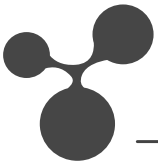
Technische Universität Dresden – Fakultät Informatik
Professur für Multimediatechnik, Privat-Dozentur für Angewandte Informatik

Prof. Dr.-Ing. Klaus Meißner
PD Dr.-Ing. habil. Martin Engelien
(Hrsg.)



GeNeMe '11

Gemeinschaften in Neuen Medien

an der
Fakultät Informatik der Technischen Universität Dresden

mit Unterstützung der

3m5. Media GmbH, Dresden
Communardo Software GmbH, Dresden
GI-Regionalgruppe, Dresden
FERCHAU Engineering GmbH, Dresden
IBM, Dresden
itsax.de | pludoni GmbH, Dresden
Kontext E GmbH, Dresden
objectFab GmbH, Dresden
queo GmbH, Dresden
Robotron Datenbank-Software GmbH, Dresden
SALT Solutions GmbH, Dresden
SAP AG, Resarch Center Dresden
Saxonia Systems AG, Dresden
T-Systems Multimedia Solutions GmbH, Dresden
Transinsight GmbH, Dresden
xima media GmbH, Dresden

am 07. und 08. September 2011 in Dresden

www.geneme.de
info@geneme.de

E.2 Referenzarchitektur eines Frameworks für die Entwicklung kompositer, ubiquitärer Anwendungen

Oliver Mroß [1], Thomas Schlegel [2]
Technische Universität Dresden, Institut für Software und Multimediatechnik
[1] Lehrstuhl für Multimediatechnik
[2] Juniorprofessur für Software Engineering ubiquitärer Systeme

1 Zusammenfassung

Durch die zunehmende Anzahl mobiler Endgeräte wie Smartphones sowie der Digitalisierung des öffentlichen Raums stehen ubiquitäre Umgebungen verstärkt im Fokus der Forschung. Werden die Endgeräte und zugehörigen Dienste einer solchen Umgebung zu einer kompositen, ubiquitären Anwendung (kubi App) zusammengefasst, so kann der Anwender auf diese über eine komposite Benutzerschnittstelle zugreifen, z. B. von seinem persönlichen Smartphone aus. Aus der Übertragung des Mashup-Entwicklungsansatzes auf ubiquitäre Umgebungen ergeben sich Implikationen, die im vorliegenden Beitrag untersucht werden sollen. Anhand der Untersuchung verwandter Forschungsarbeiten lassen sich wichtige funktionale Anforderungen aus der Perspektive der ubiquitären Umgebung und der Endgeräte formulieren, die im Rahmen dieses Beitrages vorgestellt werden. Anhand der Anforderungen wird eine Referenzarchitektur zu einem Entwicklungsframework für kubi Apps vorgestellt, die als Grundlage zur Entwicklung zukünftiger Laufzeitumgebungen dienen soll.

2 Einführung

Der Anwender kann heutzutage mit Hilfe mobiler Endgeräte, z. B. Smartphones, von überall auf die Dienste und Informationen des mobilen Internets zugreifen. Sie besitzen neben der Telefonie zusätzliche Kommunikationsfunktionen (Email, VoIP, Videokommunikation, etc.) und ermöglichen somit eine Form der ortsunabhängigen Kommunikation. Neben der zunehmenden Anzahl mobiler Endgeräte ist der Trend zur Digitalisierung des öffentlichen Raums erkennbar. Hierbei handelt es sich um die Bereitstellung von öffentlich zugänglichen ubiquitären Diensten über interaktive Geräte wie *Public Displays*, Kiosk- oder Sensorsysteme, z. B. das *Touch & Travel*-Pilotprojekt[1] der Deutschen Bahn. Durch den Trend wird deutlich, dass sich im Anwendungsbereich der mobilen Endgeräte neue Möglichkeiten ergeben. Dies soll durch das folgende Anwendungsszenario beispielhaft verdeutlicht werden.
Lisa und Daniel möchten sich zu einem gemeinsamen Ausflug am Bahnhof treffen. Wo sie sich treffen, entscheiden beide ad-hoc über ihr Smartphone. Beim Betreten

1 http://www.touchandtravel.de/site/touchandtravel/de/start.html

des Bahnhofs erhält Daniel eine Übersicht zu den möglichen Tätigkeiten, die er innerhalb des Bahnhofs ausführen kann. Eine dieser Tätigkeiten ist das Auffinden von Freunden im Bahnhof. Nachdem Daniel die Tätigkeit ausgewählt hat, wird auf seinem Smartphone eine mobile Anwendung erzeugt, die sich aus mehreren UI-Komponenten zusammensetzt, z. B. eine Kartenkomponente und eine Kontaktliste. Aus der Kontaktliste wählt Daniel Lisa als Kommunikationspartner aus und versendet eine Einladung zur gemeinsamen Interaktion. Lisa erhält die Einladungsnachricht über ihr Smartphone und bestätigt sie. Damit Lisa weiß, wo Daniel sich im Moment aufhält, wählt er die Kartenkomponente seiner mobilen Anwendung aus und sendet ein Replikat an Lisas Smartphone. Dort empfangen erscheint die Komponente im Display und wird im selben Interaktionszustand dargestellt wie vor der Übertragung (Beibehaltung des Kartenausschnitts und aller Ortsmarken, einschließlich Daniels Position). Verändert Daniel den Interaktionszustand seiner Kartenkomponente, z. B. die Position, kann Lisa die Veränderungen mit Hilfe der übertragenen Komponente nach dem „*What You See Is What I See*"-Prinzip verfolgen. Da die Synchronisation bidirektional erfolgt, kann auch Daniel Lisas Position bestimmen. Über die Kartenkomponente können nun beide eine Route berechnen und durch sie zueinander finden.

Das Anwendungsszenario verdeutlicht, dass mobile Endgeräte in Zukunft als Kollaborationswerkzeuge angesehen und insbesondere in ubiquitären Umgebungen als solche genutzt werden können. Zur Unterstützung der Interaktion zwischen lokal voneinander getrennten Personen werden die UI-Komponenten im Anwendungsszenario zwischen den verschiedenen Endgeräten übertragen, wobei der Interaktionszustand der Komponenten erhalten bleibt. Der Vorgang wird als UI-Migration bezeichnet und dient im Rahmen dieses Beitrages primär zum Zweck der synchronen Kollaboration. Da ubiquitäre Umgebungen heterogener Natur sind, ist es notwendig, dass zukünftige mobile Anwendungen sich an die Gegebenheit der Umgebung anpassen und die zur Verfügung stehenden Ressourcen (Dienste und Geräte) auf geeignetem Weg dem Anwender zur Verfügung stellen können. Eine Möglichkeit hierzu stellen komposite Anwendungen dar, welche die verschiedenen domänenspezifischen *Web Services* und gerätespezifischen Dienste der ubiquitären Umgebung in eine einheitliche Benutzerschnittstelle integrieren können. Im Rahmen dieses Beitrages soll dieser neuartige Anwendungstyp als komposite, ubiquitäre Anwendung (kurz: *kubi App*) bezeichnet werden. Als Grundlage zur Erforschung der *kubi Apps* dient eine Referenzarchitektur, die in diesem Beitrag vorgestellt wird. Im folgenden Abschnitt 3 werden zunächst thematisch verwandte Arbeiten präsentiert und im Anschluss daran werden die Anforderungen im Bereich der ubiquitären Umgebung (Abschnitt 4.1) und im Bereich der Endgeräte (Abschnitt 4.2) erläutert. In Abschnitt 4.3 wird die Referenzarchitektur eines Frameworks zur Entwicklung kompositer, ubiquitärer Anwendungen beschrieben und zum Abschluss werden in Abschnitt 5 Schlussfolgerungen zusammengefasst und ein Ausblick ermöglicht.

3 Stand der Forschung

In [1] wird das Konzept des *Meta-User Interfaces* erläutert, mit deren Hilfe der Anwender einen Überblick und die Kontrolle über die verschiedenen Geräte und deren Ein- und Ausgabemöglichkeiten innerhalb der ubiquitären Umgebung erhält. Praktische Umsetzungen des Konzepts werden in den Ansätzen nach [2,3] untersucht. Um die verschiedenen Dienste der ubiquitären Umgebung strukturiert einsetzen zu können, werden in [4] Workflow-Templates eingesetzt. Hierbei handelt es sich um einen Ansatz zur dynamischen Servicekomposition, in dem mit Hilfe von Stellvertreterelementen (*Proxy*-Elemente) Dienste zum Instanziierungszeitpunkt in die Komposition aus der Menge der aktuell verfügbaren Services eingebunden werden. Der Nachteil des Ansatzes im Hinblick auf die Dynamik ubiquitärer Umgebungen ist die fehlende Anpassungsfähigkeit an die Änderungen im Kontext der ubiquitären Umgebung, z. B. das Ersetzen eines ausgefallenen Dienstes zur Laufzeit. In [5] werden verschiedene Ansätze zur dynamischen Servicekomposition verglichen. Ein Ergebnis der Analyse ist die Erkenntnis, dass vollständig dezentrale Architekturen für die dynamische Servicekomposition praktisch nicht geeignet sind. Die grundlegende Idee hinter dem CRUISe-Ansatz [6] ist die Überführung des Paradigmas der Service-orientierten Architektur auf die Präsentationsebene. Ein Service und dessen Benutzerschnittstelle bilden eine abgekapselte Einheit, die als UI-Service (UIS) bezeichnet wird. Die UIS können als Bestandteile komplexer Web-Anwendungen integriert werden – sogenannte *Mashups*. Veränderungen in der Komposition des *Mashups* zur Laufzeit, wie sie in ubiquitären Umgebungen auftreten können, z. B. durch neu verfügbare oder verschwindende Dienste, werden im CRUISe-Ansatz nicht berücksichtigt. Auch kollaborative Aspekte sind in CRUISe nicht Gegenstand der Forschung. Eine Möglichkeit zur Kooperation in ubiquitären Umgebungen stellt das Konzept der UI-Migration [7,8] dar. Hierbei wird die Benutzerschnittstelle eines Dienstes von einem Ausgangs- auf ein Zielgerät transferiert und der Interaktionszustand des UIs bleibt während des Transfers erhalten bzw. kann zwischen verteilten UI-Komponenten synchronisiert werden. Die UI-Migration im Ansatz nach [8] wird auf Basis clientseitiger Web-Technologien (HTML, CSS und JavaScript) durchgeführt. Sie bieten auf der einen Seite den Vorteil, dass die Benutzerschnittstelle geräte- und plattformunabhängig beschrieben wird, was wiederum vorteilhaft für die Anpassung des UIs an die spezifischen Bedingungen des Endgerätes ist. Auf der anderen Seite gehören Web-Browser bereits zur Standardsoftware auf Geräten wie Smartphones oder Tablets. Somit kann das Web als universale Plattform zur gemeinsamen Kollaboration eingesetzt werden. Ein Beispiel dafür ist der CoCAB-Ansatz [9], in dem mehrere Anwender synchron gemeinsam über die Grenzen heterogener Geräte hinweg eine Web-Seite betrachten und interagieren können. Der durch das CoCAB-System erbrachte Dienst kann bspw. als UIS nach dem Vorbild von CRUISe im Rahmen einer kompositen Web-Anwendung eingebettet werden, so dass verteilte Anwender kollaborativ über die Grenzen unterschiedlicher Endgeräte in ubiquitären Umgebungen zusammenarbeiten können.

4 Referenzarchitektur des Frameworks

Ausgehend von den Erkenntnissen der zuvor zitierten Arbeiten sollen nun die Anforderungen und die Referenzarchitektur zum Entwicklungsframework kompositer, ubiquitärer Anwendungen im Überblick vorgestellt werden. Da dezentrale Architekturen nicht möglich sind, muss zur Ausführung kompositer, ubiquitärer Anwendungen zwischen dem Kontext der ubiquitären Umgebung und dem lokalen Kontext eines Endgerätes unterschieden werden. In den folgenden Abschnitten werden die Anforderungen daher aus der Perspektive der ubiquitären Umgebung und aus der Endgeräteperspektive betrachtet. Hierbei werden aus Platzgründen nur die wichtigsten Anforderungen formuliert.

4.1 Funktionale Anforderungen an die intelligente Umgebung

Besonders mobile Endgeräte und zugehörige Dienste können die ubiquitäre Umgebung zu einem beliebigen Zeitpunkt betreten und auch wieder verlassen. Aus diesem Grund muss die ubiquitäre Umgebung eine Funktion bereitstellen, mit deren Hilfe die variabel verfügbaren Endgeräte und Dienste zur Laufzeit erkannt werden können (**Geräte- und Diensterkennung/Skalierbarkeit**). Um sie im Rahmen einer kompositen, ubiquitären Anwendung bestimmten Workflow-Teilschritten zuordnen zu können, müssen sie an zentraler Stelle, z. B. einer *Service Registry*-Komponente bereitgestellt werden (**Dienstbereitstellung**). Auf der anderen Seite muss die Umgebung mit Situationen umgehen können, in denen die Dienste aus dem Kontext der ubiquitären Umgebung verschwinden (**Ausfallbehandlung**). Da innerhalb der ubiquitären Umgebung Kollaboration zwischen verteilten Anwendern möglich sein soll, muss ein **Kollaborationsdienst** die UI-Migration zwischen den Endgeräten zum Zweck der Zusammenarbeit ermöglichen. Während der UI-Migration sollen sich die UIS durch interne und externe Mechanismen an die Bedingungen des Zielgerätes anpassen können (**Adaptionsunterstützung**).

4.2 Funktionale Anforderung an die Endgerätelaufzeitumgebung

Die Laufzeitumgebung muss die Ressourcen eines Endgerätes und dessen Eigenschaften erkennen und klassifizieren können (**Ressourcenerkennung**), z. B. Ein- und Ausgabegeräte. Die Funktionen des Endgerätes, z. B. Lagebestimmung im Raum oder komplexe Anwendungslogik (*Apps*) sollen nach außen als Service sichtbar sein. Sie sollen durch semantische Meta-Informationen charakterisiert werden, z. B. Geräte- und Informationstypen (**Servicesichtbarkeit**). Mit Hilfe der erweiterten Dienstebeschreibungen sollen aus der Perspektive der intelligenten Umgebung dynamische Servicekompositionen möglich sein. Die Dienste, welche Eingaben vom Anwender benötigen, stellen ein vordefiniertes UI in einem geräte- und plattformübergreifenden Format bereit, z.B. HTML/CSS (**Servicezugriff**). Die lokale Laufzeitumgebung des Endgerätes besitzt eine Komponente, welche

die UI-Beschreibungen der verschiedenen gerätespezifischen Dienste darstellen kann (**Servicedarstellung**). Damit der aktuelle Zustand eines jeden Endgerätes zu einem beliebigen Zeitpunkt bestimmt werden kann, muss die Laufzeitumgebung Informationen zu den QoS-Parameter zur Laufzeit erfassen und nach außen zur Verfügung stellen können. Will der Anwender von seinem Endgerät auf die angebotenen Dienste zugreifen, so müssen die UIC während der Laufzeit in den Kontext des Endgerätes integriert werden können (**Serviceintegration**). Dies geschieht ebenfalls während der UI-Migration auf Seiten des Empfangsgerätes. Hier muss die Laufzeitumgebung die empfangene UI-Komponente integrieren und in den vorherigen Interaktionszustand versetzen können. Zur Übertragung des Interaktionszustandes einer beliebigen UIC muss die Laufzeitumgebung diesen für alle UIC kontinuierlich erfassen können (**Erfassung des Interaktionszustands zur Laufzeit**). Treten Fehler während der UI-Migration oder während des Zugriffs auf externe Dienste auf, so muss die lokale Laufzeitumgebung diese unter Einsatz geeigneter Mittel behandeln können (**Fehlerbehandlung**). Die UIS sollen mit anderen entfernten UIS sicher kommunizieren können, da in ubiquitären Umgebungen Daten kabellos übertragen werden und somit für Angreifer zugänglich sind (**Sichere, asynchrone Kommunikation**).

4.3 Referenzarchitektur im Überblick

In Abbildung 1 wird die Referenzarchitektur eines Frameworks zur Entwicklung kompositer, ubiquitärer Anwendungen dargestellt. Wie bereits erläutert, sind vollständig dezentrale Architekturen zur Entwicklung kompositer, ubiquitärer Anwendungen in der Praxis nicht geeignet. Das heißt, ein Ansatz, in dem alle Anforderungen aus Abschnitt 4.1 und 4.2 ausschließlich durch die Endgeräte (z.B. durch ein Smartphone) erfüllt werden, ist praktisch nicht möglich. Aus diesem Grund wird in Abbildung 1 in den Kontext der ubiquitären Umgebung und in den des Endgerätes unterschieden. Die dargestellten Komponenten der Kontexte werden in den nachfolgenden Abschnitten benannt und erläutert.

Komponenten der ubiquitären Umgebung

Die ***Device Service Discovery***-Komponente ist für die Erkennung der einzelnen Geräte und deren Dienste innerhalb der ubiquitären Umgebung verantwortlich. Sie erhält die Geräte- und die Dienstinformationen vom *Device Information Manager* des Endgerätes. Die erkannten gerätespezifischen Dienste werden an das ***Ubiquitous Service Repository*** übergeben, wo sie zur weiteren Nutzung bereitgestellt werden. Der ***Collaboration Service*** soll die Zusammenarbeit zwischen den verschiedenen Anwendern in Multi-Device-Szenarien ermöglichen. Dies bedeutet, er ist auch für die Migration der UIC, der Verteilung von Einladungen an Teilnehmer, der Übertragung von Nachrichten und der Synchronisation der Daten sowie der Zustände innerhalb der

gemeinsamen Interaktion verantwortlich. Die Überwachung und das Erkennen von Veränderungen im Kontext der ubiquitären Umgebung sind die Aufgaben der ***Context Management***-Komponente. Sie erfasst modellhaft den Zustand der ubiquitären Umgebung. Zum Beispiel enthält sie Informationen darüber welche Geräte und Dienste momentan innerhalb der Umgebung verfügbar sind. Dazu setzt sie die ***Device Service Discovery***- und die ***Ubiquitous Service Repository***-Komponente ein. Die letztere dient als zentrale *Service Registry* nach dem Vorbild des SOA-Paradigmas, über welche die verschiedenen Dienste der ubiquitären Umgebung aufgerufen werden können. Durch den Einsatz von Ontologien können bei Dienstausfällen alternative Dienste bestimmt werden, indem *Reasoning*-Methoden eingesetzt werden (*Context Management*).

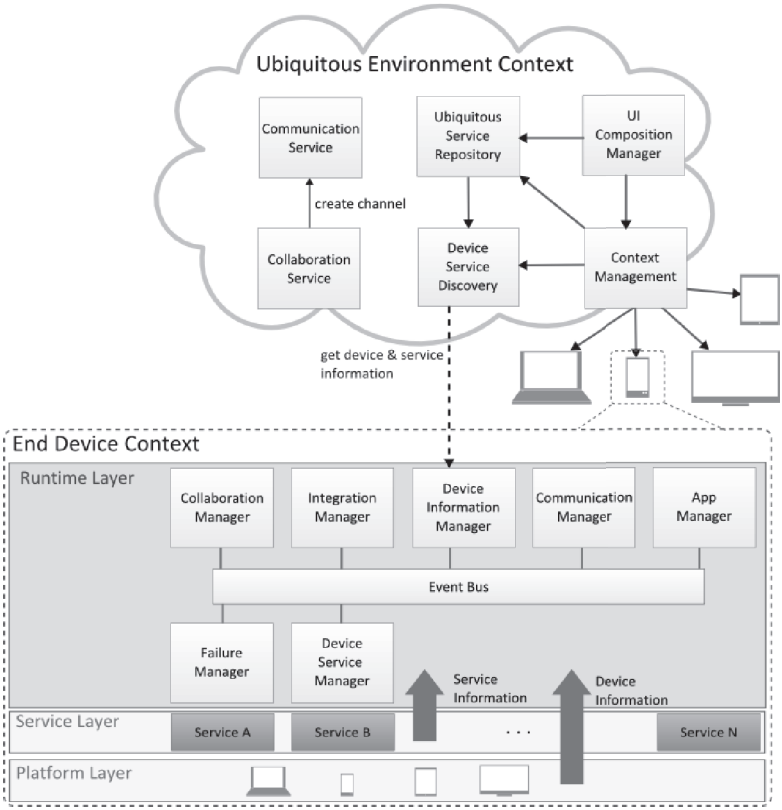


Abbildung 1: Überblick zur Referenzarchitektur

Der ***UI Composition Manager*** hat die Aufgabe, anhand der Kompositionsbeschreibung einer Anwendung (auch Workflows) und der Menge der momentan verfügbaren Dienste, für die Erzeugung und den Erhalt der kompositen, ubiquitären Anwendung zur Laufzeit zu sorgen. Fällt ein Dienst aus, so muss die Komponente in Wechselbeziehung mit dem *Context Management* die Dienstalternativen bestimmen und zur Laufzeit die Komposition anpassen (Ersetzen des ausgefallenen Dienstes). Dies kann zur Folge haben, dass ein anderes Gerät in die Komposition einbezogen werden muss. Hierbei sind zeitliche, sowie gerätetechnische Bedingungen (Aufwachen aus dem Standby u. ä.) durch die Komponente zu berücksichtigen. Ist eine Komposition zur Laufzeit nicht möglich, so muss dies ausgehend vom *UI Composition Manager* an das Gerät propagiert werden, auf dem die komposite, ubiquitäre Anwendung ausgeführt wird.

Komponenten der Endgerätelaufzeitumgebung

Ist ein Dienst in der ubiquiären Umgebung ausgefallen, so erhält der **App Manager** eine Nachricht und sorgt dafür, dass die UIC des ausgefallenen Dienstes entfernt und die UIC der möglichen Dienstalternative in die Anwendung integriert wird. Damit innerhalb der *Context Management*-Komponente das Modell zur ubiquitären Umgebung erstellt werden kann, benötigt sie Informationen zu jedem Endgerät sowie dessen Dienste. Die lokalen Kontextinformationen des Endgerätes werden durch den ***Device Information Manager*** gesammelt. Zudem ist sie für die Erfassung des Endgeräte- sowie Dienstausführungszustand zur Laufzeit verantwortlich. Da sich die Geräteinformationen, z. B. Typ der Sensoren, nicht zur Laufzeit verändern, können sie während der Registrierung des Gerätes innerhalb der ubiquitären Umgebung übertragen werden. Andere Informationen wie bspw. die Batterielebenszeit eines Smartphones müssen kontinuierlich in das Modell des interaktiven Raums übertragen werden. Der ***Collaboration Manager*** ist auf Seiten des Endgerätes dafür zuständig, die UICs im aktuellen Interaktionszustand an andere Anwender und dessen Endgeräte auf Wunsch des Benutzers zu migrieren und falls notwendig Replikate einer UI-Komponente anzufertigen, damit sie an mehrere Mitarbeiter verteilt werden können. Die Behandlung auftretender Fehler während der UI-Migration, des Zugriffs auf entfernte Dienste, der Initialisierung und Ausführung von UICs innerhalb einer kompositen, ubiquitären Anwendung soll durch den ***Failure Manager*** übernommen werden. Das Auftreten von Fehlern während der Ausführung eines gerätespezifischen Dienstes muss innerhalb der ubiquitären Umgebung an alle in der Komposition befindlichen Laufzeitumgebungen propagiert werden. Der *Failure Manager* soll hierzu ein Ereignis-Objekt erzeugen, welches an die *Context Management*-Komponente übertragen wird. Somit kann der Benutzer in Multi-Device-Szenarien erkennen, auf welchem Endgerät ein Fehler aufgetreten ist. Wird ein entfernter Dienst der ubiquitären Umgebung im Kontext der lokalen Laufzeitumgebung in die komposite, ubiquitäre Anwendung einbezogen, so erfolgt dies auf der Ebene des UIs nach dem Vorbild der CRUISe-Runtime. Der Vorgang der Einbettung des Dienstes

liegt im Verantwortlichkeitsbereich der ***Integration Manager***-Komponente. Eine weitere Funktion, die durch die Komponente ausgeführt wird, ist die Integration migrierender UI-Komponenten und die Benachrichtigung des *Collaboration Manager* zur Datensynchronisation im Rahmen der verteilten Interaktion. Die ***Device Service Manager***-Komponente hat auf der einen Seite die Funktion, die gerätespezifischen Dienstinformationen nach außen sichtbar zu machen und auf der anderen Seite die Funktion, die bereitgestellten Dienste mit den Dienstanfragenden (*Service Requester*) zu verbinden. Da nicht jeder Anwender jedem Zugriff auf sein persönliches Endgerät ermöglichen möchte, kann er über diese Komponente bestimmen, welche Dienste des Endgerätes nach außen verborgen bleiben und welche nicht.

5 Fazit

Die Implikationen, die sich aus der in Abschnitt 2 skizzierten *kubi App* ergeben, sind vielfältig und werden nachfolgend beschrieben. Da besonders mobile Endgeräte eingeschränkte Ressourcen besitzen, müssen Funktionen wie bspw. die Erkennung der verteilten Geräte und Dienste in ein externes System ausgelagert werden. Dies wird in der vorgestellten Referenzarchitektur berücksichtigt (siehe Abbildung 1) und es wird zwischen dem Kontext der ubiquitären Umgebung und dem Endgerätekontext unterschieden. Die ubiquitäre Umgebung besitzt Funktionen die in dem Beitrag in Abschnitt 4.3 erläutert wurden. Sie bilden die Grundlage zur Ausführung der *kubi Apps*. Um eine *kubi App* aus der Anwenderperspektive, bspw. auf dem Smartphone, ausführen zu können, ist eine Laufzeitumgebung notwendig, deren Funktionen ebenfalls in Abschnitt 4.3 erläutert wurden. Sie fasst die Benutzerschnittstellen der ubiquitären Dienste entsprechend der Komposition in einer Gesamtanwendung zusammen, die bspw. auf dem Smartphone ausgeführt wird. In der vorgestellten Referenzarchitektur werden, im Vergleich zum CRUISe-Ansatz, die dynamischen Veränderungen in der Servicekomposition zur Laufzeit und kollaborative Aspekte berücksichtigt. In ubiquitären Umgebungen entsteht aufgrund der Endgerätemobilität das Problem der variabel verfügbaren Dienste, wodurch sich eine *kubi App* an neu verfügbare oder fehlende Dienste anpassen muss. Obwohl CRUISe die Dienstvariabilität bisher nicht berücksichtigt, kann es aufgrund bereits existierender Dienste, bspw. die Kontextverwaltung, als Grundlage zur Entwicklung einer serverseitigen Middleware nach dem Vorbild des Meta-UI-Ansatzes dienen. Des Weiteren wurde in CRUISe eine clientseitige Laufzeitumgebung konzipiert, in der die Anforderung aus Abschnitt 4.2 noch nicht berücksichtigt werden. Auch in diesem Punkt können die Ergebnisse von CRUISe als Grundlage für Erweiterungen verwendet werden. Ein weiteres Problem stellt der Kollaborationsaspekt dar. In CRUISe wird dieser bisher nicht betrachtet. Hier besteht die Möglichkeit, die Ergebnisse des CoCAB-Ansatz in CRUISe als zusätzlichen Dienst einzubinden und durch das Konzept der UI-Migration zu erweitern.

6 Danksagungen

Der Beitrag wurde im Rahmen des DoCUMA-Projektes (ESF-080951831) erstellt, welches durch den Europäischen Sozialfonds (ESF), den Freistaat Sachsen und dem Industriepartner T-Systems Multimedia Solutions GmbH finanziert wird.

Literaturverzeichnis

[1] J. Coutaz, “Meta-User Interfaces for Ambient Spaces,” TAMODIA‘06 Proceedings of the 5th international conference on Task models and diagrams for users interface design, 2007, pp. 1-15.

[2] N. Ducheneaut, T.F. Smith, J. “Bo” Begole, M.W. Newman, and C. Beckmann, “The orbital browser,” CHI ’06 extended abstracts on Human factors in computing systems - CHI ’06, New York, New York, USA: ACM Press, 2006, p. 321.

[3] M.W. Newman, A. Elliott, and T.F. Smith, “Providing an Integrated User Experience of Networked Media , Devices , and Services through End-User Composition,” Proceeding Pervasive ’08 Proceedings of the 6th International Conference on Pervasive Computing, Springer Berlin, Heidelberg, 2008, pp. 213-227.

[4] A. Ranganathan and S. McFaddin, “Using workflows to coordinate Web services in pervasive computing environments,” Proceedings. IEEE International Conference on Web Services, 2004., 2004, pp. 288-295.

[5] J. Brønsted, K.M. Hansen, and M. Ingstrup, “Service Composition Issues in Pervasive Computing,” IEEE Pervasive Computing, vol. 9, Jan. 2010, pp. 62-70.

[6] S. Pietschmann, M. Voigt, A. Ruempel, and K. Meißner, “CRUISe: Composition of Rich User Interface Services,” Proceedings of the 9th International Conference on Web Engineering (ICWE 2009), San Sebastian: Springer LNCS, 2009, p. 473–476.

[7] S. Berti, F. Paternò, and C. Santoro, “A Taxonomy for Migratory User Interfaces,” INTERACTIVE SYSTEMS. DESIGN, SPECIFICATION, AND VERIFICATION Lecture Notes in Computer Science, 2006, Springer Berlin, Heidelberg, 2006, pp. 149 - 160.

[8] G. Ghiani, F. Paternò, and C. Santoro, “On-demand Cross-Device Interface Components Migration,” MobileHCI ’10 Proceedings of the 12th international conference on Human computer interaction with mobile devices and services, New York, New York, USA: 2010, pp. 299-307.

[9] M. Niederhausen, S. Pietschmann, T. Ruch, and K. Meißner, “Web-Based Support By Thin-Client Co-Browsing,” Emergent Web Intelligence: Advanced Semantic Technologies, Springer London, 2010, pp. 395-428.