Context-Aware Choreography Adaptation: A Survey

Senait Behre
Matrikelnummer:

Markus Schütterle
Matrikelnummer:

Marcel Zeller
Matrikelnummer:

***Zusammenfassung*—The abstract goes here. The abstract goes here.**

1. Einleitung

Menschen besitzen die Fähigkeit aus einem Gespräch den Kontext einer Situation oder Handlung zu verstehen. Dadurch erhöht sich der Informationsgehalt enorm. Computer sind dagegen nicht ohne Weiteres in der Lage aus einer Interaktion mit einem Menschen den Kontext zu extrahieren und zu nutzen. Um trotzdem basierend auf dem Kontext arbeiten zu können, muss eine Anwendung andere Informationsquellen bemühen. Dabei kann der Kontext als jede Information definiert werden, die verwendet werden kann, um die Situation einer Entität zu beschreiben. Eine Entität kann ein Ort, eine Person oder ein Objekt sein, das relevant für die Anwendung ist. Dies schließt Benutzer und die Anwendung selbst ein [1]. Nutzt eine Anwendung den Kontext bei der Durchführung ihrer Arbeit, ist sie demnach Kontext-sensitiv. Eine Service-Choreographie ist eine globale Beschreibung der teilnehmenden Services, die durch den Austausch von Nachrichten, Interaktionsregeln und Vereinbarungen zwischen zwei oder mehr Endpunkten definiert wird. Eine Choreographie verwendet einen dezentralisierten Ansatz für die Zusammenstellung von Dienstleistungen. Das bedeutet, dass die Services sich untereinander selbst organisieren. Eine Choreographie beschreibt im Gegensatz zur Orchestrierung die Interaktionen zwischen mehreren dezentralen Diensten. Wohingegen die Orchestrierung einen zentralen Dienst hat, von dem andere Services aufgerufen werden. Dabei wird das Zusammenspiel aller teilnehmenden Services vom zentralen Service gesteuert.

Da sich eine Infrastruktur ähnlich wie Businessentscheidungen ändern können, muss sich die Choreographie daran anpassen. Dies trifft auf funktionale Anforderungen wie auf einen zusätzlichen Service, sowie auf nicht-funktionale Anforderungen, wie Änderungen am Service-Level-agreement, zu [2]. Je nach Anforderung ist außerdem eine manuelle oder automatische Anpassung nötig. Insbesondere bei automatischen Anpassungen muss das System selbstständig entscheiden, was diese Anpassung im Detail erreichen soll und wie diese durchgeführt wird. Wird dazu der Kontext der Services herangezogen, ergibt sich eine Kontext-sensitive Choreographieanpassung.

Ziel der Arbeit ist die Identifizierung und Analyse des Stands der Technik in der kontextsensitiven Choreographie Anpassung. Die Identifizierung der relevanten Literatur erfolgt durch eine systematische Literaturrecherche [3], [4]. Im Gegensatz zum üblichen Prozess der Literaturrecherche verringert die systematische Literaturrecherche die Verzerrung und folgt einer genauen und strengen Abfolge von methodischen Schritten, die sich auf ein genau definiertes Protokoll stützen.

2. Forschungsmethode

Um den aktuellen Forschungsstand zur erfassen und zu evaluieren wird die systematische Literaturrecherche (SLR) nach Kitchenham [5], [6] verwendet. Die Methode dient dazu, den aktuell verfügbaren Forschungsstand zu einem bestimmten Thema zu identifizieren und zusammenzufassen. Dabei ist eine klare Methodik vorgegeben was die Ergebnisse weniger verzerrt [6]. Das Protokoll zur systematischen Literaturrecherche umfasst alle notwendigen Informationen, wie die Fragestellungen, Suchanfragen oder die Kriterien zur Auswahl der Ergebnisse. Durch das Protokoll ist die Recherche wiederholbar, sodass sich die Ergebnisse jederzeit reproduzieren lassen.

Das Protokoll der systematischen Literaturrecherche wurde auf Basis der Richtlinien von Kitchenham und Charters [6] entwickelt und durchgeführt. Für die Aufbau des Protokolls dienten die Protkolle von Kitchenham [4], Leite et al. [2] und Dyba [7] als Vorbild.

2.1. Forschungsfragen

Die Forschungsfragen sind an die Fragen der SLR von Leite et al. [2] orientiert. Dabei wurden die Fragen auf die Thematik der vorliegenden SLR angepasst. Dabei wurden folgende Forschungsfragen verwendet:

RQ1: Welche Strategie wählte die ausgewählte Studie für Context Aware Choreography Adaptation?

RQ2: Wie wählt eine ausgewählte Studie ihre Adaptionsstrategie gemäß den folgenden Aspekten aus?

RQ2.1 *Ziel*: Unterstützt die Anpassung funktionale oder nicht funktionale Anforderungen?

RQ2.2 *Erforderlicher Eingriffsgrad*: Wird die Anpassung automatisch durchgeführt? Oder ist menschliches Eingreifen notwendig?

RQ2.3 *Kontextsensitivität*: Welchen Kontext bezieht die Choreography Adaption Strategie in ihre Entscheidungsfindung mit ein? Wie stark wird die Entscheidungsfindung durch den Kontext beeinflusst?

RQ2.4 *Auswirkungen auf die Skalierbarkeit*: Ist die Strategie für die Skalierbarkeit der Choreographie diskutiert? Ist eine solche Diskussion informell oder enthält sie formale Beweise / Experimente?

RQ2.5 *Implementierungen*: Wird der vorgestellte Ansatz von einem Tool oder Prototyp implementiert? Ist die Implementierung des Systems verfügbar? Wenn ja, ist es Open-Source-Software?

RQ2.6 *Zugrundeliegende Modelle*: Welche Choreography Modelle oder Standards werden in der Stategie verwendet?

RQ3: Was sind die Hauptlimitationen und offene Forschungsfragen der Ansätze von "Context-aware choreography adaption"

2.2. Datenquellen

Um die Reproduzierbarkeit des durchgeführten systemantischen Literatur Reviews, wurde der Suchprozess durch Skripte automatisiert. Die Datenquellen wurden mit einem benutzerdefinierten Python Skript durchsucht. Dazu wurden die folgenden wissenschaftlichen Quellen mit einer definierten Zeichenabfolge durchsucht: IEEE Xplore[1] SpringerLink[2] und Google Scholar[3]

2.3. Abfragezeichenfolge

Die Suchen wurden mit Hilfe von Suchanfragen durchgeführt. Dabei wurden Titel und Abstract der Paper durchsucht. Die Suchanfragen sind an der Suchanfragen der SLR von Leite [2] orientiert und wird durch Suchwörter zu Context Aware erweitert.

Aus der Kombination entstanden 3 Suchblöcke. Der erste Block ist für *context-ware*, der zweite für *choreography* und der dritte Block für *adaption*. Die verschiedenen Strings aus einer Gruppe sind mit ODER (OR) verbunden. Die Suchblöcke werden mit UND (AND) verbunden. Dadurch entstehen Anfragen wobei aus jedem Block ein String enthalten sein muss. Daraus wurden sämtliche Möglichkeiten gebildet. Die Suchanfrage ist die folgende:

1. https://ieeexplore.ieee.org/Xplore/home.jsp
2. https://link.springer.com/
3. https://scholar.google.de/

(
„Context-aware/Context aware“ OR
„Context-specific/Context specific“ OR
„Context-dependent/Context dependent“ OR
„Context-sensitive/Context sensitive“ OR
„Situation-aware/Situation aware“
)
AND
ab hier vgl. Leite et al. [2])
(
„choreography“ OR
„decentralized composition“ OR
„decentralized service composition“ OR
„distributed composition“
)
AND
(
„adapt*“ OR
„self-config*“ OR
„auto-config*“ OR
„reconfig*“ OR
„sensitiv“ OR
„behaviour“
)

2.4. Einschluss und Ausschluss Kriterien

Da nicht jedes Ergebnis zum Thema passt, werden die Ergebnisse der Suche überprüft. Dabei werden nur Paper in die Ergebnisse der systematischen Literaturrecherche aufgenommen, die sich mit kontextabhängiger Anpassung von Servicechoreografien befassen. Dabei werden Ergebnisse ausgeschlossen, die keinen Kontext in die Anpassungsstrategie einbeziehen, oder für Orchestrationen sind. Kontextabhängige Anpassung einzelner Services wird auch ausgeschlossen, da der Fokus auf Choreografien mehrerer Services liegt.

Die gefundenen Ergebnisse durchlaufen, wie in Tabelle 1 gezeigt, mehrere Schritte, wobei passende Ergebnisse ausgewählt werden. Nach der automatisierten Suche waren nach Entfernen der Duplikate insgesamt 4435 Ergebnisse vorhanden. Die Duplikate wurde mit Hilfe von Microsoft Excel über den Titel der Ergebnisse aussortiert.

Nach dem Entfernen der Duplikate wurden die Ergebnisse auf Basis des Veröffentlichungsjahres und nach der Anzahl der Zitationen vorgefiltert. Alle Paper vor dem Jahr 2000 wurden ausgeschlossen. Für die Jahre 2000-2010 wurden mindestens 10 Zitationen als Einschlusskriterium verwendet. Für das Jahr 2011 wurden mindestens 9 Zitationen, für das Jahr 2012 mindestens 8 Zitationen, für das Jahr 2013 mindestens 7 Zitationen, für das Jahr 2014 mindestens 6 Zitationen, für das Jahr 2015 mindestens 5 Zitationen, für das Jahr 2016 mindestens 4 Zitationen, für das Jahr 2017 mindestens 3 Zitationen, für das Jahr 2018 mindestens 2 Zitationen und für das Jahr 2019 mindestens 1 Ziation vorausgesetzt. Als Ergebnis blieben 2722 Ergebnisse übrig.

Die verbleibenden Ergebnisse wurden manuell weiter sortiert. Dabei wurden vorher festgelegte Einschluss- und Ausschlusskriterien auf die Titel der Ergebnisse und im Zweifelsfall auf die Zusammenfassung angewendet, um die Zahl weiter zu reduzieren. Als Resultat blieben 89 Ergebnisse übrig.

Bei den verbleibenden 89 Ergebnissen wurden die Einschluss- und Ausschlusskriterien auf den ganzen Text angewendet. Daraus resultieren die Ergebnisse der vorliegenden systematischen Literaturrecherche. Die Ergebnisse werden in Abschnitt 3 zusammengefasst.

Tabelle 1. STUDIENAUSWAHLVERFAHREN

Stufe	Beschreibung
Stufe 1	Die Suchanfragen wurden auf allen angegeben Datenquellen angewendet und die Ergebnisse gespeichert.
Stufe 2	Duplikate und ungültige Paper wurden ausgeschlosssen.
Stufe 3	Die Inklusions- und Exklusionskriterien wurden auf die Titel der Paper und bei Unklarheit auf den Abstract angewendet.
Stufe 4	Die Inklusions- und Exklusionskriterien wurden auf den ganzen Text angewendet.

2.5. Datenextrahierung

Die Daten wurden mithilfe eines automatisierten Suchskriptes extrahiert. Jede der Anfragen wurde einzeln auf jeder der genannten Suchmaschinen ausgeführt. Die Ergebnisse der einzelnen Suchanfragen wurden jeweils in einer CSV-Datei gespeichert. Dabei wurden die folgenden Metadaten extrahiert:

- Titel
- Author(en)
- Veröffentlichungsjahr
- Papertyp
- Link zum Ergebnis
- Link zum PDF
- Zugehörige Suchanfrage
- Datenquelle
- Zahl der Zitationen
- Digitale Objekt ID (DOI) des Ergebnisses

Die Ergebnisse der einzelnen Suchfragen wurden anschließend in einer einzelnen CSV-Datei zusammengefasst, wobei 9168 Paper als Ergebnis resultieren. Aufgrund der Suchanfragen auf verschiedenen Suchmaschinen sind darin Duplikate enthalten. Die Duplikate wurden deshalb mit Hilfe von Microsoft Excel aus den Ergebnissen entfernt. Nach dem Entfernen der Duplikate bleiben 4435 Ergebnisse übrig.

2.6. Protokol Evaluierung und Limitierungen

Die systematische Literaturrecherche wurde mit einem Python-Script[4] automatisiert. Dazu wurden die Suchanfragen auf den verschiedenen Datenquellen ausgeführt und die Metadaten, wie Titel, Author oder Anzahl der Zitationen heruntergeladen. Dabei waren nicht für jedes Ergebnisse alle Metadaten vorhanden. Manche haben zum Beispiel keine Zitationen oder keinen *Digital Object Identifier (DOI Object Identifier* (DOI)

Bei Springer Link gibt es keine offizielle API für automatische Suchen. Die Anzahl der möglichen Anfragen ist nicht begrenzt und auch bei einer Vielzahl von Anfragen wird kein Bann gegen den Benutzer verhängt. Allerdings können maximal 1000 Seiten pro Suche von Springer Link abgefragt werden, weshalb der Crawler limitiert wurde.

Google Scholar beschränkt die Anzahl der Anfragen. Dies führt bei automatischen Suchen, wie im Rahmen dieser Arbeit wird der Benutzer temporär gebannt. Der Bann wird nach dem Crawlen von ungefähr 40 Seiten aktiv. Der Crawler für Google Scholar wurde deswegen auf 10 Seiten pro Suchanfrage begrenzt. Pro Seite werden 10 Ergebnisse angezeigt. Folglich werden für jede Suchanfrage maximal 100 Ergebnisse gecrawlt.

IEEE Xplore bietet eine API[5] für automatisierte Suchanfragen an.

3. Context-Aware Choreography Adaptation Stategien

Dieser Abschnitt präsentiert die synthetisierte Daten, um die Forschungsfragen zu beantworten. Zunächst wird ein Überblick über kontextsensitive Choreographie-Adaptionsstrategien gezeigt, die **RQ1** beantworten. Anschließend werden die Ergebnisse über die kontextsensitive Adaptionsstrategie zusammengefasst, was **RQ2** beantwortet. Am Ende wird **RQ3** beantwortet und die Hauptlimitationen und offenen Forschungsfragen aufgezeigt. Werden Fragen nicht beantwortet bedeutet dies, dass in dem Paper darauf nicht eingegangen wurde.

3.1. Automated Context-Aware Adaptation of Web Service Executions

Narendra und Gundugola [8] zeigen einen Ansatz zur Anpassung von zusammengesetzten Web Service Ausführungen, wobei funktionale Anforderungen angepasst werden (RQ2.1). Dabei zeigen sie, wie die Anpassungen mithilfe von Kontextontologien modelliert werden können. Zur automatischen Anpassung werden Kontextinformationen auf verschiedenen Ebenen der Web Services identifiziert (RQ2.2). Diese werden in verschiedene Typen klassifiziert: C-Kontext für den zusammengesetzten Service, W-Kontext für den Web Service Provider und I-Kontext für einzelne Service-Instanzen, die von den Web Service Providern erstellt werden (RQ2.3) [8]. Anpassungen in einem Web Service werden immer dann benötigt, wenn sich die Spezifikation einer Komponente ändert [8]. Dies kann beispielsweise der Austausch eines Services durch einen

4. https://github.com/Lenny925/Context-Aware-Choreography-Adaptation

5. https://developer.ieee.org/docs/read/Searching_the_IEEE_Xplore_Metadata_API

anderen mit erweiterten Anforderungen sein. Als Beispiel für die Workflow-Anpassung wird die Ausnahmebehandlung genannt, die sich mit Mechanismen zur Wiederherstellung des Zustands des Workflows im Falle eines Fehlers während der Workflow-Ausführung befasst [8]. Dieser Fall tritt ein, wenn eine der Web-Service-Komponenten nicht ausgeführt wird [8]. Zur automatischen Anpassung werden Spheren verwendet, welche eine Sammlung von Aufgaben repräsentieren, die entweder komplett oder überhaupt nicht ausgeführt werden [8]. Spheren werden mit einem Algorithmus identifiziert. Der Algorithmus zur Anpassung des Workflows, spiegelt die Rückwärtsbewegungen des Spheren-Bestimmungsalgorithmus wider [8]. Der Ansatz soll in Zukunft noch in einem Prototyp implementiert werden, um die Machbarkeit zu beweisen (RQ2.5). Außerdem sollen die Anpassungstechnik weiter erweitert werden, um weiteren Kontext einzubeziehen (RQ3).

3.2. Modeling of Context-Aware Self-Adaptive Applications in Ubiquitous and Service-Oriented Environments

Der Ansatz von Geihs et al. [9] zielt auf Umgebungen mit dynamisch verfügbaren Diensten ab, die von Anwendungen genutzt werden können. Dabei ist das Ziel die Anpassung der funktionalen Anforderungen (RQ2.1). Die Anpassungsentscheidungen hängen dabei nicht nur von den Kontext-Eigenschaften ab, sondern auch von der Verfügbarkeit der Dienste (RQ2.3). Um Dienste dynamisch anzupassen, muss der Adaptionsansatz verschiedene Anforderungen erfüllen. Dazu gehören die Variabilität der Anwendung, eine dynamische Serviceerkennung, Heterogenität und die Integration von Service- und Kontext-Eigenschaften in die Planung [9]. Zur Adaption werden bei einem signifikanten Kontextwechsel alle verfügbaren Anwendungsvarianten von der Middleware ausgewertet und auf Basis verschiedener QoS-Metadaten verglichen, die den beteiligten Komponentenrealisierungen zugeordnet sind [9]. Komponententypen sind Variationspunkte, die einer bestimmten Teilefunktionalität der Anwendung entsprechen. Die Komponententypen werden rekursiv ausgewählt und der Prozess wird gestoppt, wenn ein atomarer Realisierungsplan ausgewählt wurde. Wenn Dienste für die Middleware nicht mehr verfügbar sind, wird der Realisierungsplan verworfen und ein neuer Anpassungsprozess wird ausgelöst [9]. Der Ansatz ist bereits teilweise implementiert und wurde verwendet, um erste Prototypen zu implementieren (RQ2.5). Weitere Forschungsmöglichkeiten gibt es im Bezug auf Performance und Skalierbarkeit (RQ3).

3.3. Semantic Web Service Adaptation Model for a Pervasive Learning Scenario

Lau et al. [10] liefern einen halbautomatischen Adaptionsansatz zur Abstimmung und Anpassung von relevanten Web-Services (RQ2.2). Das Ziel ist die Anpassung von funktionalen und nicht funktionalen Anforderungen (RQ2.1). Um einen Service anzupassen oder durch einen anderen zu ersetzen wird eine *Service Requirement Specification* verwendet, in welcher beschrieben wird, wie ein Service auszusehen hat. Außerdem werden Dienste mit einem sogenannten *Service Descriptor* beschrieben. Die Anpassung besteht aus zwei Schritten die hintereinander ablaufen. Zuerst werden relevante Dienste basierend auf der Beschreibung des Services aus einer Datenbank gesucht, um die Serviceanforderungen für eine Aufgabe zu erfüllen. Die relevanten Services dienen als Input für die folgende Adaptionsphase, die dazu dient die relevanten Dienste an die aktuelle Situation und den Interessen der Nutzer anzupassen (RQ2.3). Der Adaptionsprozess besteht aus den drei Phasen Bewertung/Klassifizierung, Filterung und Ranking [10]. Bei der Klassifizierung werden die Services entsprechend der aktuellen Situation und Klassen eingeteilt. In der Phase des Filterns werden irrelevante Dienste über die Klassifizierung herausgefiltert. Anschließend wird basierend auf ausgewählten Merkmalen ein Ranking erstellt, wobei die Services in eine Rangfolge gebracht werden. Der Ansatz wurde in einem Prototyp implementiert, um die Funktionalität zu validieren (RQ2.5).

3.4. A Conceptual Model for Adaptable Context-aware Services

Autili et al. [11] schlagen einen Ansatz für anpassbare kontext- und serviceorientierte Anwendungen vor. Dieser basiert auf einer 2-Layer-Architektur, welche aus Service-Layer und Komponenten-Layer besteht. Der Komponenten-Layer bildet die Rechenressourcen für die vernetzen Dienste ab, welche auf dem Serice-Layer abgebildet werden. Softwarekomponenten können zu jeder Zeit bereitgestellt, aktualisiert oder entfernt werden, ohne die Funktionalität zu beeinflussen (RQ ii). Das Mapping zwischen Komponenten und Services wird auf Service-Layer Ebene realisiert. Die Adaption der Services soll funktionale und nicht funktionale Anforderungen verbessern, wobei das Ziel ist, einen optimalen Kompromiss zwischen Bedürfnissen der Nutzer und den aktuellen verfügbaren Ressourcen zu erreichen (RQ2.1) [11]. Jeder Service hat eine Beschreibung, wobei zusätzliche QoS (Quality of Service) Attribute hinzugefügt werden. Alle verfügbaren Services werden in einer Service-Registry gespeichert. Services werden durch Serviceanfragen nachgefragt, wobei mehrere Anforderungen an den Service enthalten sind. Wenn ein Service aus der Registry passt wird dieser verwendet. Die Anfrage enthält zusätzliche Informationen wie verfügbare Speicher, Größe des Bildschirms oder der Typ der Netzwerkverbindung, was Informationen zum Kontext sind in der der Service laufen wird (RQ2.3). Diese werden verwendet, um den nachgefragten Service an die Eigenschaften des Gerätes anzupassen, auf dem er laufen soll. Der Ansatz wurde als Prototyp implementiert (RQ2.5). Es gibt weitere Forschungsmöglichkeiten bei der Verfeinerung des Kontextes (RQ3).

3.5. Context-Aware Service Composition A Methodology and a Case Study

Bastida et al. [12] zeigen einen Ansatz und die notwendigen Schritte, um kontextsensitive Servicekompositionen zu entwickeln. Dabei ist das Ziel die Verbesserung von funktionalen und nicht funktionalen Anforderungen (RQ2.1) Dieser besteht aus sechs Schritten. Im ersten Schritt wird die Architekturspezifikation entwickelt, wobei auch Architekturalternativen identifiziert und bewertet werden, um die Systemanforderungen zu erfüllen. Im zweiten Schritt wird die funktionale Spezifikation ausgearbeitet, wobei alle Subfunktionalitäten identifiziert werden, um die endgültige Funktionalität der Komposition abzudecken [12]. Im dritten Schritt wird die geeignete Komposition zur Designzeit identifiziert, sodass funktionale als auch nicht-funktionale Anforderungen erfüllt werden. Im vierten Schritt werden variable Punkte in der Komposition identifiziert. Dabei wird zur Designzeit identifiziert, welche Teile der Komposition sich ändern können und Anforderungen zur Anpassung definiert (RQ2.3) [12]. Das Ziel dabei ist die Funktionalität der Komposition zur Laufzeit automatisch aufrechtzuerhalten (RQ2.2). Im fünften Schritt werden geeignet Services anhand der abstrakten Beschreibungen ausgewählt, um die endgültige ausführbare Komposition zu erhalten. Im letzten Schritt wird verifiziert, dass die ausgewählten Dienste ihre spezifizierte Funktionalität und QoS-Anforderungen erfüllen [12]. Die Anpassung der Komposition wird mithilfe von Regeln realisiert. Bei Variationspunkten werden die Regelbedingungen ausgeführt und ein konkreter Dienst für den Variationspunkt ausgewählt. Dies geschieht auf Basis von definierten Varianten oder Alternativen für einen Variationspunkt. Der Ansatz wurde mithilfe einer Fallstudie evaluiert, wobei das Hauptproblem die Notwendigkeit einer stabilen Plattform war, welche dynamische Kompositionen zur Design- und zur Laufzeit unterstützt (RQ2.5) [12]. Außerdem gibt es Verbesserungsmöglichkeiten des Ansatzes, um die möglichen Varianten dynamisch anzupassen. Zudem bestehen Forschungsmöglichkeiten bei der nachträglichen Anpassung der Komposition, bei der Teile durch eine Subkomposition ersetzt werden, um die gleiche Funktionalität zu erhalten (RQ3).

3.6. Towards Context-aware Semantic Web Service Discovery through Conceptual Situation Spaces

Um Kontextanpassungsfähigkeit zu erreichen, schlagen Dietze et al. [13] *Conceptual Situation Spaces* (CSS) vor, welche die Beschreibung von situationsabhängigen Charakteristiken in geometrischen Vektorräumen erlauben (RQ2.3). Das Ziel ist die Verbesserung von funktionalen und nicht funktionalen Anforderungen (RQ2.1). Dabei beschreibt ein CSS einen bestimmten Kontext für eine bestimmte Situation. Die CSS sind dabei auf semantische Web Services (SWS) ausgerichtet. Die semantische Ähnlichkeit zwischen Situationen wird dabei in Form ihrer euklidischen Entfernung innerhalb eines CSS berechnet [13]. Dabei werden die am besten geeigneten Ressourcen, wie Dienste oder Daten, automatisch basierend auf der semantischen Ähnlichkeit identifiziert (RQ2.2). Dieser werden aufgrund einer vorgegeben realen Situation aus Ressourcenbeschreibungen ausgewählt [13]. Der Ansatz wurde in einem Prototyp implementiert, mit dem die Machbarkeit nachgewiesen wurde (RQ2.5). Der Ansatz wird als Schritt nach vorne gesehen, wobei weitere Forschungsmöglichkeiten bei der Priorisierung der verschiedenen Dimensionen eines Vektorraumes genannt werden, um den Ressourcenallokationsprozess besser auf die Präferenzen des Benutzers anzupassen. Außerdem sollen die Beschreibung weiterer relevanter Kontextinformationen durch das CSS-Modell ermöglicht werden (RQ3).

3.7. Context aware service composition

Vukovic [14] präsentiert einen Ansatz, um kontextsensitive Anwendungen zu entwickeln. Dabei werden die Anwendungen als dynamische Zusammensetzung von Diensten betrachtet. Ändert sich der Kontext, in dem die Anwendung läuft, kann dies zu einer dynamischen Anpassung der Zusammensetzung führen (RQ2.2). Dabei wird der aktuelle Kontext des Benutzers der Anwendung verwendet (RQ2.3). Fragt der Nutzer einen Service nach, wird in der Anfrage zum einen eine Beschreibung der Aufgabe übermittelt und zum anderen kontextabhängige Parameter, wie das Gerät von dem die Anfrage kommt, oder die Information das der Nutzer sich gerade auf einer Straße mit einem Fahrzeug bewegt. Der Ansatz zielt auch auf Ausfälle von Diensten ab. Tritt ein Fehler bei der Erstellung der Zusammensetzung auf, wird versucht die Anfrage in eine Alternative Zusammensetzung zu ändern. Änderungen von Kontext oder Ausfälle werden auch während der Ausführung überwacht und die Zusammensetzung der Services wird entsprechend angepasst. Das Ziel der Anpassung sind folglich die funktionale sowie nicht-funktionale Anforderungen (RQ2.1). Um Ausfälle zu kompensieren und den Kontext zu erfassen wird ein System namens GoalMorph vorgestellt, welches bei Ausfällen auf alternative Zusammensetzungen wechselt. Der Ansatz beschreibt die Implementierung eines Frameworks für kontextsensitive Service Kompositionen. Der Ansatz wurde als Prototyp implementiert (RQ2.5). Als weitere Forschungsmöglichkeiten wird eine benutzerdefinierte Anpassung an die Wünsche, eine Priorisierung der Anfragen sowie eine Verbesserung der QoS. Zudem werden Datenschutz, Sicherheit und Vertrauen als weitere Forschungsmöglichkeiten in Umgebungen mit Services von verschiedenen Anbietern genannt, da Kontextinformationen von Nutzern oft sensibel sind (RQ3).

3.8. Behavioural Self-Adaptation of Services in Ubiquitous Computing Environments

Camara, Canal und Salaün [15] stellen einen Ansatz zur Selbstanpassung von Services vor. Dabei wird sowohl die Zusammensetzung der Services zur Designzeit als auch die dynamische Anpassung der Zusammensetzung zur Laufzeit

thematisiert. Dabei zielt der Ansatz auf das Protokoll oder die Verhaltensregeln ab. Dazu schlagen die Autoren ein Service Schnittstellenmodell vor, dass sowohl die Signatur als auch das Verhalten beinhaltet. Der Ansatz zur Anpassung ist ein dynamisches System, bei dem sich Dienste zur Laufzeit dynamisch an- und abmelden können. Der Ansatz ist vollständig automatisiert und wird durch eine Anforderungsbeschreibung vom Benutzer gesteuert (RQ2.2, RQ2.3) [15]. Wenn ein neuer Service der Choreography beitritt oder sie verlässt laufen drei Aufgaben ab. Im ersten Schritt wird ein Vektor aus mehreren Serviceschnittstellen durch abstrakte Operationssignaturen instanziiert [15]. Im zweiten Schritt wird eine Liste von stabilen Zuständen ermittelt, welche Dienstprotokolle und Vektoren enthalten. Ein stabiler Zustand des Systems ist einer, bei dem sich jeder der Dienste im System in einem stabilen Zustand befindet [15]. Dienste können nur in solch einem stabilen Zustand hinzugefügt oder entfernt werden. Nach den ersten beiden Schritten wird die Ausführung des Systems gestartet. Dazu wird eine Laufzeit-Engine vorgestellt, die solch ein System ausführt. Zur Selbstanpassung werden Regeln vorgeben, die beispielsweise die Sicherheit betreffen. Dort wird beispielsweise definiert, was nicht passieren darf, wenn ein Service mit dem restlichen System kommuniziert. Zudem werden sogenannte Lebendigkeitseigenschaften definiert, was passieren soll, wenn das System mit dem Rest interagiert. Diese gewährleisten, dass der Ablauf eines Services an einem gewünschten Punkt gehalten wird. Die Anpassung durch Hinzufügen oder Entfernen von Services im Ablauf geschieht immer unter Einhaltung der Regeln. Der Ansatz zielt auf die Verbesserung von funktionalen und nicht funktionalen Anforderungen ab (RQ2.1) Von dem vorgeschlagenen Ansatz existiert kein Prototyp, wobei die Entwicklung eines solchen in den weiteren Forschungsmöglichkeiten thematisiert wird (RQ2.5, RQ3).

3.9. Context-aware Adaptive Service Mashups

Dorn et al. [16] stellen einen Ansatz zur kontextabhängigen Selbstanpassung von Service Kompositionen vor, bei dem die Fähigkeiten der Services mit den permanent aktualisierten Anforderungen an die Fähigkeiten abgeglichen werden. Dabei ist das Ziel die Verbesserung von funktionalen Anforderungen (RQ2.1). Der Anpassungsprozess besteht aus zwei Phasen, wobei in der ersten Phase Kontextänderungen und der betroffene Teil der Service Zusammensetzung ermittelt wird. Eine Konfiguration wird neu bewertet, wenn Anforderungsregeln durch die Kontextänderung betroffen sind. Die Anforderungsregeln generieren neue Fähigkeitseinschränkungen und eine Neuanpassung wird gestartet, wenn die bestehende Service Konfiguration nicht mehr mit den Einschränkungen übereinstimmt [16]. Mit den Anforderungen wird eine Liste aus Services zusammengestellt, deren Fähigkeiten die Anforderungen am besten erfüllen. Die Services werden nach Gruppen geordnet und die Services jeder Gruppe bewertet, aus denen der Entwickler dann die beste Kombination auswählt. Der Ansatz ist folglich halbautomatisch (RQ2.2). Dies soll durch zukünftige Arbeiten durch automatisches generieren der Anforderungen automatisiert werden. Als Kontext zur Anpassung wird die Reputation der Services, die Organisationen die Service-Zusammensetzung verwenden und der physische Ort verwendet (RQ2.3). Der Ansatz wurde mit einer Fallstudie evaluiert (RQ2.5).

3.10. Towards Context-Aware Adaptable Web Services

Keidl und Kemper [17] zeigen einen Ansatz für ein Kontext-Framework, das die Entwicklung und Bereitstellung von kontextabhängigen und anpassbaren Webservices erleichtert. Dabei ist das Ziel die Anpassung von funktionalen und nicht funktionalen Anforderungen (RQ2.1) Der verwendete Kontext beinhaltet alle Informationen über den Client eines Web Services, die vom Web Service verwendet werden können, um die Ausführung und Ausgabe anzupassen und dem Client ein individuelles und personalisiertes Verhalten zu ermöglichen (RQ2.3) [17]. Der Kontext des Clients wird in der Anfrage an den Service übergeben, wobei der übergebene Kontext optional ist. Wenn der angefragte Service während seiner Ausführung einen anderen Service aufruft, wird der aktuelle Kontext automatisch in die Anfrage an den weiteren Service eingefügt (RQ2.2). Die Services nehmen sich jeweils die relevanten Kontextinformationen heraus. Der Kontext wird in der Serviceantwort zurückgeschickt. Bei dem Ansatz wird zwischen explizitem und automatischen Kontext unterschieden. Expliziter Kontext sind Informationen, die in den Anfragen übergeben wurden und automatischer Kontext wird automatisch aus dem expliziten Kontext generiert. Der Kontext wird durch verschiedenen Kontexttypen definiert. Es werden Typen wie Standort, Client, Verbraucher und Verbindungseinstellungen verwendet. Der Typ Client enthält Informationen, wohingegen der Typ Verbraucher Informationen wie Name oder E-Mail enthält. Der Typ Verbindungseinstellungen enthält wichtige Informationen zur Verbindung selbst. Das Kontext-Framework der Prototyp sind in Java implementiert und basieren auf Standards wie XML, SOAP, UDDI und WSDL (RQ2.5, RQ2.6) [17]. Bei dem Ansatz bestehen weitere Forschungsmöglichkeiten bei zusätzlichen Kontexttypen und einer genaueren Verarbeitung des Kontextes (RQ3). Außerdem gibt es weiteren Forschungsbedarf bei den Sicherheitsrichtlinien, sodass festgelegt werden kann, welche Services Zugriff auf welchen Kontext haben und welche Dinge damit gemacht werden dürfen (RQ3).

3.11. Context-Aware Service Composition in Pervasive Computing Environments

Mokhtar et al. [18] zeigen einen Ansatz zur dynamischen und kontextabhängigen Zusammenstellung von Services, der auf der Integration des Workflows basiert. Dabei ist das Ziel die Berücksichtigung des Kontextes des Benutzers und der kontextuellen Anforderungen der Dienste (RQ2.1) [18]. Darüber hinaus ist das Ziel die Sicherstellung der QoS-Anforderungen des Benutzers. In dem Ansatz werden die

Services mit der Ontology Web Language for Services (OWL-S) beschrieben (RQ2.6). Diese werden mit Kontextinformationen erweitert, wobei sich diese in hochrangige Kontextattribute und kontextuelle Voraussetzungen und Auswirkungen aufteilen. Kontextattribute sind beispielsweise Standort oder physische Bedingungen und kontextuelle Voraussetzungen sind Bedingungen, die für die Ausführung eines Services erfüllt sein müssen. Wie die Services werden auch die Benutzeraufgaben in OWL-S Prozessen beschrieben. Diese werden mit nicht-funktionalen Anforderungen, wie QoS-Anforderungen und Kontextbedingungen erweitert [18]. Das Kontextmanagement basiert auf einem Kontextmanager, der bei Bedarf Kontextinformationen bereitstellt [18]. Zur Modellierung der OWL-S Prozesse wird eine formale Modellierung als endliche Automaten eingeführt. Der Ansatz zur kontextabhängigen Servicekomposition ist zweistufig. Im ersten Schritt werden Services identifiziert, die atomare Prozesse bereitstellen und semantisch äquivalent in Bezug auf die bereitgestellten Fähigkeiten sind [18]. Darüber hinaus werden dabei die kontextuellen Anforderungen mit den Kontextattributen der Dienste abgeglichen. Im zweiten Schritt werden die Services integriert, die im ersten Schritt identifiziert wurden, um den Prozess des Benutzers dem Kontext anzupassen. Die Integration wird mit dem neu eingeführten Modell der endlichen Automaten durchgeführt. Der Ansatz wurde zum Zeitpunkt des Papers noch nicht durch einen Prototypen validiert, aber es soll Teil künftiger Arbeiten werden, diesen zu bauen und den Ansatz zu validieren (RQ2.5, RQ3).

3.12. Towards Context-Aware Composition of Web Services

Luo et al. [19] präsentieren ein Framework zu kontextabhängigen Komposition von Web-Services, wobei passende Services nach den Anforderungen des Benutzers gesucht, zusammengestellt und mit Petrinetzen auf Korrektheit überprüft werden. Der Ansatz unterstützt funktionale und nicht funktionale Anforderungen (RQ2.1). Zur Modellierung der Dienste wird OWL-S verwendet (RQ2.6). Angebotene Services werden von den Anbietern in einer Registry registriert. Für das automatische Suchen, Zusammenstellen, Aussuchen und Ausführen der Services werden Agents verwendet (RQ2.2). Für das Annehmen der Anfragen sowie für das Antworten des Ergebnisses wird ein User Agent verwendet. Dieser sammelt Kontextinformationen über den Benutzer. Diese beinhalten beispielsweise die ID, der Ort, die Fähigkeiten des Endgeräts oder die Qualität des gewünschten Web-Services (RQ2.3). Außerdem speichert der User Agent die Präferenzen des Benutzers von bisherigen Anfragen. Für das Suchen und Auswählen der Services ist der Broker Agent verantwortlich. Dieser extrahiert den Kontext aus der Anfrage und versucht den am besten passenden Service aus der Registry zu ermitteln. Gelingt ihm dies nicht exakt, wird die Anfrage in verschiedene Unterziele zerlegt, um annähernd passende Dienste zu finden [19]. Dabei wird eine Liste von möglichen Kompositionen zusammengestellt. Nach dem Finden und Zusammenstellen der Services sucht der Broker Agent die beste Komposition aus. Dies geschieht auf Basis des aktuellen Kontextes und der Einschränkungen der Service Anbieter. Am Ende wird eine Ausführungssequenz generiert und die Informationen an den Service Execution Agent übergeben. Dieser ist für das Aufrufen der involvierten Services verantwortlich. Der Service Execution Agent besteht aus einem Prozess Verifizierer und einer Ausführungs-Engine. Der Prozess Verifizierer prüft mit Petrinetzen ob die Ausführungssequenz machbar ist und ob alle Services miteinander kompatibel sind [19]. Die Ausführungs-Engine führt die verifizierten Services der Reihe nach aus. Nach dem Ausführen wird das Ergebnis an den Benutzer übermittelt und das Prozessmodel wird zusammen mit dem Kontext in dem Cache für zukünftige Anfragen gespeichert. Das vorgeschlagene Modell mit den Petrinetzen wurde mit Hilfe eines Erreichbarkeitsbaumes validiert. Der Ansatz wurde in einem Projekt implementiert (RQ2.5). In zukünftigen Arbeiten soll die Modifikation der Service Kompositionen zur Laufzeit untersucht werden, um beispielsweise auf Ausfälle von Services zu reagieren (RQ3).

3.13. A Conceptual Framework for Provisioning Context-aware Mobile Cloud Services

La und Kim [20] stellen ein Framework vor, das verteilte Services, die von mobilen Endgeräten eingebunden werden, kontextbasiert anpasst. Dabei wird sowohl die Auswahl die einzubindenden Services als auch dessen Anpassung bei Kontextänderungen beeinflusst. Das Framework bietet eine Entscheidungsfindung bei der Frage nach der notwendigen Anpassung bei einer Kontextänderung, indem zunächst der Unterschied zwischen altem und neuem Kontext analysiert und kategorisiert wird. Darauf basierend wird ein passender Adaptertyp aus einer vordefinierten Menge ausgewählt. Beispiele sind ein Interface-Adapter, der unterschiedliche Services mit der gleichen Funktionalität, die sich nur im Kontext wie etwa dem Land, für das sie gedacht sind, unterscheiden, bedienen kann sowie Reroute-Adapter, der verschiedene Instanzen desselben Services bedienen kann, zum Beispiel um die Latenz nach einem Ortswechsel zu verbessern.

Das Framework kann mit verschiedenen Kontexttypen umgehen (RQ2.3): Geräte-Kontext, Nutzer-Einstellungen, Situations-Kontext und Service-Kontext. Der Kontext wird dabei von mobilen Endgeräten wie Smartphones erhoben. Die Kategorien der Auswahl des passenden Adapters beinhalten sowohl funktionale als auch nicht-funktionale Anforderungen (RQ2.1). Die Anpassungen werden automatisch vorgenommen (RQ2.2). Das Framework zielt auf die Nutzung mit Services in Cloudumgebungen ab. Die dadurch mögliche Skalierung wird nicht eingeschränkt, aber auch nicht gefördert (RQ2.4). La und Kim [20] haben das Framework in einer Case Study auf ein System angewendet und positive Ergebnisse erhalten (RQ2.5). Das Framework beruht auf verteilten Services, die von mobilen Endgeräten direkt eingebunden werden (RQ2.6).

3.14. A context-aware adaptive web service composition framework

Cao et al. [21] nutzen ein Framework aus drei Teilen, um eine Web Service Choreography Kontext-basiert anzupassen. Das *Web Service Composition Module* verwaltet die Choreography, das *Context-Aware Module* prüft den Kontext auf Änderungen und das *Adaptive Management Module* ändert basierend auf den festgestellten Kontextänderungen die Komposition der Services. Dazu wird mit Hilfe der Änderungen die am besten passenden Policy aus einer Policy-Bibliothek ausgewählt. Eine Policy gibt eine bestimmte Komposition der Services vor. Diese wird dann auf die Web Service Komposition angewendet.

Das Framework von Cao et al. [21] nimmt ausschließlich Nicht-Funktionale Änderungen vor (RQ2.1). Es arbeitet automatisch (RQ2.2) und unterscheidet zwischen Service-, Physical- und User-Kontext. Diese Kategorien werden sehr weit gefasst, sie schränken den beachteten Kontext nicht ein. Es wird dabei nicht auf die Erhebungsmethode eingegangen (RQ2.3). Auf Skalierbarkeit wird nicht eingegangen (RQ2.4). Ein Prototyp wird von Cao et al. [21] vorgestellt (RQ2.5). Das Framework nutzt eine auf BPEL aufbauende Web Service Choreography (RQ2.6).

3.15. Context-Aware Service Adaptation: An Approach Based on Fuzzy Sets and Service Composition

Madkour et al. [22] beschreiben sehr detailliert, wie Kontext formal dargestellt werden kann. Dazu werden einem relevanten Objekt, etwa dem Nutzer, verschiedene Verbindungen zu anderen Objekten zugewiesen. Eine Verbindung hat dabei eine Mächtigkeit. Zum Beispiel hat ein Nutzer eine Verbindung zu einem Ort-Objekt mit einer Mächtigkeit, die die Entfernung zu diesem Ort angibt. Die formale Beschreibung ermöglicht die gewünschten Anpassungen der Service Komposition, die sich aus der am besten zur Situation passenden Policy ergibt, ebenfalls formal darzustellen. Dies ist dann eine formal definierte Anforderung zur Änderung der Komposition. Daraus werden unter Einsatz verschiedener Methoden abstrakte Ziel-Kompositionen erstellt. Madkour et al. [22] nennen als Beispielmethode Artificial Intelligence Planning, was jedoch nicht weiter beschrieben ist. Eine abstrakte Ziel-Komposition kann dann auf die tatsächliche Service-Komposition angewendet werden.

Der Ansatz unterstützt Funktionale und Nicht-Funktionale Anforderungen (RQ2.1) und funktioniert automatisch (RQ2.2). Es werden der Nutzer- und Service-Kontext beachtet, wobei die Umgebung und Situation dem Nutzer zugeordnet wird und damit ebenfalls abgedeckt ist (RQ2.3). Skalierbarkeit wird nicht thematisiert (RQ2.4). Ein Prototyp ist nicht vorhanden, jedoch angedacht (RQ2.5). Der Ansatz beruht auf Service-Kompositionen, die als Choreography klassifiziert werden können (RQ2.6).

3.16. PerCAS: An Approach to Enabling Dynamic and Personalized Adaptation for Context-Aware Services

Yu et al. [23] stellen einen Model-basierten Ansatz zur Kontext-basierten Adaption von Web Service Kompositionen vor. Dazu wird die grundlegende Logik eines Service in einem *Base Functionality Model* dargestellt. Die Kontext-abhängige Logik wird als *Personalized Context-Awareness Logic Model* dargestellt. Ein solches Model besteht aus Regeln, die je aus einer Voraussetzung und einer daraus resultierenden Aktion bestehen sowie einem Typ. Regeln desselben Typs können zur Laufzeit dynamisch ausgetauscht werden. Dadurch kann jederzeit ein für einen Nutzer passendes Regel-Set erstellt und angepasst werden. *Aspects* ordnen eine Regel einer Aktivität der grundlegenden Logik zu, indem diese eine Aktivität und einen Regel-Typ enthalten. Wird bei Durchführung einer Aktivität eine Voraussetzung einer zugeordneten Regel erfüllt, wird die Aktion dieser Regel ausgeführt.

Regel dieses Ansatzes können für Funktionale und Nicht-Funktionale Anforderungen erstellt werden (RQ2.1). Die Auswahl der Regeln erfolgt manuell oder durch ein zusätzliches System (RQ2.2). Es wird ausschließlich Nutzer-Kontext genutzt (RQ2.3). Aus Skalierbarkeit wird nicht eingegangen (RQ2.4), ein Prototyp ist vorhanden (RQ2.5). Der Ansatz nutzt eine Web Service Choreography (RQ2.6).

3.17. Context-aware autonomous web services in software product lines

Alférez und Pelechano [24] haben in ihrem Ansatz detailliert beschrieben, wie mit Hilfe von *Software Product Lines (SPL)* Web Services automatisch Kontext-basiert angepasst werden können. Der Ansatz beruht auf *Dynamic Software Product Lines* [25], was es ermöglicht, SPL-basierte Services dynamisch zu laden. Dazu werden bereits bei der Entwicklung Punkte in der Ausführsequenz festgelegt, an denen die Software je nach definierten Parametern unterschiedlich fortgeführt wird. Diese Punkte werden genutzt, um Änderungen an der Service-Komposition einzuleiten. Der Ansatz setzt externe Services als Kontextquelle voraus, die die genannten Parameter definieren.

Der Ansatz unterstützt je nach Kontextquelle Funktionale und Nichtfunktionale Anforderungen (RQ2.1). Der Ansatz funktioniert selbstständig (RQ2.2), er geht nicht auf Skalierbarkeit ein (RQ2.4). Der verwendete Kontext hängt komplett von externen Services ab (RQ2.3) und beruht auf einer Web Service Choreography (RQ2.6). Eine Umsetzung ist als Case Study vorhanden (RQ2.5).

3.18. Context-aware pervasive service composition and its implementation

Zhou et al. [26] beschreiben sehr abstrakt, welche Schritte für eine Kontext-basierte Anpassung von Service-Kompositionen notwendig sind. Dies sind (i) das Sammeln

von Kontext-Informationen, (ii) die Beschreibung vorhandener Services, (iii) das Untersuchen der modellierten Services gegen die gesammelten Daten und (iv) die Anpassung der Service-Komposition aufgrund der Ergebnisse des dritten Schritts. Der letzte Schritt kann die verwendeten Services ändern, die Logik eines Service ändern und Hilfsfunktionen ändern. Der Ansatz beschreibt nicht, wie all dies umgesetzt werden kann, sondern bietet lediglich eine beispielhafte Umsetzung (RQ2.5).

Schritt (ii) erfolgt manuell, die anderen automatisch (RQ2.2). Der Ansatz ist gedacht für Service Choreographies (RQ2.6). Da der Ansatz keine Funktionsweise, sondern lediglich eine abstrakte Theorie beinhaltet, sind RQ2.1, RQ2.3 und RQ2.4 nicht zutreffend.

3.19. Context-aware pervasive service composition and its implementation

Romero et al. [27] stellen eine Plattform für Kontext-basierte Web Services vor. Diese nutzt mehrere Frameworks für verschiedene Aufgaben und kombiniert diese. Das *Context Entities Composition and Sharing (COSMOS)* Framework [28] ist dafür verantwortlich, Kontextinformationen zu sammeln und bereitzustellen. Dafür werden alle für einen Service notwendigen Kontextinformationen in Policies modelliert. Diese wiederrum werden in möglichst kleine Teile geteilt und hierarchisch gruppiert. Jedes Teil ist für eine Information verantwortlich und sammelt diese mit Hilfe von Operatoren. Standardoperatoren, etwa zum Auslesen von Betriebssystemdaten und Benutzereinstellungen, sind im Framework enthalten. Alle weiteren müssen selbst implementiert werden. Die Kontext Policies werden mit Hilfe des im Ansatz vorgestellten *SPACES* Framework [29] in einer verteilten Umgebung auf die Services verteilt.

Die Plattform nutzt Kontrollzyklen bestehend aus der Kontextbeschaffung, der Analyse des Kontexts, des Planens der Anpassungen an den Services und der Durchführung der Anpassungen. Schritt 1 wird mit *COSMOS* durchgeführt. Für Schritt zwei und drei werden servicespezifische Entscheidungsalgorithmen vorausgesetzt. Ein Service muss einen solchen Algorithmus bereitstellen. Mit Hilfe eines Anpassungsplans kann die Plattform *FraSCAti* genutzt werden, die Services zur Laufzeit ändern, neu ausrollen und neu binden kann.

Die Plattform von Romero et al. [27] macht keine Einschränkungen hinsichtlich der Anforderungen (RQ2.1) und funktioniert automatisch (RQ2.2). Es kann jeder beliebige Kontext genutzt werden, sofern die notwendigen Kontextoperatoren hinzugefügt werden (RQ2.3). Die Plattform macht keine Einschränkungen hinsichtlich der Skalierbarkeit (RQ2.4). Ein Beispiel ist vorgestellt (RQ2.5). Die Plattform beruht vollständig auf Web Service Choreographies (RQ2.6).

4. Diskussion

5. Verwandte Arbeiten

Leite et al. [2] haben eine *Systematic Literature Review* zu *Service Choreography Adaption* erstellt. Die gefundenen Arbeiten werden nach ihrer verwendeten Adaptionsstrategie kategorisiert. Ein Drittel der ausgewählten Arbeiten werden der Model-basierten Kategorie zugeordnet. Dabei werden vorzunehmende Änderungen in Modellen der zu ändernden Anwendung vorgenommen und damit auf einer höheren Abstraktionsebene als auf der Code-Ebene. Eine Änderung an einem Model ist auch gleichzeitig der Auslöser für eine Adaption, da davon ausgegangen wird, dass Nicht-Informatiker die Änderungen je nach Bedarf selbstständig manuell vornehmen können. Jeweils ein Sechstel der Arbeiten werden in die Kategorien Messung-basiert, Multi-Agent-Systeme und Formale Methoden eingeordnet. Messung-basierte Ansätze werden durch das Überschreiten von Grenzwerten ausgelöst. Die Auswahl der neuen Komposition geschieht durch Berechnung der zugrundeliegenden Werte der Alternativ-Kompositionen und einem Vergleich dieser Werte. Multi-Agent-Systeme bestehen aus unabhängig agierenden Agenten, die wenn nötig Nachrichten und damit Informationen austauschen. Die Agenten passen sich dabei selbst an, je nach Informationen, die ihnen vorliegen. Dazu wird häufig *Artificial Intelligence* genutzt. Die Kategorie der Formalen Methoden enthält Arbeiten, die auf Prozesskalkülen oder Endlichen Zustandsautomaten beruhen. Diese erlauben die Beschreibung der Services auf höheren Abstraktionsebenen. Änderungen werden ähnlich den Model-basierten Arbeiten an den Abbildungen vorgenommen und daraus automatisiert die Adaption der Services abgeleitet. Die beiden verbleibenden Kategorien mit jeweils einem Zwölftel der Arbeiten sind Semantische Ansätze und Proxy-Layer Ansätze. Bei Semantischen Ansätzen werden gezielt die logischen Relationen zwischen Services genutzt. Die Choreographie Komposition wird so angepasst, dass eine Kommunikation zwischen den Services möglich ist, sollte dies aufgrund von Änderungen durch Dritte nicht mehr möglich sein. Proxy-Layer Ansätze beruhen darauf, die Kommunikation zwischen Services zu ändern, anstatt die Komposition tatsächlich zu ändern. Dazu wird der Proxy zwischengeschaltet, alle Nachrichten laufen über den Proxy und werden wenn nötig abgeändert. Das *Systematic Literature Review* von Leite et al. [2] gibt einen Überblick zu Adaptionsstrategien von Service Choreographien. Der Grund einer Adaption, etwa aus dem Kontext heraus, wird dabei nicht beachtet.

6. Zusammenfassung

Literatur

[1] A. Dey, "Understanding and using context, personal and ubiquitous computing, vol. 5," 2001.

[2] L. A. Leite, G. A. Oliva, G. M. Nogueira, M. A. Gerosa, F. Kon, and D. S. Milojicic, "A systematic literature review of service choreography adaptation," *Service Oriented Computing and Applications*, vol. 7, no. 3, pp. 199–216, 2013.

[3] D. Budgen and P. Brereton, "Performing systematic literature reviews in software engineering," in *Proceedings of the 28th international conference on Software engineering*. ACM, 2006, pp. 1051–1052.

[4] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering–a systematic literature review," *Information and software technology*, vol. 51, no. 1, pp. 7–15, 2009.

[5] B. A. Kitchenham, T. Dyba, and M. Jorgensen, "Evidence-based software engineering," in *Proceedings of the 26th international conference on software engineering*. IEEE Computer Society, 2004, pp. 273–281.

[6] C. S. Kitchenham B, "Guidelines for performing systematic literature reviews in software engineering," Technical report, Ver. 2.3 EBSE Technical Report. EBSE, Tech. Rep., 2007.

[7] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Information and software technology*, vol. 50, no. 9-10, pp. 833–859, 2008.

[8] N. C. Narendra and S. Gundugola, "Automated context-aware adaptation of web service executions," in *IEEE International Conference on Computer Systems and Applications, 2006.* IEEE, 2006, pp. 179–187.

[9] K. Geihs, R. Reichle, M. Wagner, and M. U. Khan, "Modeling of context-aware self-adaptive applications in ubiquitous and service-oriented environments," in *Software engineering for self-adaptive systems*. Springer, 2009, pp. 146–163.

[10] B. S. Lau, C. Pham-Nguyen, C. Lee, and S. Garlatti, "Semantic web service adaptation model for a pervasive learning scenario," in *2008 IEEE Conference on Innovative Technologies in Intelligent Systems and Industrial Applications*. IEEE, 2008, pp. 98–103.

[11] M. Autili, V. Cortellessa, A. Di Marco, and P. Inverardi, "A conceptual model for adaptable context-aware services," in *International Workshop on Web Services–Modeling and Testing (WS-MaTe 2006)*, 2006, p. 15.

[12] L. Bastida, F. J. Nieto, and R. Tola, "Context-aware service composition: a methodology and a case study," in *Proceedings of the 2nd international workshop on Systems development in SOA environments*. ACM, 2008, pp. 19–24.

[13] S. Dietze, A. Gugliotta, and J. Domingue, "Towards context-aware semantic web service discovery through conceptual situation spaces," in *Proceedings of the 2008 international workshop on Context enabled source and service selection, integration and adaptation: organized with the 17th International World Wide Web Conference (WWW 2008)*. ACM, 2008, p. 6.

[14] M. Vukovic and P. Robinson, "Context aware service composition." Ph.D. dissertation, University of Cambridge, UK, 2007.

[15] J. Camara, C. Canal, and G. Salaun, "Behavioural self-adaptation of services in ubiquitous computing environments," in *2009 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 2009, pp. 28–37.

[16] C. Dorn, D. Schall, and S. Dustdar, "Context-aware adaptive service mashups," in *2009 IEEE Asia-Pacific Services Computing Conference (APSCC)*. IEEE, 2009, pp. 301–306.

[17] M. Keidl and A. Kemper, "Towards context-aware adaptable web services," in *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. ACM, 2004, pp. 55–65.

[18] S. B. Mokhtar, D. Fournier, N. Georgantas, and V. Issarny, "Context-aware service composition in pervasive computing environments," in *International Workshop on Rapid Integration of Software Engineering Techniques*. Springer, 2005, pp. 129–144.

[19] N. Luo, J. Yan, M. Liu, and S. Yang, "Towards context-aware composition of web services," in *2006 Fifth International Conference on Grid and Cooperative Computing (GCC'06)*. IEEE, 2006, pp. 494–499.

[20] H. J. La and S. D. Kim, "A conceptual framework for provisioning context-aware mobile cloud services," in *2010 IEEE 3rd International Conference on Cloud Computing*. IEEE, 2010, pp. 466–473.

[21] Z. Cao, X. Zhang, W. Zhang, X. Xie, J. Shi, and H. Xu, "A context-aware adaptive web service composition framework," in *2015 IEEE International Conference on Computational Intelligence & Communication Technology*. IEEE, 2015, pp. 62–66.

[22] M. Madkour, D. El Ghanami, and A. Maach, "Context-aware service adaptation: An approach based on fuzzy sets and service composition," in *Journal of Information Science and Engineering*. JISE, 2013, pp. 1–16.

[23] J. Yu, J. Han, Q. Z. Sheng, and S. O. Gunarso, "Percas: An approach to enabling dynamic and personalized adaptation for context-aware services," in *International Conference on Service-Oriented Computing*. Springer, 2012, pp. 173–190.

[24] G. H. Alferez and V. Pelechano, "Context-aware autonomous web services in software product lines," in *2011 15th International Software Product Line Conference*. IEEE, 2011, pp. 100–109.

[25] S. Hallsteinsen, M. Hinchey, S. Park, and K. Schmid, "Dynamic software product lines," in *Computer*. IEEE, 2008, pp. 93–95.

[26] J. Zhou, E. Gilman, J. Palola, J. Riekki, M. Ylianttila, and J. Sun, "Context-aware pervasive service composition and its implementation," in *Personal and Ubiquitous Computing*. Springer, 2011, pp. 291–303.

[27] D. Romero, R. Rouvoy, L. Seinturier, S. Chabridon, D. Conan, and N. Pessemier, "Enabling context-aware web services: A middleware approach for ubiquitous environments," in *Enabling Context-Aware Web Services: Methods, Architectures, and Technologies*. Chapman and Hall/CRC, 2010, pp. 113–135.

[28] R. Rouvoy, D. Conan, and L. Seinturier, "Software architecture patterns for a context-processing middleware framework," in *IEEE Distributed Systems Online*. IEEE, 2008.

[29] L. Seinturier, P. Merle, D. Fournier, N. Dolet, V. Schiavoni, and J.-B. Stefani, "Reconfigurable sca applications with the frascati platform," in *2009 IEEE International Conference on Services Computing*, 2009, pp. 268–275.