smart scheduling tool for mobile cloud computing. The Journal of Supercomputing, 71(8):2985–3008, 2015.

[Spi15] SPIER, ALEXANDER: Das große akku-fressen. Heise Verlag c't, 22:87–89, 2015.

[SPN+13] SHI, CONG, PRANESH PANDURANGAN, KANGQI NI, JUYUAN YANG, MOSTAFA AMMAR, MAYUR NAIK und ELLEN ZEGURA: IC-Cloud: Computation offloading to an intermittently-connected cloud. Technischer Bericht GT-CS-13-01, Georgia Institute of Technology, 2013.

[SPUP02] SCHEIN, ANDREW I, ALEXANDRIN POPESCUL, LYLE H UNGAR und DAVID M PENNOCK: Methods and metrics for cold-start recommendations. In: *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, Seiten 253–260. ACM, 2002.

[SRT+11] SHEPARD, CLAYTON, AHMAD RAHMATI, CHAD TOSSELL, LIN ZHONG und PHILLIP KORTUM: Livelab: measuring wireless networks and smartphone users in the field. ACM SIGMETRICS Performance Evaluation Review, 38(3):15–20, 2011.

[SS02] SUN, JUN-ZHAO und JAAKKO SAUVOLA: On fundamental concept of mobility for mobile communications. In: *Personal, Indoor and Mobile Radio Communications, 2002. The 13th IEEE International Symposium on*, Band 2, Seiten 799–803. IEEE, 2002.

[SSRB13] SCHMIDT, DOUGLAS C, MICHAEL STAL, HANS ROHNERT und FRANK BUSCHMANN: *Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects*, Band 2. John Wiley & Sons, 2013.

[SSSL12] SMIT, MICHAEL, MARK SHTERN, BRADLEY SIMMONS und MARIN LITOIU: Partitioning applications for hybrid and federated clouds. In: *Proceedings of the 2012 Conference of the Center for Advanced Studies on Collaborative Research*, Seiten 27–41. IBM Corp., 2012.

[SSX+12] SAARINEN, AKI, MATTI SIEKKINEN, YU XIAO, JUKKA K NURMINEN, MATTI KEMPPAINEN und PAN HUI: Smartdiet: offloading popular apps to save energy. ACM SIGCOMM Computer Communication Review, 42(4):297–298, 2012.

[ST94] SCHILIT, BILL N und MARVIN M THEIMER: Disseminating active map information to mobile hosts. IEEE network, 8(5):22–32, 1994.

[Sta16] STATISTA GMBH: Global smartphone user penetration 2014-2019 | statistic. https://www.statista.com/statistics/203734/global-smartphone-penetration-per-capita\-since-2005/, 2016. aufgerufen am 31.12.2016.

[SVHVV15] SIMOENS, PIETER, LANDER VAN HERZEELE, FREDERIK VANDEPUTTE und LUC VERMOESEN: Challenges for orchestration and instance selection of composite services in distributed edge clouds. In: *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, Seiten 1196–1201. IEEE, 2015.

[SVL+06] SOHN, TIMOTHY, ALEX VARSHAVSKY, ANTHONY LAMARCA, MIKE Y CHEN, TANZEEM CHOUDHURY, IAN SMITH, SUNNY CONSOLVO, JEFFREY HIGHTOWER, WILLIAM G GRISWOLD und EYAL DE LARA: Mobility detection using everyday gsm traces. In: *International Conference on Ubiquitous Computing*, Seiten 212–224. Springer, 2006.

[SW97] STOER, MECHTHILD und FRANK WAGNER: A simple min-cut algorithm. Journal of the ACM (JACM), 44(4):585–591, 1997.

[TCMA12] TRAN, LE HUNG, MICHELE CATASTA, LUCAS KELSEY MCDOWELL und KARL ABERER: Next place prediction using mobile data. In: *Proceedings of the Mobile Data Challenge Workshop (MDC 2012)*, Nummer 182131 in *EPFL-CONF*, 2012.

[Tel16] TELEKOM DEUTSCHLAND GMBH: Coverage Checker. https://t-map.t-mobile.de/TMAP4/index/index.jsp?functionalArea=coverage_checker&region=de, 2016. aufgerufen am 31.12.2016.

[TLH+16] TEREFE, MATI B, HEEZIN LEE, NOJUNG HEO, GEOFFREY C FOX und SANGYOON OH: Energy-efficient multisite offloading policy using markov decision process for mobile cloud computing. Pervasive and Mobile Computing, 27:75–89, 2016.

[Tom16] TOMESCU, G.: How accurate is the iphone 5s cpu performance chart? http://gabrieltomescu.com/blog/2013/10/25/

how-accurate-is-the-iphone-5s-cpu-performance-chart, 2016. aufgerufen am 31.12.2016.

[TS12] TAIVALSAARI, ANTERO und KARI SYSTÄ: Cloudberry: An html5 cloud phone platform for mobile devices. IEEE software, 29(4):40–45, 2012.

[TUK77] TUKEY, JW: *EXPLORATORY DATA ANALYSIS*. Pearson, 1977.

[Tv08] TANENBAUM, ANDREW S. und MAARTEN VAN STEEN: *Verteilte Systeme*. it-informatik. Pearson Studium, 2 Auflage, 2008.

[VB14] VANSYCKEL, SEBASTIAN und CHRISTIAN BECKER: A survey of proactive pervasive computing. In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, Seiten 421–430. ACM, 2014.

[VCMB+07] VAN CUTSEM, TOM, STIJN MOSTINCKX, ELISA GONZALEZ BOIX, JESSIE DEDECKER und WOLFGANG DE MEUTER: Ambienttalk: object-oriented event-driven programming in mobile ad hoc networks. In: *Chilean Society of Computer Science, 2007. SCCC'07. XXVI International Conference of the*, Seiten 3–12. IEEE, 2007.

[VIP16] VERT, GREGORY, S SITHARAMA IYENGAR und VIR V PHOHA: *Introduction to Contextual Processing: Theory and Applications*. CRC Press, 2016.

[VNMW+05] VAN NIEUWPOORT, ROB V, JASON MAASSEN, GOSIA WRZESIŃSKA, RUTGER FH HOFMAN, CERIEL JH JACOBS, THILO KIELMANN und HENRI E BAL: Ibis: a flexible and efficient java-based grid programming environment. Concurrency and Computation: Practice and Experience, 17(7-8):1079–1107, 2005.

[VP08] VERVERIDIS, CHRISTOPHER N und GEORGE C POLYZOS: Service discovery for mobile ad hoc networks: a survey of issues and techniques. IEEE Communications Surveys & Tutorials, 10(3):30–45, 2008.

[VRC11] VALLINA-RODRIGUEZ, NARSEO und JON CROWCROFT: Erdos: achieving energy savings in mobile os. In: *Proceedings of*

the sixth international workshop on MobiArch, Seiten 37–42. ACM, 2011.

[VREXC11] VALLINA-RODRIGUEZ, NARSEO, CHRISTOS EFSTRATIOU, GEOFFREY XIE und JON CROWCROFT: Enabling opportunistic resources sharing on mobile operating systems: Benefits and challenges. In: *Proceedings of the 3rd ACM Workshop on Wireless of the Students, by the Students, for the Students*, Seiten 29–32. ACM, 2011.

[VRMCL08] VAQUERO, LUIS M, LUIS RODERO-MERINO, JUAN CACERES und MAIK LINDNER: A break in the clouds: towards a cloud definition. ACM SIGCOMM Computer Communication Review, 39(1):50–55, 2008.

[VSDTD12a] VERBELEN, TIM, PIETER SIMOENS, FILIP DE TURCK und BART DHOEDT: Aiolos: Middleware for improving mobile application performance through cyber foraging. Journal of Systems and Software, 85(11):2629–2639, 2012.

[VSDTD12b] VERBELEN, TIM, PIETER SIMOENS, FILIP DE TURCK und BART DHOEDT: Cloudlets: bringing the cloud to the mobile user. In: *Proceedings of the third ACM workshop on Mobile cloud computing and services*, Seiten 29–36. ACM, 2012.

[VSDTD13] VERBELEN, TIM, TIM STEVENS, FILIP DE TURCK und BART DHOEDT: Graph partitioning algorithms for optimizing software deployment in mobile cloud computing. Future Generation Computer Systems, 29(2):451–459, 2013.

[vSvHW+06] SINDEREN, MARTEN J VAN, AART T VAN HALTEREN, MAARTEN WEGDAM, HENDRIK B MEEUWISSEN und E HENK EERTINK: Supporting context-aware mobile applications: an infrastructure approach. IEEE Communications Magazine, 44(9):96–104, 2006.

[Wal99] WALDO, JIM: The jini architecture for network-centric computing. Communications of the ACM, 42(7):76–82, 1999.

[WC13] WEI, EDWIN JY und ALVIN TS CHAN: Campus: A middleware for automated context-aware adaptation decision making at run time. Pervasive and Mobile Computing, 9(1):35–56, 2013.

[WD12] WANG, SHAOXUAN und SUJIT DEY: Cloud mobile gaming: modeling and measuring user experience in mobile wireless net-

works. ACM SIGMOBILE Mobile Computing and Communications Review, 16(1):10–21, 2012.

[Wei91] WEISER, MARK: The computer for the 21st century. Scientific american, 265(3):94–104, 1991.

[Wei93] WEISER, MARK: *Some computer science issues in ubiquitous computing*, Band 36.7. ACM, 1993.

[WHB13] WU, HUIJUN, DIJIANG HUANG und SAMIA BOUZEFRANE: Making offloading decisions resistant to network unavailability for mobile cloud collaboration. In: *Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference Conference on*, Seiten 168–177. IEEE, 2013.

[Wik13] WIKIPEDIA: Samsung galaxy s4. `https://en.wikipedia.org/w/index.php?title=Samsung_Galaxy_S4&oldid=588346596`, 2013. aufgerufen am 31.12.2016.

[Wik16] WIKIPEDIA: Viewar butlers screenshot - erweiterte realität. `https://upload.wikimedia.org/wikipedia/commons/b/b0/ViewAR_BUTLERS_Screenshot.jpg`, 2016. aufgerufen am 31.12.2016.

[WKWS16] WU, HUAMING, WILLIAM KNOTTENBELT, KATINKA WOLTER und YI SUN: An optimal offloading partitioning algorithm in mobile cloud computing. In: *International Conference on Quantitative Evaluation of Systems*, Seiten 311–328. Springer, 2016.

[WL04] WANG, CHENG und ZHIYUAN LI: Parametric analysis for adaptive computation offloading. In: *ACM SIGPLAN Notices*, Band 39.6, Seiten 119–130. ACM, 2004.

[WLHL13] WANG, XUDONG, XUANZHE LIU, GANG HUANG und YUNXIN LIU: Appmobicloud: improving mobile web applications by mobile-cloud convergence. In: *Proceedings of the 5th Asia-Pacific Symposium on Internetware*, Seite 14. ACM, 2013.

[Woo09] WOOLDRIDGE, MICHAEL: *An introduction to multiagent systems*. John Wiley & Sons, 2009.

[WSW13] WANG, XIAO SOPHIA, HAICHEN SHEN und DAVID WETHERALL: Accelerating the mobile web with selective offloading. In:

Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing, Seiten 45–50. ACM, 2013.

[WZL16] WANG, SHIQIANG, MURTAZA ZAFER und KIN K LEUNG: Online placement of multi-component applications in edge computing environments. In: *arXiv preprint arXiv:1605.08023*, 2016.

[XDL+14] XIA, FENG, FANGWEI DING, JIE LI, XIANGJIE KONG, LAURENCE T YANG und JIANHUA MA: Phone2cloud: Exploiting computation offloading for energy saving on smartphones in mobile cloud computing. Information Systems Frontiers, 16(1):95–111, 2014.

[XLC15] XIANG, XUDONG, CHUANG LIN und XIN CHEN: Ecoplan: energy-efficient downlink and uplink data transmission in mobile cloud computing. Wireless Networks, 21(2):453–466, 2015.

[XP12] XUE, WENWEI und HUNG KENG PUNG: Context-aware middleware for supporting mobile applications and services. In: *Handbook of Mobile Systems Applications and Services*, Seiten 269–304. CRC Press, 2012.

[YCWL06] YU, PING, JIANNONG CAO, WEIDONG WEN und JIAN LU: Mobile agent enabled application mobility for pervasive computing. In: *International Conference on Ubiquitous Intelligence and Computing*, Seiten 648–657. Springer, 2006.

[YCY+13] YANG, LEI, JIANNONG CAO, YIN YUAN, TAO LI, ANDY HAN und ALVIN CHAN: A framework for partitioning and execution of data stream applications in mobile cloud computing. ACM SIGMETRICS Performance Evaluation Review, 40(4):23–32, 2013.

[YHE14] YUAN, BINGCHUAN, JOHN HERBERT und YALDA EMAMIAN: Smartphone-based activity recognition using hybrid classifier. In: *In proceeding of the 4th International Conference on Pervasive and Embedded Computing and Communication Systems*, Seiten 14–23, 2014.

[YHK+12] YAP, KOK-KIONG, TE-YUAN HUANG, MASAYOSHI KOBAYASHI, YIANNIS YIAKOUMIS, NICK MCKEOWN, SACHIN KATTI und GURU PARULKAR: Making use of all the networks around us: a case study in android. In: *Proceedings of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design*, Seiten 19–24. ACM, 2012.

[YKC+13] YANG, SEUNGJUN, YONGIN KWON, YEONGPIL CHO, HAYOON YI, DONGHYUN KWON, JONGHEE YOUN und YUNHEUNG PAEK: Fast dynamic execution offloading for efficient mobile cloud computing. In: *Pervasive Computing and Communications (PerCom), 2013 IEEE International Conference on*, Seiten 20–28. IEEE, 2013.

[YLM14] YURUR, OZGUR, CHI LIU und WILFRIDO MORENO: A survey of context-aware middleware designs for human activity recognition. IEEE Communications Magazine, 52(6):24–31, 2014.

[YOC08] YANG, KUN, SHUMAO OU und HSIAO-HWA CHEN: On effective offloading services for resource-constrained mobile devices running heavier mobile internet applications. IEEE communications magazine, 46(1):56–63, 2008.

[YYJZ13] YAO, DEZHONG, CHEN YU, HAI JIN und JIEHAN ZHOU: Energy efficient task scheduling in mobile cloud computing. In: *IFIP International Conference on Network and Parallel Computing*, Seiten 344–355. Springer, 2013.

[Zap12] ZAPLATA, SONJA: *Flexibilisierung verteilter Prozessausführung: Zur dynamischen Verteilung, Überwachung und Steuerung individueller Prozessinstanzen auf Anwendungsebene*. Dissertation, Universität Hamburg, Fachbereich Informatik, Vogt-Kölln-Str. 30, 22527 Hamburg, Germany, 10 2012.

[ZKJG11] ZHANG, XINWEN, ANUGEETHA KUNJITHAPATHAM, SANGOH JEONG und SIMON GIBBS: Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing. Mobile Networks and Applications, 16(3):270–284, 2011.

[ZLJF12] ZHANG, YUAN, HAO LIU, LEI JIAO und XIAOMING FU: To offload or not to offload: an efficient code partition algorithm for mobile cloud computing. In: *Cloud Networking (CLOUDNET), 2012 IEEE 1st International Conference on*, Seiten 80–86. IEEE, 2012.

[ZN10] ZHENG, PEI und LIONEL NI: *Smart phone and next generation mobile computing*. Morgan Kaufmann, 2010.

[ZSG+09] ZHANG, XINWEN, JOSHUA SCHIFFMAN, SIMON GIBBS, ANUGEETHA KUNJITHAPATHAM und SANGOH JEONG: Securing elastic applications on mobile devices for cloud computing. In:

Proceedings of the 2009 ACM workshop on Cloud computing security, Seiten 127–134. ACM, 2009.

[ZTC14] ZHAO, BO, BYUNG CHUL TAK und GUOHONG CAO: *Mobile web browsing using the cloud*. Springer, 2014.

[ZTD+10] ZHANG, LIDE, BIRJODH TIWANA, ROBERT P DICK, ZHIYUN QIAN, Z MORLEY MAO, ZHAOGUANG WANG und LEI YANG: Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In: *Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2010 IEEE/ACM/IFIP International Conference on*, Seiten 105–114. IEEE, 2010.

[ZWL02] ZHU, WENZHANG, CHO-LI WANG und FRANCIS CM LAU: Jessica2: A distributed java virtual machine with transparent thread migration support. In: *Cluster Computing, 2002. Proceedings. 2002 IEEE International Conference on*, Seiten 381–388. IEEE, 2002.

[ZXC+10] ZHAO, BO, ZHI XU, CAIXIA CHI, SENCUN ZHU und GUOHONG CAO: Mirroring smartphones for good: A feasibility study. In: *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, Seiten 26–38. Springer, 2010.

[ZZLY13] ZHU, YIN, ERHENG ZHONG, ZHONGQI LU und QIANG YANG: Feature engineering for semantic place prediction. Pervasive and mobile computing, 9(6):772–783, 2013.

Eidesstattliche Versicherung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Hamburg, den 6. Dezember 2017

Gabriel Orsini