Universität Stuttgart

Anforderungsbasierte Modellierung und Ausführung von Datenflussmodellen

Von der Fakultät für Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Dipl.-Inf. Pascal Hirmer

aus Sindelfingen

Hauptberichter: Prof. Dr. Bernhard Mitschang

Mitberichter: Prof. Dr. Florian Daniel

Tag der mündlichen Prüfung: 15.06.2018

Institut für Parallele und Verteilte Systeme

2018

Inhaltsverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
AWS	Amazon Web Services
BPEL	Business Process Execution Language
BPMN	Business Model and Notation
CEP	Complex Event Processing
CRUD	Create-Retrieve-Update-Delete
CSAR	Cloud Service Archive
DBMS	Datenbank-Management-System
DPM	Datenprozessierungsmodell
DVM	Datenverarbeitungsmodell
DVM$^+$	Datenverarbeitungsmodell$^+$
ETG	Enterprise Topology Graph
ETL	Extraction, Transformation, Load
ER-Diagramm	Entitäten-Relationen-Diagramm
GUI	Grafische Benutzerschnittstelle
HATEOAS	Hypermedia As The Engine Of Application State
HDFS	Hadoop Distributed File System
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service

IT	Informationstechnik
IoT	Internet of Things (dt.: Internet der Dinge)
JAR	Java Archive
JDK	Java Development Toolkit
JAXB	Java Architecture for XML Binding
JSON	JavaScript Object Notation
KDD	Knowledge Discovery in Databases
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
OASIS	Organization for the Advancement of Structured Information Standards
OWL	Web Ontology Language
PaaS	Platform as a Service
PHP	PHP Hypertext Preprocessor
RAM	Random Access Memory
REST	Representational State Transfer
RMP	Ressourcen-Management-Plattform
RSS	Rich Site Summary / Really Simple Syndication
SaaS	Software as a Service
SOA	Serviceorientierte Architektur
SOC	Serviceorientiertes Computing
SSH	Secure Shell
SQL	Structured Query Language
TOSCA	Topology and Orchestration Specification for Cloud Applications
UDDI	Universal Description, Discovery and Integration
UI	Benutzerschnittstelle
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	Extensible Markup Language
XSD	XML Schema Definition
WAR	Web Archive

WS	Web Service
WS-BPEL	Web Services Business Process Execution Language
WSDL	Web Services Description Language

Zusammenfassung

Heutzutage steigen die Menge an Daten sowie deren Heterogenität, Änderungshäufigkeit und Komplexität stark an. Dies wird häufig als das "Big-Data-Problem" bezeichnet. Durch das Aufkommen neuer Paradigmen, wie dem Internet der Dinge oder Industrie 4.0, nimmt dieser Trend zukünftig noch weiter zu. Die Verarbeitung, Analyse und Visualisierung von Daten kann einen hohen Mehrwert darstellen, beispielsweise durch die Erkennung bisher unbekannter Muster oder durch das Vorhersagen von Ereignissen. Jedoch stellen die Charakteristiken von Big-Data, insbesondere die große Datenmenge und deren schnelle Änderung, eine große Herausforderung für die Verarbeitung der Daten dar. Herkömmliche, bisher angewandte Techniken, wie zum Beispiel Analysen basierend auf relationalen Datenbanken, kommen hierbei oft an ihre Grenzen. Des Weiteren ändert sich auch die Art der Anwender der Datenverarbeitung, insbesondere in Unternehmen. Anstatt die Datenverarbeitung ausschließlich von Programmierexperten durchzuführen, wächst die Anwendergruppe auch um Domänennutzer, die starkes Interesse an Datenanalyseergebnissen haben, jedoch diese nicht technisch umsetzen können. Um die Unterstützung von Domänennutzern zu ermöglichen, entstand ca. im Jahr 2007, im Rahmen der Web-2.0-Bewegung, das Konzept der Mashups, die es auf einfachem Wege erlauben sollen, Anwender aus unterschiedlichen Domänen beim Zusammenführen von Programmen,

grafischen Oberflächen, und auch Daten zu unterstützen. Hierbei lag der Fokus vor allem auf Webdatenquellen wie RSS-Feeds, HTML-Seiten, oder weiteren XML-basierten Formaten. Auch wenn die entstandenen Konzepte gute Ansätze liefern, um geringe Datenmengen schnell und explorativ durch Domänennutzer zu verarbeiten, können sie mit den oben genannten Herausforderungen von Big-Data nicht umgehen. Die Grundidee der Mashups dient als Inspiration dieser Dissertation und wird dahingehend erweitert, moderne, komplexe und datenintensive Datenverarbeitungs- und Analyseszenarien zu realisieren. Hierfür wird im Rahmen dieser Dissertation ein umfassendes Konzept entwickelt, das sowohl eine einfache Modellierung von Datenanalysen durch Domänenexperten ermöglicht – und somit den Nutzer in den Mittelpunkt stellt – als auch eine individualisierte, effiziente Ausführung von Datenanalysen und -verarbeitung ermöglicht. Unter einer Individualisierung wird dabei verstanden, dass die funktionalen und nichtfunktionalen Anforderungen, die je nach Anwendungsfall variieren können, bei der Ausführung berücksichtigt werden. Dies erfordert einen dynamischen Aufbau der Ausführungsumgebung. Hierbei wird dem beschriebenen Problem durch mehrere Ebenen begegnet: 1) Die Modellierungsebene, die als Schnittstelle zu den Domänennutzern dient und die es erlaubt Datenverarbeitungsszenarien abstrakt zu modellieren. 2) Die Modelltransformationsebene, auf der das abstrakte Modell auf verschiedene ausführbare Repräsentationen abgebildet werden kann. 3) Die Datenverarbeitungsebene, mit der die Daten effizient in einer verteilten Umgebung verarbeitet werden, und 4) die Datenhaltungsebene, in der Daten heterogener Quellen extrahiert sowie Datenverarbeitungs- oder Analyseergebnisse persistiert werden. Die Konzepte der Dissertation werden durch zugehörige Publikationen in Konferenzbeiträgen und Fachmagazinen gestützt und durch eine prototypische Implementierung validiert.

Abstract

Nowadays, the volume of data as well as their velocity and veracity increases. This is oftentimes referred to as the “Big Data” problem. Due to the upcoming paradigms Internet of Things or Industrie 4.0, this trend even increases in the future. Analytics and visualization of this data could lead to high benefits, for example, through the recognition of previously unknown patterns or the prediction of occurring events. However, the characteristics of Big Data, especially volume and velocity, pose a great challenge. Traditional techniques, such as static analytics based on relational databases, oftentimes cannot cope with these issues due to limitations, for example, regarding scalability. Furthermore, the kind of users of data analytics and visualization is shifting from programming experts to domain users that are highly interested in analysis results, however, cannot implement these analyses themselves. To cope with this issue, circa in 2007, as part of the Web 2.0 movement, the concept of Mashups appeared that allow domain users to integrate program logic, graphical user interfaces, and data through graphical modeling. Concerning data processing, these approaches were focused on web data, such as RSS feeds, HTML websites, or other XML-based formats. Although these approaches offer interesting ideas to process small amounts of data in a fast and explorative manner, they cannot cope with the complex characteristics of Big Data. However, the basic idea of

Mashups serves as inspiration for this PhD thesis and is being extended by means to realize modern, complex, and data-intensive data processing and analytics scenarios. To realize this, a holistic concept is presented in this PhD thesis that enables abstract modeling of data analytics and data processing by domain experts – i.e., a concept that puts the user in the center – as well as a personalized, efficient execution of data analytics and data processing scenarios. Personalization means that the various functional and nonfunctional requirements of different use case scenarios are considered. This requires a dynamic construction of the execution environment. The mentioned challenges are coped with by a four-layered approach: 1) The modeling layer, serving as interface to the users to model data processing and analysis scenarios. 2) The model transformation layer that transforms the abstract model into executable representations. 3) The data processing layer, on which data is processed efficiently in a distributed environment, and 4) the data storage layer, in which data of heterogeneous sources are extracted and analytic results are persistently stored. The concepts of this PhD thesis are supported by corresponding publications in conference proceedings and journal articles and are validated by a prototypical implementation.

Danksagungen

Einen herzlichen Dank an alle, die mich bei der Erstellung meiner Dissertation unterstützt haben. Zuerst möchte ich meinem Doktorvater Prof. Dr. Bernhard Mitschang für seine hervorragende fortwährende Unterstützung und sein Vertrauen in meine Arbeit danken. Außerdem möchte ich mich bei Prof. Dr. Florian Daniel bedanken, der sich bereit erklärt hat die Rolle des Mitberichters zu übernehmen sowie bei den weiteren Mitgliedern des Prüfungskommittees, Vorsitzender Prof. Dr. Frank Leymann und Mitprüfer Prof. Dr. Ralf Küsters. Mein Dank gilt auch allen Kollegen der Abteilung AS des IPVS, mit denen ich stets gehaltvolle Diskussionen führen konnte, die die Resultate dieser Dissertation anreicherten. Insbesondere bedanke ich mich bei meinem ehemaligen Zimmerkollegen und Post-Doc Dr. Matthias Wieland sowie bei Dr. Peter Reimann, Ana Cristina Franco da Silva und Mathias Mormul, die mir stets wertvolles Feedback geben konnten. Des Weiteren möchte ich mich bei allen Personen bedanken, mit denen ich zahlreiche wissenschaftliche Publikationen verfassen durfte sowie bei den Studierenden, die mich bei der Erstellung von Prototypen meiner Konzepte unterstützt haben. Zu guter Letzt gilt mein Dank meinen Freunden und meiner Familie, die mir stets mit viel Geduld zur Seite standen.

Kapitel 1

Einführung

In diesem Kapitel wird eine Einführung in die Dissertation gegeben. Dabei werden zuerst die Forschungsfragen und Ziele der Dissertation, deren Herausforderungen, und der Stand bisheriger Arbeiten beschrieben. Anschließend wird eine Übersicht über die Beiträge gegeben sowie der Aufbau der Arbeit beschrieben.

1.1 Motivation

Unter dem Begriff *Big-Data* wird die seit einigen Jahren immer weiter ansteigende Komplexität erzeugter oder vorhandener Datenmengen beschrieben [MBD+12; MCB+11]. Die hohe Komplexität entsteht vor allem durch hohes Datenvolumen, hohe Datenvielfalt sowie eine häufige Änderung der Daten. Der hohe Anstieg an Daten ist auf die in den letzten Jahren stark anwachsende Digitalisierung und somit auf hinzugekommene Hardwaregeräte und Softwareprogramme zurückzuführen, die sich durch beinahe alle Domänen zieht (z.B. Produktion, Medizin, Verkehr). Das aufstrebende *Internet der Dinge* [GBMP13; VF14] verstärkt diesen Effekt zukünftig noch weiter. Bereits im Jahr 2020 sollen weltweit zwischen 10 und 15 Milliar-

den Geräte, wie zum Beispiel Smart-Phones, Tablets oder Mini-Computer, verbunden über eine Internetverbindung, existieren [Nor16]. Diese Geräte produzieren laufend Daten, die einerseits aus Softwareprogrammen stammen, andererseits aber auch von in die Geräte eingebetteten Sensoren, die Standortdaten, Wetterdaten oder, durch Wearables [PJ03], biometrische Daten, wie Herzschlag und Blutdruck von Menschen, erfassen können.

Die dadurch entstehende große Menge an Daten enthält oftmals nützliche Informationen, die einen hohen Mehrwert darstellen können. Beispielsweise ist es mittlerweile üblich, dass große Internetfirmen wie Google Daten erfassen und analysieren, um den Anwendern ein individualisiertes, auf ihre Bedürfnisse zugeschnittenes, Produkt anbieten zu können. Ein bekanntes Beispiel ist auch der Online-Händler Amazon, der individuelle Vorschläge basierend auf der Analyse des Kaufverhaltens seiner Kunden ermöglicht. Derartige Analysen werden in der Regel nach aufwendigen Untersuchungen von IT-Abteilungen solcher großen Firmen implementiert, erprobt und eingesetzt. Die Extraktion von Wissen durch Analysen erfordert die Durchführung der Schritte Datenauswahl, Datenbereinigung, Datentransformation, Data-Mining, und Interpretation der Ergebnisse. Diese Schritte wurden im sogenannten *Knowledge-Discovery-Prozess* von Fayyad et al. [FPS96] definiert. Insbesondere durch die komplexen Eigenschaften von Big-Data – Datenvolumen, Datenvielfalt und Änderungshäufigkeit – stellen diese Schritte jedoch selbst für erfahrene Entwickler eine große Herausforderung dar, da die einzelnen Schritte des Prozesses dadurch sehr komplex werden können. Handelt es sich zusätzlich um strombasierte Daten, also Datensätze, die sich oft ändern und dadurch schwer zu verarbeiten sind, werden zusätzlich fortgeschrittene Methoden und Ansätze, wie beispielsweise *Complex-Event-Processing* [Luc01], *Stream-Processing* [GR07] oder *Data-Stream-Mining* [GZK10], benötigt. Für die Ausführung der Datenverarbeitung wurden in der Vergangenheit eine Vielzahl an Datenverarbeitungssystemen geschaffen.

Aufgrund dessen, dass sich die funktionalen und nichtfunktionalen Anforderungen an die Datenverarbeitung je nach Anwendungsfall stark unterscheiden (z.B. Effizienzanforderungen, Kosten, Sicherheit), ist es wichtig, dass sich Datenverarbeitungssysteme dynamisch anpassen können. Dabei

sollten Techniken und Technologien zur Anwendung kommen, die die spezifischen Anforderungen eines Anwendungsfalls erfüllen. Bisherige Lösungen sind jedoch sehr statisch und beschränkt auf wenige, vordefinierte Verarbeitungstechniken, d.h., sie können mit der erforderlichen Dynamik nicht umgehen. Derartige statische Ansätze führen dazu, dass sich ändernde oder neu auftretende Anforderungen nicht bedient werden können. Nichtfunktionale Anforderungen werden von den meisten Systemen nicht beachtet. Im schlimmsten Fall müssen für verschiedene Anwendungsfälle unterschiedliche Datenverarbeitungssysteme verwendet werden. Der Fokus der vorliegenden Dissertation liegt daher darauf, mit diesem Problem umzugehen, indem eine flexible, anwendungsfallabhängige Datenverarbeitung ermöglicht wird.

Darüber hinaus wächst seit einigen Jahren der Bedarf daran, Datenverarbeitung und -analyse von Anwendern außerhalb der IT-Domäne durchzuführen. Dadurch werden die entstehenden Vorteile für einen breiteren Anwenderkreis verfügbar gemacht. Das kann beispielsweise eine große Kostenersparnis für Unternehmen durch die Einsparung von IT-Expertise bei der Datenverarbeitung bedeuten.

Erste Ansätze um eine Abstraktion technischer Details für Domänennutzer bei der Datenverarbeitung und -analyse zu ermöglichen wurden ca. im Jahr 2007 im Rahmen der Web-2.0-Bewegung [Lew06; ORe05] geschaffen. Dabei wurde der Begriff *"Mashup"* [DM14] geprägt, ein Überbegriff für das Zusammenführen von Programmen, grafischen Oberflächen, und auch Daten verschiedener, oftmals verteilter, Quellen. Hierbei stand die Anwendbarkeit durch Nutzer außerhalb des IT-Bereichs im Vordergrund. Das Hauptaugenmerk lag auf dem Prinzip des *grafischen Programmierens* sowie der Idee, gewünschte Ergebnisse durch eine *explorative* Vorgehensweise zu erlangen. Das heißt, über die Bedienung durch grafische Elemente, wie eine Modellierung mittels *Drag & Drop*, sollte die Möglichkeit entstehen, in kurzer Zeit Modelle zu schaffen, die die Zusammenführung von Programmelementen (Programmlogik, Benutzerschnittstellen, Daten) definieren. Deren Ausführung und eine anschließende Betrachtung der Ergebnisse kann dann wiederum zu Anpassungen im Ursprungsmodell führen. Dieses explorative Vorgehen wiederholt sich bis das gewünschte Ergebnis eintritt.

Hierbei entstanden auch schon erste Ansätze, um Daten zu verarbeiten und zusammenzuführen. Diese Ansätze wurden unter dem Begriff *Data-Mashup* zusammengefasst. Hierbei wurden grafische Datenflussmodelle geschaffen, die definieren, in welcher Art und in welcher Reihenfolge Daten verarbeitet werden sollen. Die Knoten dieser Modelle repräsentieren Datenoperationen, wie zum Beispiel Datenextraktion oder Filterung, die Kanten die Reihenfolge deren Ausführung. Das bekannteste Beispiel hierfür ist die mittlerweile nicht mehr verfügbare Web-Anwendung *Yahoo Pipes!* [Pru07].

Der Fokus lag dabei jedoch auf Datenquellen aus dem Web, wie zum Beispiel RSS-Feeds, HTML-Dateien oder andere XML-basierte Formate. Hierbei wurden in den Data-Mashup-Ansätzen vor allem das Filtern und Aggregieren von Daten unterstützt, komplexe Datenanalysen jedoch nicht. Nachdem Web-2.0-Mashups für einige Zeit hohe Popularität erlangten, verschwanden sie bis Anfang 2010 fast vollständig.

Die Grundidee von Mashups, Nutzern über grafische Elemente die Kontrolle über die Datenverarbeitung zu übertragen, dient als Inspiration der vorliegenden Dissertation. Dabei liegt der Fokus im Gegensatz zu Web-2.0-Mashups jedoch auf der Verarbeitung, Integration, und Analyse komplexer Datenmengen. In dieser Dissertation wird auf dem Grundgedanken der Mashups, insbesondere auf der grafischen Modellierung von Datenflussmodellen sowie der explorativen Vorgehensweise, aufgebaut. Es soll also keine Weiterentwicklung von traditionellen Mashup-Konzepten entstehen, sondern ein moderner, *Mashup-inspirierter* Ansatz zur Verarbeitung und Analyse von Daten durch Domänennutzer. Hierdurch kann aufgrund von Kostenersparnis ein hoher Mehrwert für verschiedene Domänen geschaffen werden.

Zusammenfassend wird im Rahmen der vorliegenden Dissertation ein gesamtheitliches Konzept vorgestellt, um Daten basierend auf Nutzeranforderungen zu verarbeiten. Dabei steht die Bedienbarkeit durch Domänennutzer im Vordergrund. Alle im Folgenden beschriebenen Konzepte basieren auf etablierten Standards oder de-facto Standards. Die Beiträge dieser Dissertation wurden auf nationalen und internationalen Konferenzen, in Fachmagazinen, und Buchkapiteln veröffentlicht. Eine Übersicht der Publikationen befindet sich am Ende des Dokuments (S. 229 ff.).

1.2 Forschungsfragen und Ziele der Dissertation

Führt man die beschriebenen Problemstellungen, (1) das Verarbeiten und Analysieren komplexer Daten unter Berücksichtigung funktionaler und nichtfunktionaler Anforderungen und (2) die Unterstützung von Domänennutzern, zusammen, ergibt sich ein komplexes Problem, das verschiedenartige Lösungskonzepte erfordert.

Die resultierenden Forschungsfragen sind:

- Wie können unterschiedliche funktionale und nichtfunktionale Anforderungen berücksichtigt werden, um eine anwendungsfallabhängige Datenverarbeitung zu ermöglichen?
- Wie kann es Domänennutzern ermöglicht werden, komplexe Datenanalysen zu erstellen, ohne spezielle IT-Kenntnisse vorauszusetzen?
- Wie kann ein automatisierter, dynamischer Aufbau der Ausführungsumgebung ermöglicht werden?
- Wie kann mit den Charakteristiken von Big-Data (Volumen, Vielfalt, Änderungshäufigkeit) umgegangen werden, die die Komplexität der Datenverarbeitung und Datenanalyse erhöhen?
- Wie können dynamische, sich oft ändernde Umgebungen, beispielsweise im Bereich Internet der Dinge oder Industrie 4.0, bestmöglich bei der Datenverarbeitung unterstützt werden?

Die aufgeführten Fragestellungen werden im Rahmen der Dissertation behandelt und führen zur Definition der folgenden resultierenden Ziele:

1.2.1 Ziel 1: Anforderungsabhängige Auswahl der Ausführungsumgebung

Abhängig vom Anwendungsfall gelten verschiedene funktionale und nichtfunktionale Anforderungen an die Datenverarbeitung. Während in manchen Szenarien eine möglichst effiziente Ausführung der Datenverarbeitung wichtig ist, sind in anderen Szenarien beispielsweise die Robustheit, Sicherheit,

oder möglichst geringe Kosten von hoher Wichtigkeit. Bestehende Ansätze bieten eine derartige Flexibilität nicht. Daher ist es notwendig neuartige Konzepte zu schaffen, um unterschiedliche Anforderungen dynamisch, abhängig vom Anwendungsfall, erfüllen zu können.

Das Ziel ist daher eine Individualisierung der Datenverarbeitung, zugeschnitten auf die Bedürfnisse der Domänennutzer sowie auf verschiedenartige Anwendungsszenarien. Abhängig von den funktionalen und nichtfunktionalen Anforderungen des spezifischen Anwendungsfalls soll die Ausführungsumgebung dynamisch, baukastenartig zusammengestellt werden. Dabei ist es wichtig, dass auch bei der Definition der Anforderungen technische Details für Domänennutzer verborgen werden, zum Beispiel spezielle Verschlüsselungstechniken zur Erhöhung der Sicherheit.

1.2.2 Ziel 2: Unterstützung von Domänennutzern bei der Erstellung komplexer Datenanalysen

Die Durchführung von komplexen Datenanalysen stellt für Domänennutzer außerhalb des Bereichs der Datenverarbeitung oder sogar außerhalb des IT-Bereichs eine große Herausforderung dar. Dies erfordert tiefes Wissen über die Daten selbst sowie über deren enthaltenes Potential bezüglich Datenanalysen. Die Entscheidung wie Daten extrahiert, vorverarbeitet und analysiert werden sollten, um die angestrebten Analyseziele zu erreichen, kann von Domänennutzern oftmals nicht getroffen werden. Eines der Hauptziele dieser Dissertation ist es daher, die Domänennutzer bei der Durchführung der Datenverarbeitung und -analyse durch neuartige Konzepte zu unterstützen.

Hierfür soll es ermöglicht werden, die Datenverarbeitung vor der Ausführung abstrakt zu modellieren, ohne dabei besondere Kenntnisse über Datenquellen, Datenformate oder Analysealgorithmen zu benötigen. Durch mehrfache Ausführung des Modells kann das gewünschte Ergebnis explorativ angenähert werden. Darüber hinaus soll der Modellierer bestmöglich bei der Modellierung unterstützt werden, beispielsweise durch Vorschlagsgenerierung bei der Auswahl von durchzuführenden Datenverarbeitungsoperationen. Neben der Unterstützung von Domänennutzern ohne jegliche Kenntnisse

der Datenverarbeitung sollen darüber hinaus auch Nutzer mit Grundkenntnissen oder tieferen Kenntnissen der Datenverarbeitung bestmöglich dabei unterstützt werden ihr Wissen einzubringen.

1.2.3 Ziel 3: Automatisierter Aufbau der Ausführungsumgebung

Die im ersten Ziel angestrebte anforderungsabhängige Auswahl der Ausführungsumgebung basierend auf funktionalen und nichtfunktionalen Anforderungen erfordert es, die Ausführungsumgebung dynamisch aufzubauen. Der Aufbau der Ausführungsumgebung sollte möglichst (voll-)automatisiert geschehen, um die Kosten gering zu halten. Wurden die Anforderungen an die Datenverarbeitung definiert, müssen passende, die Anforderungen erfüllende, Softwarekomponenten aufgefunden werden. Diese Komponenten können einerseits bereits verfügbar sein, beispielsweise bereitgestellt durch einen Platform-as-a-Service-Anbieter, oder sie müssen erst noch bereitgestellt werden. In diesem Fall sollen etablierte Softwareprovisionierungstechnologien zum Einsatz kommen, um die Komponenten aufzusetzen. Wurde die Ausführungsumgebung bereitgestellt, kann die Datenverarbeitung in dieser, unter Erfüllung der Anforderungen, durchgeführt werden.

1.2.4 Ziel 4: Effektive Datenverarbeitung

Die Charakteristiken von Big Data stellen eine große Herausforderungen bei der Datenverarbeitung dar. Daher soll ein Konzept geschaffen werden, um mit diesen bestmöglich umzugehen. Hierfür sollen insbesondere Konzepte geschaffen werden, um mit hohem Datenvolumen, Geschwindigkeit und Änderungshäufigkeit der Daten umgehen zu können. Dadurch wird die Umsetzung moderner, datenintensiver Szenarien ermöglicht, beispielsweise in den Bereichen Internet der Dinge oder Industrie 4.0. Hierfür sollen moderne Techniken und Technologien, beispielsweise zur verteilten Datenverarbeitung, kombiniert werden.

1.2.5 Ziel 5: Unterstützung dynamischer Umgebungen

Dynamische Umgebungen sind dadurch charakterisiert, dass oft neue Datenquellen hinzukommen oder Bestehende wegfallen. Dies kann beispielsweise durch die Durchführung von Simulationen und die Bereitstellung deren Ergebnissen geschehen oder durch einen Ausfall oder die absichtliche Abschaltung eines Systems. Besonders die Hinzunahme neuer Datenquellen kann zu einem verbesserten Ergebnis der Datenverarbeitung und -analyse führen. Darüber hinaus stellt der Wegfall von Datenquellen eine große Herausforderung dar, da es dadurch oftmals zu Systemabstürzen kommt.

Eines der Ziele dieser Dissertation ist es daher, mit dynamischen Umgebungen umgehen zu können. Es soll eine Möglichkeit geschaffen werden, um neue Datenquellen zur Datenverarbeitung und -analyse hinzuzuziehen oder möglichst robust auf wegfallende Datenquellen reagieren zu können. Hierfür sollen neuartige Konzepte für eine Middleware geschaffen werden, mit denen dynamisch, möglichst vollautomatisch, auf das Hinzukommen und Wegfallen von Datenquellen reagiert werden kann.

1.3 Herausforderungen und Stand bisheriger Arbeiten

Bei der Realisierung der in Abschnitt 1.2 beschriebenen Ziele treten einige Herausforderungen auf, die in diesem Abschnitt diskutiert werden. Neben der Beschreibung der Herausforderungen werden außerdem bisherige Ansätze vorgestellt, die das Ziel haben mit diesen umzugehen. Die Herausforderungen wurden durch eine umfassende Literaturrecherche basierend auf den Zielen und Forschungsfragen identifiziert und umfassen: (1) unterschiedliche Bedürfnisse von Domänennutzern, (2) Anforderungsdefinition, (3) automatische Bereitstellung von Softwarekomponenten, (4) Charakteristiken von Big-Data, und (5) Umgang mit dynamischen Umgebungen.

1.3.1 Unterschiedliche Bedürfnisse von Domänennutzern

Eine der Herausforderungen stellen die unterschiedlichen Bedürfnisse von Domänennutzern dar. Dies liegt daran, dass sich die Vorkenntnisse von Domänennutzern sehr stark unterscheiden. Häufig wird bei bekannten Softwareprodukten, wie zum Beispiel IBM SPSS [IBMa], zwischen zwei Typen von Anwendern unterschieden – *Casual User* und *Power User*. Nutzer mit geringem oder wenig technischem Verständnis werden als Casual Users bezeichnet, technische Experten hingegen als Power User, denen umfangreichere Konfigurationsmöglichkeiten geboten werden.

Im Gegensatz hierzu wird im Rahmen der vorliegenden Dissertation zwischen drei Typen von Domänennutzern unterschieden, um eine feingranularere Unterscheidung zu ermöglichen. Diese sind:

Definition 1.1 (Domänennutzer ohne jegliche IT-Vorkenntnisse)
Dabei handelt es sich um Anwender, die keine Erfahrungen mit der Entwicklung von Software, Softwareanalysen, oder der Verarbeitung von Daten haben und auch keine Vorkenntnisse in Statistik besitzen. Meist handelt es sich dabei um Personen mit sehr spezifischem Domänenwissen, wie zum Beispiel Ärzte. Diese Anwendergruppe ist daran interessiert, Nutzen aus anfallenden Datenmengen zu ziehen und hat auch Vorstellungen, welche Informationen in den Daten enthalten sein könnten, jedoch nicht die technischen Mittel, um derartige Datenanalysen zu erstellen und auszuführen.

Definition 1.2 (Domänennutzer mit Kenntnissen aus der Statistik)
Diese Anwendergruppe besitzt umfassende Kenntnisse aus dem Bereich Mathematische Statistik und kennt daher statistische Methoden, um Wissen aus Daten zu generieren. Gute Programmierkenntnisse oder Kenntnisse über Algorithmik besitzt diese Gruppe von Anwendern nicht. Das bedeutet, dass zwar die Anwendung und konkrete Parametrisierung von statistischen Methoden oder Data-Mining-Algorithmen durchgeführt werden können, aber keine Datenanalysen selbstständig implementiert werden können.

Definition 1.3 (Domänennutzer mit Kenntnissen der Programmierung) *Die letzte Gruppe von Domänennutzer besitzt umfassende Kenntnisse im Bereich Programmierung und Statistik. Dadurch ist es dieser Gruppe nicht nur möglich, Datenanalysen detailliert zu parametrisieren, sondern sie sind auch in der Lage, eigene Algorithmen zu implementieren, um maßgeschneiderte Analysen zu schaffen. Für die Durchführung der Datenverarbeitung ist diese Gruppe am besten geeignet, da sie umfassendes Wissen aus dem Bereich Statistik, Daten, Datenanalysen, und zusätzlich, Programmierkenntnisse besitzen. Jedoch stellt diese Anwendergruppe eher die Seltenheit dar, da die Erlangung von derartigem Wissen mehrjährige Erfahrung erfordert.*

Von allen Typen von Domänennutzern wird vorausgesetzt, dass sie umfassendes Wissen über ihre Domäne (z.B. Health, Produktion, Business) besitzen und die Bedeutung der Daten sowie deren Potential bezüglich möglicher Analysen bzw. enthaltener Informationen kennen. Diese Anwendergruppen verfolgen dabei jedoch unterschiedliche Ziele bei der Verarbeitung von Daten. Beispielsweise ist es für einen Domänennutzer mit Kenntnissen der Programmierung und Statistik wichtig, dass er die Analysen möglichst detailliert parametrisieren oder sogar eigenen Programmcode, zum Beispiel bisher nicht verfügbarer Algorithmen, mit einbringen kann. Für einen Domänennutzer ohne jegliche Statistik- und IT-Kenntnisse hingegen ist es wichtig, dass möglichst wenig konfiguriert werden muss, um auf die gewünschten Ergebnisse zu kommen, auch wenn diese anfänglich nicht die Präzision liefern, die durch eine detaillierte Parametrisierung möglich ist. Hier steht vor allem die explorative Vorgehensweise im Vordergrund, bei der die Ergebnisse durch oftmaliges Ausführen der Datenanalysen und einer anschließenden Rekonfiguration kontinuierlich verbessert werden. Jeder der Anwendergruppen sollte es ermöglicht werden, eine individualisierte, den eigenen Bedürfnissen zugeschnittene, Durchführung der Datenverarbeitung zu ermöglichen.

Im Folgenden wird zwischen *IT-Experten* und *Domänennutzern* unterschieden. Dabei sind Experten Nutzer mit umfangreichem technischem IT-Wissen, ohne jedoch Wissen über die Domäne zu besitzen. Als Domänennutzer werden die oben beschriebenen Rollen bezeichnet (vgl. Definitionen 1.1 bis 1.3).

1.3.2 Anforderungsdefinition

In bestehenden Ansätzen zur Datenverarbeitung stehen funktionale Anforderungen im Vordergrund. Dabei gibt es eine Vielzahl von Möglichkeiten, um Daten zu verarbeiten, zu analysieren, abzuspeichern oder zu visualisieren. Nichtfunktionale Anforderungen finden dabei jedoch kaum Beachtung. Insbesondere die Kosten, Effizienz, oder Sicherheit bei der Verarbeitung von Daten sind in der Praxis jedoch von großer Wichtigkeit. Beispielsweise garantieren viele Unternehmen, dass sensible Kundendaten sicher verarbeitet werden, beispielsweise unter Verwendung von Verschlüsselungstechniken. Diese Garantie muss bei der Verarbeitung der Daten eingehalten werden. Darüber hinaus müssen auch Anforderungen an die Kosten beachtet werden. Zum Beispiel führt eine verteilte Datenverarbeitung auf einer angemieteten Recheninfrastruktur zu hohen Kosten, wohingegen eine lokale Datenverarbeitung vergleichsweise preisgünstig durchgeführt werden kann.

Die Definition von funktionalen und nichtfunktionalen Anforderungen stellt besonders für Domänennutzer eine große Herausforderung dar, da sie keine Kenntnisse, beispielsweise über bestimmte Verschlüsselungstechniken, besitzen. Daher muss eine Möglichkeit geschaffen werden, um Anforderungen abstrakt definieren zu können, ohne spezifisches technisches Detailwissen vorauszusetzen, was eine große Herausforderung darstellt. Darüber hinaus muss es auch Experten mit einem derartigen Wissen ermöglicht werden, die Anforderungen möglichst exakt festlegen zu können.

1.3.3 Automatische Bereitstellung von Softwarekomponenten

Sind die Anforderungen definiert, soll die Ausführungsumgebung dynamisch, unter deren Berücksichtigung, aufgebaut werden. Beispielsweise erfordert die Erfüllung bestimmter Effizienzanforderungen oftmals eine Verteilung der Datenverarbeitung in einem Rechencluster oder die Erfüllung von Sicherheitsanforderungen Softwaremodule zur Verschlüsselung bzw. eine Schlüsselverwaltungskomponente. Diese Komponenten sollen dynamisch, baukastenartig aufgesetzt werden, um die Ausführungsumgebung zu bilden.

Das automatisierte Aufsetzen passender Komponenten zum Erfüllen der Anforderungen stellt dabei eine der Hauptherausforderungen dar.

Um Softwarekomponenten bereitzustellen wurden in bisherigen Arbeiten Lösungen zur Softwareprovisionierung geschaffen, mit denen Software dynamisch, vollautomatisiert bereitgestellt werden kann. Dabei wird in einem ersten Schritt ein Modell der Ausführungsumgebung geschaffen, entweder mittels Skriptsprachen (vgl. Vagrant [Has], Docker [Doc]) oder durch ein grafisches Modell (vgl. TOSCA [OAS13a] bzw. Vino4TOSCA [BBK+12]). Das Modell wird anschließend für das Aufsetzen der Umgebung genutzt. Eine bisher nicht gelöste Herausforderung ist es jedoch, derartige Modelle automatisch zu erzeugen, um menschliche Interaktion und dadurch die Kosten des Aufsetzens zu reduzieren.

1.3.4 Charakteristiken von Big-Data

Die Charakteristiken von Big-Data stellen eine große Herausforderung für deren Verarbeitung dar. Sie umfassen nicht nur, wie oftmals angenommen, das große Datenvolumen, sondern darüber hinaus weitere Faktoren, die die Datenverarbeitung zusätzlich erschweren. Tatsächlich kann mit dem hohen Datenvolumen durch die Fortschritte im Bereich *Cloud-Computing* [MG+11] und der verteilten Datenverarbeitung, beispielsweise mittels Apache Spark [Spa15], noch am besten umgegangen werden.

Im Gegensatz dazu stellt die Geschwindigkeit der Daten, also deren Frequenz der Erzeugung sowie deren hohe Änderungsrate, eine große Herausforderung dar. Viele herkömmliche Ansätze führen Datenanalysen und Datenverarbeitungsoperationen basierend auf statischen Datenmengen durch. Die Daten sind dabei beispielsweise in einem *Data-Warehouse* [Dev96] persistiert. Jedoch reicht ein derartiger Ansatz oftmals nicht mehr aus, da Daten, beispielsweise erzeugt von Sensoren, oftmals strombasiert in hoher Frequenz erzeugt werden. Würden diese strombasierten Daten erst persistiert werden, um anschließend Analysen auszuführen, würde es zu hohen Effizienzeinbußen kommen. Besonders wenn Analyseergebnisse schnell oder sogar in Echtzeit erreicht werden müssen, zum Beispiel in sicherheitskritischen An-

wendungen oder im Bereich der Produktion, stellt die Persistierung aller Daten keine Option dar. Aus diesem Grund ist es notwendig, neue Techniken und Technologien einzuführen, um hiermit umgehen zu können. Diese umfassen beispielsweise strombasierte Datenverarbeitungssysteme mittels *Complex-Event-Processing* [Luc01] oder moderne Analyseverfahren wie das *Data-Stream-Mining* [GZK10].

Eine weitere Herausforderung bei der Verarbeitung von Big Data ist die Vielfalt der Daten. Oftmals können erst durch Hinzunahme mehrere Datenquellen gute Analyseergebnisse mit hoher Genauigkeit erreicht werden. Beispielsweise reicht es bei einer Untersuchung der Unfallursachen auf Straßen oftmals nicht aus, die reinen Unfalldaten des Fahrzeugs zu betrachten, sondern es müssen zusätzlich weitere Daten wie Wetterinformationen, Sonnenstand oder außergewöhnliche Ereignisse wie Baustellen oder Straßensperrungen beachtet werden [HWF+17]. Diese Daten weisen eine hohe Heterogenität bezüglich ihrer Struktur (strukturiert, unstrukturiert, semistrukturiert), ihrer Datentypen und Formate, sowie Datenwerte auf. Müssen zusätzlich ähnliche Daten integriert werden, um für Analysen verwendbar zu sein, zum Beispiel die Integration von Wetterdaten verschiedener Wetterstationen mit unterschiedlichen Schemata, steigt die Komplexität noch weiter an. Es muss automatisch entschieden werden können, ob sich Daten tatsächlich ähneln oder ob sie erst transformiert werden müssen, um integrierbar zu sein, zum Beispiel, wenn Temperaturdaten aus einer Quelle in Celsius vorliegen und aus einer anderen in Fahrenheit.

Um hiermit umzugehen, wurden einige Ansätze, vor allem im Bereich Daten- und Informationsintegration [LN07], geschaffen. Diese Ansätze umfassen beispielsweise das *Schema-Matching*, bei dem zwei Datenbankschemata in ein Einzelnes überführt werden oder das *Schema-Mapping*, in dem ein globales Schema eingeführt wird, das anschließend auf die darunterliegenden zu integrierenden Schemata abgebildet werden kann [LN07]. Neben diesen Techniken wurden verschiedene Möglichkeiten geschaffen, um die Ähnlichkeit von Daten zu bestimmen, wie zum Beispiel die *Levenshtein-Distanz* [Lev66], mit der die Ähnlichkeit zweier Zeichenketten bestimmt werden kann. Die größte Herausforderung bei der Integration von Daten

stellen Konflikte dar, also Datenwerte, die sehr unterschiedlich sind oder sich sogar widersprechen [DN09].

Zusammengefasst ist der Umgang mit den beschriebenen Herausforderungen der Big-Data-Charakteristiken ein wichtiger Bestandteil dieser Dissertation. Hierfür sollen bestehende Techniken und Technologien, wie zum Beispiel die bisher aufgeführten, kombiniert und integriert werden.

1.3.5 Dynamische Umgebungen

Dynamische bzw. sich schnell ändernde Umgebungen stellen eine große Herausforderung bei der Verarbeitung und Analyse von Daten dar. Insbesondere die modernen Paradigmen *Internet der Dinge* [VF14] sowie *Industrie 4.0* [Sen13] zeichnen sich durch eine hohe Dynamik aus. Das bedeutet, dass sich zugrundeliegende Datenquellen oft ändern. Derartige Änderungen treten dabei durch den Ausfall von Geräten oder deren Hinzufügen auf. Besonders im Bereich Internet der Dinge ist es üblich, dass mobile Geräte Umgebungen bzw. Netzwerke laufend betreten oder verlassen. Beispielsweise kann ein Smartphone, das einem Netzwerk beitritt, zusätzlich als neue Datenquelle für Analysen, beispielsweise zur Erkennung bestimmter Situationen (z.B. ob eine Person das Gebäude betreten hat), hinzugezogen werden. Beim Verlassen des Netzwerkes fällt diese Datenquelle ohne Vorwarnung weg. Bisherige Ansätze können mit dieser Dynamik oftmals nicht umgehen. Dabei werden Analyseergebnisse verfälscht oder es führt sogar zu Systemabstürzen und daraufhin zu einer aufwendigen Änderung der involvierten Schritte im Analyseprozess. Ein Umgang mit der beschriebenen Dynamik ist zwingend notwendig, um moderne Szenarien, wie zum Beispiel im Bereich Internet der Dinge oder Industrie 4.0, zu ermöglichen. Daher muss eine Möglichkeit geschaffen werden, um das dynamische Hinzufügen und Entfernen von Datenquellen zu ermöglichen.

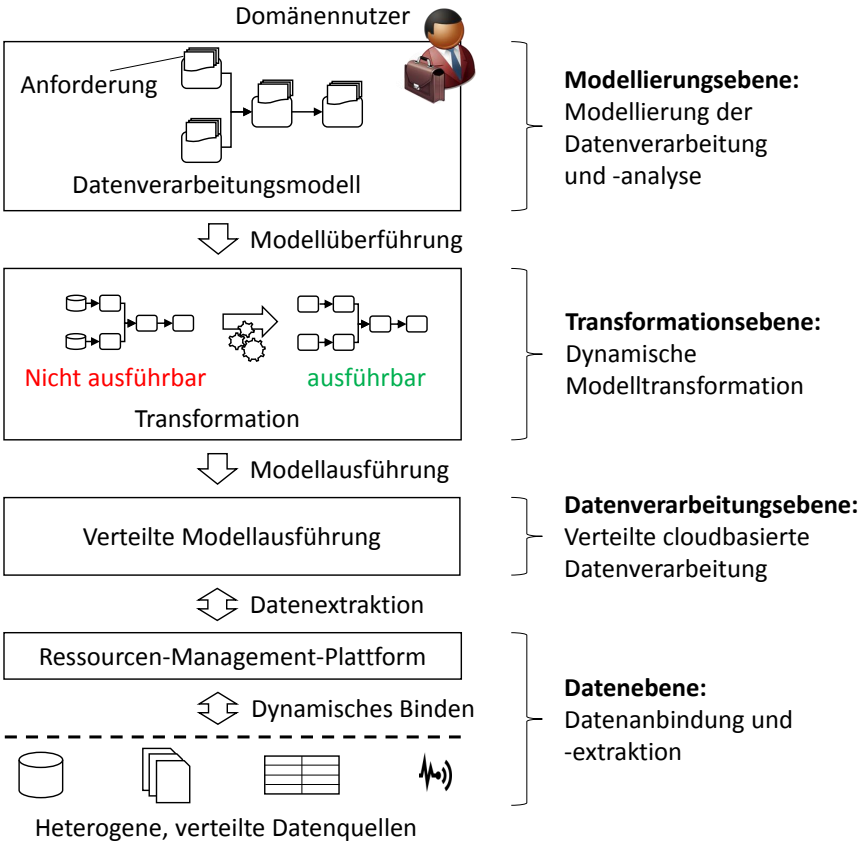

Abbildung 1.1: Übersicht der Beiträge dieser Dissertation

1.4 Beiträge der Dissertation

In diesem Abschnitt werden die Beiträge der Dissertation beschrieben, die sich auf die in Abschnitt 1.2 beschriebenen Forschungsfragen und Ziele beziehen. Dabei kann jeder Beitrag einem der Ziele zugeordnet werden. Die Beiträge bilden eine gesamtheitliche Lösung für die anforderungsbasierte Modellierung und Ausführung der Datenverarbeitung. Die im Folgenden vorgestellte Gesamtlösung basiert dabei vollständig auf etablierten Standards

(z.B. TOSCA, XML) oder de-facto Standards (z.B. BPEL, SOAP), um deren Anwendbarkeit sowie ein hohes Maß an Zukunftssicherheit zu gewährleisten.

Die Beiträge der Dissertation wurden auf nationalen und internationalen Konferenzen, in Fachmagazinen, und Buchkapiteln veröffentlicht. Eine Übersicht der Publikationen befindet sich am Ende des Dokuments (S. 229 ff.). Zusätzlich wurde im Rahmen der Dissertation entstandene Software als Open-Source-Projekte auf etablierten Online-Plattformen, wie z.B. Github[1], veröffentlicht.

Abbildung 1.1 zeigt eine Gesamtübersicht der Beiträge der Dissertation. Diese Beiträge verteilen sich auf folgende vier Ebenen:

- Die **Modellierungsebene** enthält Konzepte, die beschreiben, wie die Verarbeitung von Daten mittels Datenflussgraphen abstrakt modelliert werden kann. Dabei steht die Unterstützung von den in Abschnitt 1.3.1 beschriebenen Gruppen von Domänennutzern im Vordergrund.
- Die **Transformationsebene** überführt das nicht ausführbare Datenverarbeitungsmodell anschließend in ausführbare Repräsentationen. Dabei hängt die Auswahl der Ausführungskomponenten von den funktionalen und nichtfunktionalen Anforderungen des Modellierers ab, wodurch eine individualisierte Ausführung ermöglicht wird. Zusätzlich wird die Ausführungsumgebung vollautomatisiert, baukastenartig und basierend auf den spezifischen Nutzeranforderungen aufgebaut.
- Die **Datenverarbeitungsebene** führt dieses Modell aus, indem die Daten mittels den aufgesetzten Komponenten unter Berücksichtigung der Anforderungen verarbeitet und analysiert werden.
- Die **Datenebene** steht im engen Zusammenhang mit der Datenverarbeitungsebene. Sie dient dazu, der Datenverarbeitung die Daten zur Verfügung zu stellen und kümmert sich darüber hinaus um die dynamische, automatische Anbindung von zu verarbeitenden Datenquellen.

Die Ebenen werden im Folgenden genauer beschrieben.

[1] http://github.com/

1.4.1 Modellierungsebene: Modellierung der Datenverarbeitung

Bei der Modellierung der Datenverarbeitung (vgl. Modellierungsebene, Abbildung 1.1) stehen die Bedürfnisse der Domänennutzer im Vordergrund. Wie in Abschnitt 1.3.1 beschrieben, unterscheiden sich diese, je nach Anwender, stark voneinander. Das Ziel dieses Beitrags ist es, eine individualisierte Benutzererfahrung für jede der beschriebenen Anwendergruppen zu ermöglichen. Das bedeutet, dass es sowohl möglich sein soll, die Bedienbarkeit für Anwender ohne jegliche IT-Kenntnisse zu realisieren, als auch Expertenanwender dabei zu unterstützen, möglichst viel Kontrolle, bspw. durch Parametrisierung von Datenanalysen, über die Ausführung zu erlangen. Die Trennung der Verantwortlichkeiten wird durch verschiedene Ansichten ermöglicht, sodass die Bedürfnisse jedes Anwenders klar getrennt werden können. Als Grundlage für die Modellierung der Datenverarbeitung und -analyse dient das, im Rahmen der vorliegenden Dissertation entwickelte, *Datenverarbeitungsmodell (DVM)*. Dabei handelt es sich um einen gerichteten Datenflussgraphen, der Datenverarbeitungsschritte als Knoten sowie den Datenfluss als Kanten beschreibt.

Bei der Modellierung ist es vor allem eine große Herausforderung die Anwendbarkeit durch Nutzer ohne jegliche IT-Kenntnisse zu ermöglichen. Hierfür werden sogenannte Modellierungsmuster [Hir15; HRWM15] eingeführt, die eine vollständige Abstraktion von technischen Details bieten, sodass Anwender lediglich mit (Daten-)Objekten und Operationen arbeiten, die ihnen bekannt sind. Die Konzepte basieren teilweise auf den von Künzle et al. [Kün+11] sowie von Cohn et al. [Coh+09] vorgestellten artefaktzentrierten Ansätzen, in denen sogenannte *Business Artefakte* geschaffen werden, deren Ziel ebenfalls eine Abstraktion technischer Details darstellt. Diese Artefakte können anschließend, beispielsweise mittels den von Sun et al. [Sun+14] vorgestellten Ansätzen, auf die konkreten, technischen Artefakte abgebildet werden. Zusätzlich werden Modellierer durch die Generierung von Modellierungsvorschlägen unterstützt.

Eine Abgrenzung zu diesen und ähnlichen Ansätzen befindet sich in der Detailbeschreibung des Beitrags in Kapitel 4.

1.4.2 Transformationsebene: Anforderungsgetriebene Modelltransformation

Das Konzept zur anforderungsgetriebenen Modelltransformation (vgl. Transformationsebene, Abbildung 1.1) ermöglicht es, die Ausführungsumgebung zur Datenverarbeitung dynamisch basierend auf funktionalen und nichtfunktionalen Anforderungen abhängig vom Anwendungsfall aufzubauen.

Hierfür wird es dem Anwender ermöglicht, nichtfunktionale Anforderungen [HM16a] an die Ausführung zu definieren, die als Grundlage der Zusammensetzung einer passenden Ausführungsumgebung dienen. Derartige Anforderungen umfassen typischerweise Effizienz, Robustheit, Kosten, oder Sicherheit. Als Grundlage dient dabei ein Katalog, der eine Beschreibung der Anforderungen enthält. Eine Auswahl mehrerer Anforderungen ist dabei zwar prinzipiell möglich, jedoch können sich diese beeinflussen oder sogar beschränken. Beispielsweise führen hohe Anforderungen an die Effizienz der Ausführung typischerweise zu einer Erhöhung der Kosten. Der Umgang mit derartigen komplexen Abhängigkeiten ist Teil dieses Beitrags.

Basierend auf den ausgewählten Anforderungen wird anschließend die bestmögliche Ausführungsumgebung mit deren Komponenten und Abhängigkeiten bestimmt. Das Aufsetzen der Ausführungsumgebung funktioniert baukastenartig und vollautomatisch. Die Konzepte zum automatischen Aufsetzen basieren dabei auf dem Standard *Topology and Orchestration Specification for Cloud Applications* (TOSCA) [OAS13a; OAS13b] der *Organization for the Advancement of Structured Information Standards* (OASIS).

Durch diesen Ansatz können die Details der Ausführung vor den Anwendern verborgen und trotzdem eine Möglichkeit der Beeinflussung durch die Definition der Anforderungen ermöglicht werden. Das Verbergen von Details ist insbesondere bei Anwendern mit wenig oder ohne IT-Kenntnisse wichtig, da die Details der Datenverarbeitung (z.B. monolithisch oder verteilt) nicht durch die Anwender selbst definiert werden müssen.

Zusammengefasst wird es durch die Konzepte des Beitrags ermöglicht, eine individualisierte Ausführung der Datenverarbeitung basierend auf spezifischen Nutzeranforderungen zu schaffen.

1.4.3 Datenverarbeitungsebene: Ausführung der Datenverarbeitung

Dieser Beitrag beschäftigt sich mit der Ausführung der Datenverarbeitung (vgl. Ausführungsebene, Abbildung 1.1) und nimmt dabei u.a. auf die Charakteristiken von Big Data Bezug (vgl. Abschnitt 1.3.4). Eine Verteilung von Daten ermöglicht beispielsweise viele Vorteile, vor allem bezüglich Effizienz, aber auch bezüglich Verfügbarkeit und Ausfallsicherheit. In den letzten Jahren ergaben sich viele neue Möglichkeiten, insbesondere durch das Cloud-Computing [MG+11], durch Datenverarbeitungskonzepte wie zum Beispiel Map-Reduce [TSJ+09], oder durch neuartige Technologien wie Apache Spark [Spa15]. Heutzutage ist eine Verarbeitung großer Datenmengen ohne derartige Konzepte undenkbar. In diesem Beitrag wird behandelt, wie derartige Konzepte verknüpft und eingesetzt werden können. Durch die Verknüpfung entsteht eine gesamtheitliche Lösung für die Verarbeitung von Daten, die die Vorteile der einzelnen Konzepte zur Geltung bringt. Dadurch können wichtige Anforderungen wie Effizienz, Robustheit und Ausfallsicherheit sowie Datensicherheit realisiert werden.

1.4.4 Datenebene: Ressourcen-Management-Plattform

Die Ressourcen-Management-Plattform [HBS+16; HWBM16a; HWBM16b] (RMP, vgl. Datenebene, Abbildung 1.1) stellt einen wichtigen Beitrag dieser Dissertation dar. Dabei handelt es sich um ein umfassendes Konzept, um mit der in Abschnitt 1.3.5 beschriebenen Dynamik von Datenquellen umzugehen. Die Ressourcen-Management-Plattform bietet einerseits eine uniforme Schnittstelle, um Daten einer IT-Umgebung aufzufinden und auf diese zuzugreifen, abstrahiert also von den konkreten Datenquellen. Andererseits bietet die RMP die Möglichkeit, Datenquellen automatisch anzubinden, sodass es nicht mehr notwendig ist, spezifischen Programmcode zu erstellen, um Daten aus ihren Quellen zu extrahieren. Dabei werden generische Adapter verwendet, die parametrisiert und anschließend automatisch aufgesetzt werden. Zusätzlich können Daten durch die RMP transformiert und bereinigt werden, um deren Qualität zu erhöhen.

Zur Umsetzung der Ressourcen-Management-Plattform kommen unter anderem Konzepte der Topology and Orchestration Specification for Cloud Applications [OAS13a; OAS13b] zum Einsatz [HBBL14], wodurch das automatische Anschließen von Datenquellen realisiert werden kann. Zusammenfassend stellt die RMP eine zentrale Komponente dar, die von den Datenquellen abstrahiert und somit den Datenzugriff durch Anwendungen, z.B. zur Datenanalyse, deutlich vereinfacht. Die RMP findet dabei bei der in Abschnitt 1.4.3 beschriebenen Datenverarbeitung Anwendung, wodurch diese nicht direkt auf die Datenquellen zugreifen muss.

Die Beiträge dieser Dissertation wurden auf nationalen und internationalen Konferenzen, in Fachmagazinen, und Buchkapiteln veröffentlicht. Eine Übersicht der Publikationen befindet sich am Ende des Dokuments (S. 229 ff.). Zusätzlich wurde im Rahmen dieser Arbeit entstandene Software online als Open-Source-Projekte veröffentlicht.

1.5 Aufbau der Arbeit

Die vorliegende Dissertation ist wie folgt aufgebaut: Nachdem die Motivation, die Herausforderungen und Beiträge der Dissertation in Kapitel 1 beschrieben werden, behandelt Kapitel 2 die Grundlagen der Arbeit. Diese umfassen Big-Data sowie Datenanalysen, Services und Serviceorientierte Architekturen, sowie Cloud-Computing und das Internet der Dinge. Daraufhin wird in Kapitel 3 zuerst eine Übersicht über die Beiträge der Dissertation gegeben und anschließend werden die einzelnen Beiträge in den Kapiteln Kapitel 4, Kapitel 5, Kapitel 6, Kapitel 7, und Kapitel 8 beschrieben. Dabei enthält jedes Kapitel eine Beschreibung der Konzepte, verwandter Arbeiten, sowie eine dazugehörige Evaluation. Anschließend an die Beiträge wird eine qualitative und quantitative Gesamtevaluation der integrierten Beiträge in Kapitel 9 beschrieben. Abschließend werden in Kapitel 10 die Ergebnisse der Dissertation zusammengefasst und bewertet, und es werden zukünftige Arbeiten beschrieben.

Kapitel 2

Grundlagen

Dieses Kapitel beschreibt Grundlagen, die für die Beiträge dieser Dissertation wichtig sind. Diese umfassen (1) Big-Data, Datenanalysen und der KD-Prozess (engl.: Knowledge Discovery Process), (2) Services, Serviceorientierte Architekturen und Workflows und (3) Cloud-Computing und Internet der Dinge.

2.1 Big-Data

Die in Abschnitt 1.1 angesprochene Komplexität von Big-Data lässt sich in drei Kategorien aufteilen: (1) Volumen, (2) Vielfalt, und (3) Geschwindigkeit. Dies wird in der Fachliteratur oftmals als *die drei Vs* (engl.: volume, variety, velocity) des Big-Data bezeichnet [MW15]. Darüber hinaus existieren Ansätze, die diese Kategorien um weitere Aspekte auf *fünf Vs* [Jai16] oder sogar *sieben Vs* [DeV16] erweitern. Dabei umfasst die Definition mit fünf Vs, neben den drei genannten, die zusätzlichen Kategorien: (4) Veränderlichkeit (engl.: variability) sowie (5) den Wert (engl.: value) der Daten. Bei der Definition mit 7 Vs kommen die Kategorien: (6) Wahrhaftigkeit (engl.: veracity) und (7) Visualisierung (engl.: visualization) hinzu. Diese Kategorien werden im

Folgenden beschrieben. Dabei liegt der Fokus auf der Big-Data-Definition mit drei Vs, da diese für die Beiträge dieser Dissertation besonders wichtig sind, die anderen Kategorien spielen hingegen eine untergeordnete Rolle und werden daher nicht ausführlich beschrieben.

2.1.1 Volumen

Mit dem Begriff Big-Data wird in der Regel vor allem das *Volumen* von zu verarbeitenden Daten assoziiert. Auch wenn das Volumen von Daten einen wichtigen Faktor und eine große Herausforderung für deren Verarbeitung darstellt, ist dies der Aspekt von Big-Data, mit dem bislang am besten umgegangen werden kann. Die genaue Grenze, ab welcher Datenmenge das Volumen eines Datensatz in die Big-Data-Problematik fällt, ist zwar nicht genau definiert, jedoch gibt es mittlerweile einige Techniken und Technologien, die Daten im Gigabyte-Bereich effizient verarbeiten können. Eines der bekanntesten Verfahren ist dabei *Map-Reduce* [TSJ+09]. Hierbei werden Daten aufgeteilt, verteilt in einem Rechencluster verarbeitet, und anschließend wieder zusammengeführt. Dabei wächst die Größe des Rechenclusters mit der zu verarbeitenden Datenmenge. Weitere, bei großen Datenmengen eingesetzte Verfahren, sind Streaming-Architekturen. Ein bekannter Vertreter dieses Verfahrens ist das Streaming-Framework Apache Spark [Spa15]. In derartigen Frameworks werden Daten strombasiert eingelesen und im Hauptspeicher verarbeitet. Aus diesem Grund erfordert ein effektiver Einsatz dieser Technologien auch leistungsfähige, skalierbare Rechenumgebungen. In der Regel werden sie daher auf virtualisierten Cloudumgebungen ausgeführt, in denen der Hauptspeicher eines Rechenknotens im Cluster dynamisch erhöht werden kann. Dies wird als *vertikale Skalierbarkeit* [FLR+14] bezeichnet. Zusätzlich kann durch *horizontale Skalierbarkeit* [FLR+14] die Anzahl der Rechner im Cluster erhöht werden.

Diese Verfahren und Techniken ermöglichen einen guten Umgang mit großen Datenmengen. Jedoch ist nicht nur das Volumen von Big-Data ein kritischer Faktor, sondern auch die Vielfalt und Geschwindigkeit der Daten. Diese werden im Folgenden beschrieben.

2.1.2 Vielfalt

Die *Vielfalt* der Daten bezeichnet deren Heterogenität, also beispielsweise verschiedene Datentypen, Datenformate und Strukturen. Sollen große Mengen an heterogenen Daten aggregiert und verarbeitet werden, führt dies zu einem komplexen Problem. Dies liegt daran, dass diese Daten in eine einheitliche Struktur überführt werden müssen, um verarbeitet werden zu können. Insbesondere bei unstrukturierten Daten gestaltet sich dies oft als schwierig und erfordert anspruchsvolle Techniken (siehe bspw. Feldman et al. [FS06]). Aber auch bei strukturierten Daten können Probleme durch Heterogenität entstehen, beispielsweise bei Datenkonflikten, Fehlern oder gleich benannten Tabellenspalten mit unterschiedlichen enthaltenen Daten.

Derartige Probleme treten vor allem bei der Zusammenführung von Daten auf, die für Datenanalysen in der Regel notwendig ist. Der Forschungsbereich der Daten- und Informationsintegration (z.B. [Her12; LN07]) entwickelt Techniken, um hiermit umzugehen und dadurch die Datenqualität zu erhöhen. Durch eine vor der Datenverarbeitung und -integration durchgeführte "Säuberung" (engl.: Data cleansing) der Daten, bspw. durch das Entfernen von Fehlern oder duplizierten Werten, kann die Zusammenführung heterogener Daten erleichtert werden. Jedoch sind aufwendige Datentransformationen immer noch notwendig. Da sich Daten oftmals auch in ähnlichen Szenarien stark unterscheiden, müssen derartige Transformationen oft spezifisch für einen konkreten Anwendungsfall erstellt werden.

2.1.3 Geschwindigkeit

Das letzte der drei Big-Data Vs ist die *Geschwindigkeit* der Daten. Darunter wird verstanden, wie oft sich die Daten ändern, also wie oft diese aktualisiert werden. Der Umgang mit derartigen Daten stellt sich als schwierig dar, da Daten aus Effizienzgründen vor deren Verarbeitung nicht mehr persistiert werden können, wie es in Data-Warehouse-basierten Ansätzen üblich ist. Stattdessen müssen Daten strombasiert eingelesen und direkt verarbeitet und analysiert werden. Dies erfordert anspruchsvolle Datenverarbeitungs-

techniken wie das *Data-Stream-Mining* [GZK10]. Besonders mit neuartigen Paradigmen wie dem Internet der Dinge oder Industrie 4.0 ist eine derartige strombasierte Verarbeitung von Daten essentiell. Darüber hinaus unterstützen moderne Datenverarbeitungssysteme wie Apache Flink [CKE+15] oder Apache Storm [Sto14] eine strombasierte Verarbeitung von Daten.

2.1.4 Veränderlichkeit und Wert

Die Big-Data-Definition mit 5 Vs [Jai16] nimmt neben den beschriebenen Aspekten die Veränderlichkeit sowie den Wert der Daten hinzu.

Die *Veränderlichkeit* von Daten unterscheidet sich von der beschriebenen Datenvielfalt, da sie sich nicht auf die Struktur der Daten bezieht, sondern auf deren Bedeutung. Die Herausforderung ist, dass sich die Bedeutung der Daten, abhängig vom Kontext, sehr stark unterscheiden kann. Dies hat wiederum großen Einfluss auf Analyseergebnisse. Ein Beispiel hierfür sind Wörter bei der Textanalyse. Zwar sind diese oftmals gleich, jedoch variiert deren Bedeutung je nach Kontext des Satzes sehr stark. Beispielsweise kann sich das Wort “Titel” sowohl auf eine Person, als auch auf ein Buch oder einen Film beziehen. Ein Umgang mit der Veränderlichkeit der Daten stellt eine große Herausforderung für die Datenverarbeitung und -analyse dar.

Der *Wert* von Daten bezeichnet deren Nützlichkeit bezogen auf das Anwendungsszenario. Gerade bei langlaufenden Datenanalysen ist es wichtig, die zu verarbeitenden Daten möglichst gering zu halten. Dies kann durch das Filtern von Daten mit geringem Wert ermöglicht werden. Der Wert definiert sich dabei nicht durch die Qualität der Daten, sondern ob sie im Kontext der Analyse überhaupt für eine Ergebnisfindung notwendig sind. Da der Wert von Daten abhängig vom Anwendungsfall bestimmt werden muss, führt dies zu einem komplexen Problem. Es muss genau analysiert werden, in welchem Kontext Daten einen Wert besitzen.

2.1.5 Wahrhaftigkeit und Visualisierung

In manchen Publikationen, z.B. von Dev [DeV16], sind Big-Data-Definition zu finden, die zusätzlich die Wahrhaftigkeit und Visualisierung als essentielle Aspekte des Big-Data bezeichnen.

Die *Wahrhaftigkeit* von Daten bezeichnet dabei die Qualität der Daten, also beispielsweise, ob deren Genauigkeit ausreichend ist. Beispiele hierfür sind fehlerhafte Personendaten, wie zum Beispiel falsch geschriebene Namen oder Adressen. Für Datenanalysen fällt oft der Satz *"garbage in, garbage out"*, womit gemeint ist, dass Daten mit schlechter Qualität auch zu schlechten Analyseergebnissen führen. Die Analyse und Steigerung der Datenqualität ist komplex und erfordert oftmals die Hinzunahme von Menschen mit spezifischem Wissen über die Daten.

Der letzte Aspekt, die *Visualisierung* von Daten, spielt in dieser Dissertation eine untergeordnete Rolle. Dabei geht es vor allem darum, Daten in geeigneter Form für Menschen darzustellen. Diese können damit Einblicke bekommen oder sogar bereits erste Zusammenhänge und Muster selbst, ohne aufwendige Datenanalysen, erkennen. Eine geeignete Visualisierung, insbesondere für Big-Data, zu finden ist nicht einfach, da viele Visualisierungen mit der wachsenden Anzahl an Daten schnell unübersichtlich werden. Der Forschungsbereich *Visual Analytics* beschäftigt sich mit dem Umgang mit diesen Herausforderungen [KAF+08].

Zusammengefasst stellen diese Aspekte die Hauptherausforderungen bei der Verarbeitung von Big-Data und somit unter anderem auch dieser Dissertation dar. Wie Daten im Allgemeinen verarbeitet und analysiert werden wird im Folgenden beschrieben.

2.2 Datenanalysen und der Knowledge-Discovery-Prozess

Abbildung 2.1 stellt die Schritte des Knowledge-Discovery-Prozesses dar, der angewendet wird, um Wissen aus Daten zu gewinnen, was das übergeordnete Ziel dieser Dissertation darstellt. Dieser ist in fünf Schritte unterteilt: (1) Selektion, (2) Vorverarbeitung, (3) Transformation, (4) Data-Mining, und (5) Interpretation. Als Eingabe für diesen Prozess dient eine Menge an Daten, die heterogen und verteilt vorliegen kann.

Im ersten Schritt, der *Selektion*, werden die für die Analyse benötigten Daten ausgewählt. Dieser Schritt geschieht oftmals manuell, da die zu verwendenden Daten stark vom Anwendungsfall abhängen und hierfür deren Qualität und Relevanz berücksichtigt werden müssen. Dieser Schritt ist von hoher Wichtigkeit im KD-Prozess, da eine falsche Auswahl der Daten bzw. eine Nichtberücksichtigung relevanter Daten negative Auswirkungen auf die Qualität der Analyseergebnisse haben.

Sind die Daten für die Analyse ausgewählt, werden diese im zweiten Schritt vorverarbeitet. Bei der *Vorverarbeitung* ist es das Ziel, die Qualität der Daten zu erhöhen. Dies wird ermöglicht, indem beispielsweise Fehler erkannt bzw. korrigiert werden, leere Dateneinträge gelöscht, sowie fehlende Daten ergänzt werden. Aus diesem Grund wird dieser Schritt oftmals auch als *Datenbereinigung* bezeichnet. Dabei werden die Daten bezüglich deren Validität, Vollständigkeit, Einheitlichkeit sowie Integrität untersucht und es werden Schritte eingeleitet, um die Qualität dieser Aspekte zu erhöhen.

Im dritten Schritt werden die vorverarbeiteten bzw. bereinigten Daten transformiert. Eine *Transformation* der Daten ist notwendig, um diese in ein für die Analyse passendes Format zu überführen. Dabei spielt jedoch nicht nur die Struktur der Daten eine Rolle, sondern oftmals die Datenwerte selbst. Beispielsweise muss ein Temperaturwert, der in Fahrenheit vorliegt, erst transformiert werden, wenn die Analyse diesen Wert in Celsius erwartet. Diese Transformation geschieht in der Regel vollautomatisch. Die Auswahl der notwendigen Transformationen muss momentan noch manuell erfolgen.

Im vierten Schritt, dem *Data-Mining*, findet die eigentliche Analyse der Daten statt. Das Ziel ist es, interessante Muster in diesen zu finden. Bekannte

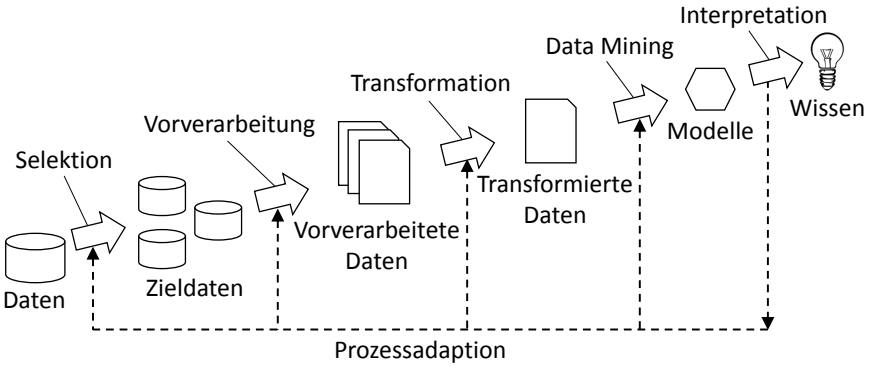

Abbildung 2.1: Schritte des Knowledge-Discovery Prozesses (basierend auf [FPS96])

Algorithmen für das Data-Mining sind beispielsweise Clusteranalysen wie *k-Means* [HW79] oder Assoziationsanalysen wie *Apriori* [AIS93a; AIS93b]. Das Ergebnis derartiger Analysealgorithmen sind sogenannte *Modelle*, eine Beschreibung von Mustern in den Daten. Am Beispiel der genannten Data-Mining-Algorithmen sind das beispielsweise die gebildeten Cluster, die Daten gruppieren, oder die Assoziationsregeln aus der Assoziationsanalyse, die beschreiben, dass bestimmte Daten voneinander abhängen.

Im letzten Schritt werden durch die *Interpretation* der Analyseergebnisse Erkenntnisse erlangt, die weitläufig als *extrahiertes Wissen* bezeichnet werden. Zum Beispiel kann eine Warenkorb-Assoziationsanalyse eines Supermarktes ergeben, dass bestimmte Produkte oftmals zusammen gekauft werden. Der Supermarktbetreiber könnte daraufhin das gewonnene Wissen nutzen, um diese Produkte möglichst nah beieinander zu platzieren. Darüber hinaus können aus den Ergebnissen auch Rückschlüsse auf den KD-Prozess selbst geschlossen werden. Sind dessen Ergebnisse nicht zufriedenstellend, können die einzelnen Schritte des Prozesses mittels *Prozessadaption* angepasst werden, zum Beispiel, indem zusätzliche Daten hinzugezogen werden. Daraus ergibt sich eine kontinuierliche Verbesserung des Prozesses, die so lange fortgeführt wird, bis dieser optimale Ergebnisse erzielt.

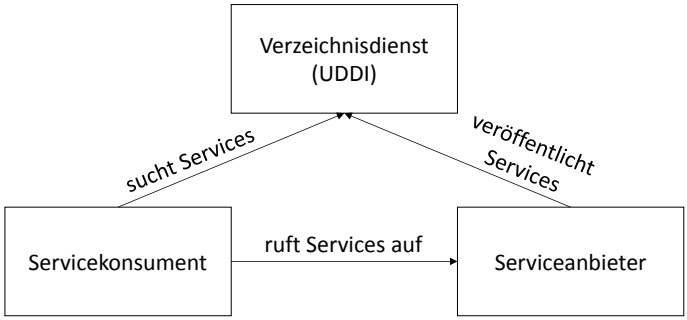

Abbildung 2.2: Das SOA-Dreieck (basierend auf [WCL+05])

2.3 Services, Serviceorientierte Architekturen und Workflows

Dieser Abschnitt beschreibt die Grundlagen von Services, Serviceorientierten Architekturen (SOA) sowie Workflows.

Serviceorientierte Architekturen [WCL+05] beschreiben Anwendungen, die aus mehreren Services zusammengesetzt sind. Ein Service ist dabei ein modulares, abgeschlossenes Softwareprogramm, das bestimmte Funktionalitäten über eine einheitliche Schnittstelle zur Verfügung stellt. Services zeichnen sich im Gegensatz zu Objekten der objektorientierten Programmierung vor allem dadurch aus, dass sie immer erreichbar (engl.: always on) sind. Durch die Verknüpfung von Services können komplexe, hochfunktionale Anwendungen entstehen. Dieser modulare Aufbau ermöglicht große Vorteile zum Beispiel bezüglich der Wartbarkeit der Anwendung. Änderungen müssen dabei nur in einzelnen Services vorgenommen werden. Solange sich deren Schnittstelle nicht ändert, erfordert dies keine Anpassung in der Anwendung.

In Abbildung 2.2 ist das grundlegende Prinzip von Serviceorientierten Architekturen zu sehen. Dabei existieren die drei Rollen *Serviceanbieter*, *Servicekonsument* und *Verzeichnisdienst*. Serviceanbieter implementieren die Services und stellen deren Funktionalität, oftmals kostenpflichtig, den Servicekonsumenten zur Verfügung. Servicekonsumenten hingegen nutzen die Services, um ihre Anwendungen zu realisieren.

Der standardisierte Verzeichnisdienst, *Universal Description, Discovery and Integration (UDDI)* [WCL+05] genannt, stellt die zentrale Komponente dieser Architektur dar. Dieser verwaltet die verfügbaren Services. Das UDDI kann dabei von Servicekonsumenten nach benötigten Services durchsucht werden, die für die Implementierung einer bestimmten Anwendung verwendet werden können. Um dies zu ermöglichen werden Informationen über die Funktionalitäten der Services von den Serviceanbietern bereitgestellt. Dabei nutzt UDDI die sogenannten *White-*, *Green-*, und *Yellow-Pages* zur Beschreibung von Services. Die White-Pages enthalten Informationen über die Serviceanbieter, beispielsweise deren Kontaktinformationen. In den Yellow-Pages werden die Services basierend auf ihren Attributen klassifiziert. Dies ist insbesondere wichtig, um diese effektiv auffinden zu können. Die Green-Pages enthalten die konkrete Spezifikation der Services, also zum Beispiel deren Schnittstellendefinitionen. Mittels sogenannten *Policys* können darüber hinaus die nichtfunktionalen Fähigkeiten oder Nutzungsbedingungen der Services beschrieben werden (z.B. Nutzungskosten).

Die Orchestrierung von Services, also deren Zusammenspiel, wird von *Workflows* [WCL+05] durchgeführt. Diese beschreiben, welche Services in welcher Reihenfolge aufgerufen werden. Hierbei werden die Ein- bzw. Ausgabedaten der Services bei deren Aufruf übergeben. Zusätzlich ermöglicht der Einsatz von Workflows wichtige Aspekte wie Fehlerbehandlung, Parallelisierung oder auch Provenance [WHW15]. Etablierte Workflowsprachen, um Services zu orchestrieren, sind WS-BPEL [ACD+03] und BPMN [Gro06].

Ein weiteres Konzept zur Umsetzung serviceorientierter Architekturen sind Servicebusse [Cha04; SHLP05]. Diese bieten eine Abstraktionsebene, um die konkreten Serviceimplementierungen bzw. Schnittstellen zu verbergen. Dadurch wird es ermöglicht, dass alle Serviceanfragen über den Servicebus laufen, der diese anschließend an die Services weiterleitet. Dies schafft vor allem den Vorteil, dass auf Änderungen der Serviceschnittstellen reagiert werden kann, indem man mehrere Versionen der Services betreibt und der Servicebus diese, abhängig von der Anfrage, auf die passende Version verteilt. Somit führt eine Anpassung von Serviceschnittstellen nicht zu Fehlern bei den Aufrufern der Services.

2.4 Cloud-Computing und der TOSCA-Standard

In diesem Abschnitt werden die Konzepte und Technologien des Cloud-Computings sowie des Standards TOSCA beschrieben.

2.4.1 Cloud-Computing

Seit einigen Jahren hat das Cloud-Computing eine große Bedeutung in der Informatik gewonnen. Insbesondere das dynamische Bereitstellen von Rechenressourcen ermöglicht neue Ansätze im Bereich der Datenverarbeitung. Dies wird durch eine Virtualisierung der Rechenressourcen ermöglicht. Laut der anerkannten National Institute of Standards and Technology (NIST)-Definition des Cloud-Computings [MG+11], definiert sich dieses durch die Charakteristiken: (1) bedarfsabhängige Selbstbedienung (engl.: On-demand self-service), (2) breiter Netzwerkzugriff (engl.: Broad network access), (3) Ressourcenpooling (engl.: Resource pooling), (4) dynamische Elastizität (engl.: Rapid elasticity), und (5) automatisierte Messungen (engl.: Measured service). Die folgenden Beschreibungen basieren auf der NIST-Definition [MG+11].

- **Bedarfsabhängige Selbstbedienung:**
  Ein Anwender kann eigenständig die verfügbaren Ressourcen, bspw. Netzwerkspeicher, für einen verwendeten Service festlegen und ändern. Diese sollen nach Bedarf automatisch angepasst werden können, ohne dass dabei menschliche Interaktion mit den Serviceanbietern notwendig ist.
- **Breiter Netzwerkzugriff:**
  Zugriff auf die Cloud wird über einen Netzwerkzugriff ermöglicht, indem standardisierte Kommunikationsmittel zum Einsatz kommen. Dadurch kann die Unterstützung sowohl von leichtgewichtigen als auch von schwergewichtigen Klientenplattformen ermöglicht werden (Mobiltelefone, Tablets, Laptops, PCs).

- **Ressourcen-Pooling:**
  Die von einem Cloud-Anbieter bereitgestellten Ressourcen werden zusammengelegt, um mehrere Anwender mit verschiedenen physischen und virtuellen Ressourcen dynamisch versorgen zu können. Abhängig vom Bedarf der Anwender werden diese Ressourcen dynamisch verteilt. Dabei hat der Anwender keine Kontrolle über die tatsächlichen, physischen Ressourcen, da dies durch Pooling abstrahiert wird.
- **Dynamische Elastizität:**
  Die bereitgestellten Ressourcen können – teilweise automatisiert – elastisch hinzugefügt und auch wieder freigegeben werden. Dies ermöglicht es einerseits, Ressourcen schnell zu skalieren, um leistungsfähige Services zu schaffen, und andererseits Kosten zu sparen, sobald Services nur wenige Ressourcen benötigen. Für den Anwender erscheinen die verfügbaren Ressourcen als unbegrenzt, sodass diese in beliebiger Anzahl jederzeit hinzugefügt werden können.
- **Automatisierte Messungen:**
  Cloud-Systeme können automatisiert ihre Ressourcen kontrollieren und optimieren. Die Nutzung von Ressourcen kann durch Monitoring, Kontrollen und Reports, auch durch Anwender, gemessen und dargestellt werden, wodurch vollständige Transparenz zwischen Anbieter und Anwender ermöglicht wird.

Neben diesen Charakteristiken definiert NIST die drei Servicemodelle Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), und Infrastructure-as-a-Service (IaaS). Diese sind in Abbildung 2.3 dargestellt.

Das SaaS-Modell beschreibt die Bereitstellung von Applikationen (Services) in virtualisierten Umgebungen, die die oben genannten Charakteristiken besitzen. Diese Applikationen sind durch verschiedenartige Klientensysteme (Web Browser oder programmierbare Schnittstellen) zugreifbar. Der Konsument der Services verwaltet die darunterliegenden Plattformen und Ressourcen (Netzwerk, Server, Speicher, Betriebssysteme) nicht selbst, dies wird vom Anbieter übernommen.

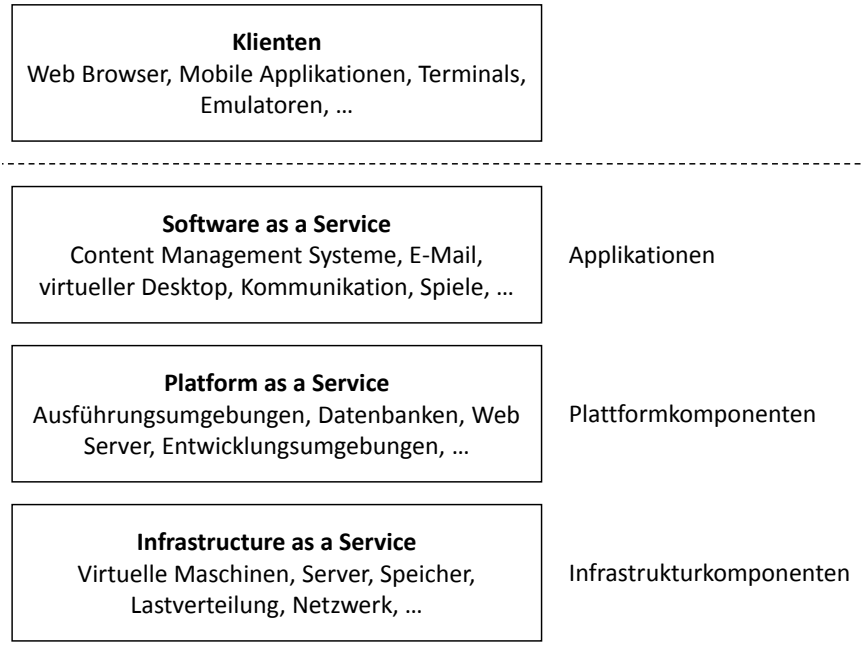

Abbildung 2.3: Servicemodelle des Cloud-Computing (basierend auf [MG+11])

Im Gegensatz zum SaaS-Modell wird im PaaS-Modell eine Plattform für die Erstellung von Anwendungen durch diverse Programmiersprachen, Bibliotheken, Werkzeuge und Services bereitgestellt. Dabei wird die darunterliegende Infrastruktur nicht selbst verwaltet. Jedoch hat der Anwender volle Kontrolle über die Bereitstellung seiner Anwendungen.

Im IaaS-Modell werden Ressourcen der Infrastruktur zur Verfügung gestellt. Dabei handelt es sich um grundlegende Ressourcen wie Speicher, Netzwerk oder Rechenressourcen. Dadurch können beliebige Software, Betriebssysteme, und Applikationen in der Cloud bereitgestellt werden. Zwar verwaltet der Anwender die darunterliegende Infrastruktur nicht selbst, hat dafür aber Kontrolle über Betriebssysteme, Speicher und Anwendungen, in manchen Fällen sogar über Netzwerkkomponenten wie Firewalls.

Neben diesen Service-Modellen beschreibt NIST außerdem die Deployment-Modelle Private-Cloud, Community-Cloud, Public-Cloud, und Hybrid-Cloud.

In Private-Clouds wird die Infrastruktur exklusiv für eine spezifische Organisation bereitgestellt, die diese i.d.R. selbst verwaltet und betreibt. Die Anwender sind dabei meist Mitarbeiter von Abteilungen dieser Organisation. In einigen Fällen wird die Verwaltung auch an Fremdanbieter ausgelagert. Dabei können Private-Clouds sowohl auf dem Gelände der Organisation betrieben werden oder außerhalb. Dieses Deployment-Modell bietet die höchste Sicherheit und ist daher vor allem für Firmen interessant.

Community-Clouds weisen große Gemeinsamkeiten mit Private-Clouds auf, dienen aber der Nutzung mehrerer Organisationen mit gleichen Interessen, wie zum Beispiel Themengebiete oder Sicherheitsanforderungen. Community-Clouds können durch einzelne Teilnehmer der Gemeinschaft, durch Fremdanbieter oder alle Teilnehmer gleichermaßen verwaltet werden. Somit kann immer noch ein hohes Maß an Sicherheit gewährleistet werden.

Public-Clouds sind im Gegensatz zu den vorigen Modellen für die Anwendung durch die breite Öffentlichkeit gedacht. Public-Clouds können durch Unternehmen, akademische Einrichtungen oder staatliche Einrichtungen besessen, verwaltet und betrieben werden.

Hybrid-Clouds stellen eine Kombination aus zwei der oben beschriebenen Modellen dar. Dabei bleiben die Cloud-Systeme eigenständig verfügbar und werden miteinander über standardisierte oder proprietäre Kommunikationsmittel vernetzt. Hierdurch werden Daten- oder Applikationsportabilität gewährleistet. Oftmals verwenden Firmen eine Kombination aus Public- und Private-Clouds, wobei firmeninterne Daten in Private-Clouds gehalten werden, öffentlich verfügbare Daten in Public-Clouds.

2.4.2 Topology and Orchestration Specification for Cloud Applications

Die Topology and Orchestration Specification for Cloud Applications (TOSCA) spielt eine wichtige Rolle für die Umsetzung der Konzepte dieser Dissertation. TOSCA dient der Provisionierung sowie dem Management von Anwendungen in Cloud-Computing-Umgebungen. Wie dem Namen zu entnehmen ist,

besteht TOSCA aus zwei Teilen: (1) einer Topologie (in TOSCA: *Topology Template*) der zu provisionierenden bzw. zu verwaltenden Anwendung, und (2) die Orchestrierung dieser Anwendung. Unter dem Begriff "Provisionierung" wird dabei das Aufsetzen von Softwarekomponenten innerhalb einer bestimmten Infrastruktur bezeichnet. Dies umfasst die Installation, Konfiguration und das Starten der Komponente.

Die Topologie beschreibt alle Komponenten, die für die Provisionierung, den Betrieb und das Management einer Anwendung benötigt werden. Dies umfasst sowohl anwendungsspezifische Komponenten, wie zum Beispiel Datenbanken, als auch Plattformkomponenten, wie Webserver oder Infrastrukturkomponenten, wie virtuelle Maschinen. Das bedeutet, dass TOSCA die Cloud-Computing-Modelle *Infrastructure-as-a-Service*, *Platform-as-a-Service*, und *Software-as-a-Service* unterstützt.

Die Topologie selbst ist ein Graph bestehend aus Knoten, den *Node-Templates*, und gerichteten Kanten, die in TOSCA als *Relationship-Templates* bezeichnet werden. Node-Templates definieren Softwarekomponenten wie Datenbanken, Programme, Webserver, virtuelle Maschinen, etc. Relationship-Templates definieren die Beziehungen zwischen den Komponenten. Der TOSCA-Standard spezifiziert bereits eine Menge an Relationship-Templates: (1) *hosted on* definiert, dass eine Komponente auf einer anderen Komponente gehostet wird, zum Beispiel eine Java-Anwendung auf einem Applikationsserver. (2) *connects to* beschreibt, dass sich eine Komponente zu einer anderen verbindet, zum Beispiel eine Applikation zu einer Datenbank. (3) *depends on* beschreibt Abhängigkeiten zwischen Komponenten, wodurch ausgedrückt werden kann, dass eine Komponente vor einer anderen aufgesetzt werden muss, damit diese lauffähig ist. Node- und Relationship-Templates sind typisiert durch Node-Types und Relationship-Types, die deren Struktur vorgeben. Der Aufbau von TOSCA-Topologien ist in Abbildung 2.4 dargestellt.

Jeder Node-Type enthält eine Liste an Eigenschaften (TOSCA: Properties), die bei der Modellierung eines Node-Templates in der Topologie angegeben werden müssen. Diese Eigenschaften dienen als Parameter für die zu Provisionierung und Management auszuführenden Funktionsaufrufe. Beispielsweise muss für das Aufsetzen einer virtuellen Maschine definiert werden, wie viel

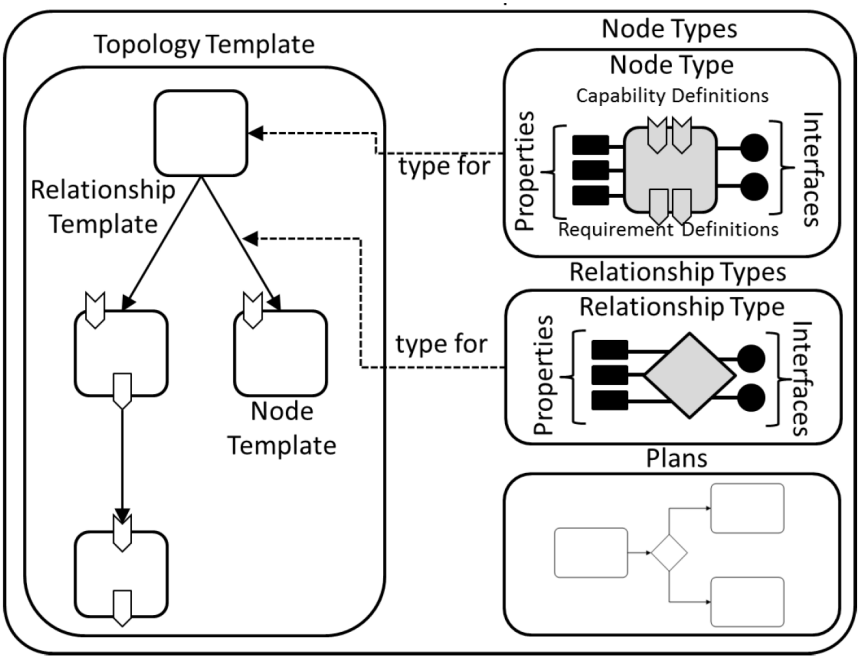

Abbildung 2.4: Aufbau von TOSCA-Topologien; Abbildung entnommen aus der TOSCA-Spezifikation [OAS13a]

Rechenressourcen (CPU-Kerne, RAM, Speicherplatz) diese besitzen soll. Ein weiteres Beispiel für Parameter sind Zugangsdaten eines Cloud-Anbieters, der Komponenten einer Topologie als Services zur Verfügung stellt.

Neben diesen Eigenschaften enthalten Node-Templates außerdem *Artefakte*. Es gibt dabei zwei Typen von Artefakten: (1) *Implementation-Artifacts* und (2) *Deployment-Artifacts*. Implementation-Artifacts sind Skripte, Installationsprogramme oder Programmcode, die das Aufsetzen einer Komponente durchführen. Typischerweise kommen hierfür Shell-Skripte zum Einsatz. Im Gegensatz zu den funktionalen Implementation-Artifacts handelt es sich bei Deployment-Artifacts um monolithische Softwarekomponenten, die für das Aufsetzen oder Betreiben einer Anwendung notwendig sind. Ein typisches Beispiel für ein Deployment-Artifact ist eine Web-Archive-Datei einer

Java-Anwendung. Ein zugehöriges Implementation-Artifact könnte dann die Logik enthalten, um dieses in einem Web-Server bereitzustellen.

Neben den Artefakten enthalten Node-Types eine Menge an *Requirements* und *Capabilities*. Diese definieren, in welchem Kontext ein instanziiertes Node-Template verwendet werden kann. Somit kann ein Node-Template mit einem bestimmten Requirement nur mit einem Node-Template verbunden werden, dessen Typ die passende Capability besitzt. Requirements und Capabilities funktionieren also nach dem Schlüssel-Schloss-Prinzip. Beispielsweise könnte das Node-Template einer Java Web-Anwendung das Requirement *"requiresWebServer"* enthalten, das definiert, dass diese Anwendung die Verbindung zu einem Web-Server-Node-Template benötigt, um lauffähig zu sein. Ein Node-Template eines Java-Web-Servers, wie zum Beispiel Apache Tomcat, könnte die erfüllende Capability *"offersWebServerCapabilities"* enthalten. Eine Verbindung dieser beiden Templates (z.B. mittels des Relationship-Templates *hosted on*) erfüllt somit das Requirement.

Relationship-Types sind gleich aufgebaut wie Node-Types, setzen jedoch statt den Komponenten die Verbindungen zwischen diesen auf. Dabei enthalten sie ebenfalls Eigenschaften, Requirements, Capabilities, und Artefakte.

Abbildung 2.5 zeigt auf der rechten Seite eine beispielhafte TOSCA-Topologie zum Aufsetzen eines browser-basierten Web-Shops. Dabei ist zu sehen, dass die PHP-basierte Web-Anwendung *Web-Shop* auf einem *Apache Web-Server* gehostet wird, dessen Betriebssystem *Ubuntu* in einer virtuellen Maschine eines *Cloud-Anbieters* (z.B. Amazon AWS) laufen soll. Des Weiteren ist diese Anwendung mit einer *MySQL-Datenbank* verbunden, um beispielsweise Produktinformationen zu speichern. Aus Skalierungsgründen ist diese Datenbank auf einer eigenen Infrastruktur gehostet (zu sehen auf der rechten Seite der Topologie in Abbildung 2.5). Die Datenbank benötigt ein zugehöriges Datenbankmanagement-System (DBMS) und, analog zum Web-Server, ein Betriebssystem, welches auf einer virtuellen Maschine betrieben wird. Des Weiteren sind beispielhaft einige der verwendeten Implementation-Artifacts und Deployment-Artifacts dargestellt. Darunter zum Beispiel die durch Implementation-Artifacts realisierten Methoden *installOS*, zur Installation eines Betriebssystems oder *installPkg*, zur Installation

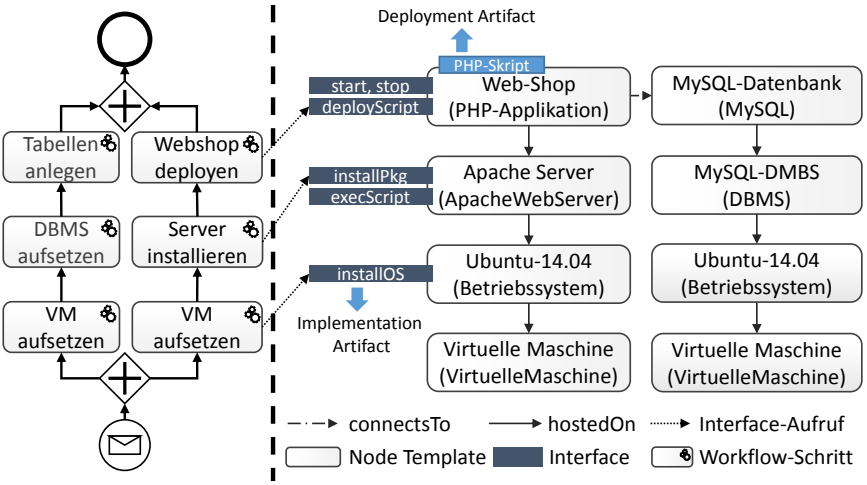

Abbildung 2.5: links: Beispiel eines TOSCA Build Plans; rechts: dazugehörige TOSCA Topologie. Die dazugehörigen Node-Types sind in Klammern angegeben (basierend auf Bildquelle aus: `http://www.opentosca.org/`)

eines Software-Pakets. Diese Artefakte werden für die Konfiguration und das Aufsetzen der Softwarekomponenten verwendet.

Ist die Topologie mit allen Node- und Relationship-Templates modelliert, wird diese mit all ihren dazugehörigen Artefakten in ein selbstbeschreibendes Archiv verpackt. Dieses wird in TOSCA *Cloud-Service-Archive (CSAR)* genannt. Das CSAR dient als Eingabe für eine TOSCA-Laufzeitumgebung, die die Orchestrierung der Anwendung durchführen kann. Ein beispielhafter Aufbau eines CSAR ist in Abbildung 2.6 zu sehen.

Nachdem die Topologie modelliert ist, kann die Orchestrierung durchgeführt werden. Diese sorgt für das Aufsetzen der in der Topologie modellierten Komponenten sowie deren Zusammenspiel. Hierfür werden in TOSCA sogenannte *Build-Plans* erstellt. Build-Plans sind Kontrollflussgraphen, beispielsweise umgesetzt mittels der Workflowsprache WS-BPEL [WCL+05]. Diese Build-Plans beschreiben, in welcher Reihenfolge die Komponenten aufgesetzt werden sollen. Dies kann aus den Verbindungen der Topologie, wie

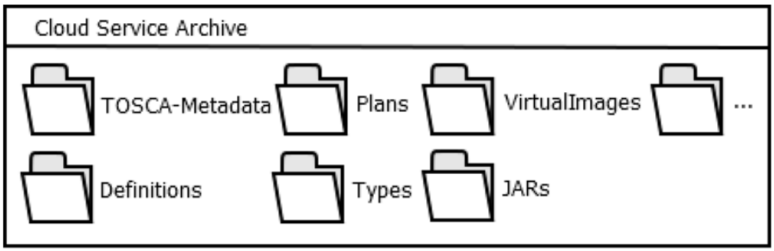

Abbildung 2.6: Beispielaufbau eines Cloud Service Archives (basierend auf [OAS13a])

hosted on oder *depends on*, geschlossen werden. Zum Beispiel muss zuerst ein Betriebssystem aufgesetzt werden, bevor eine Datenbank darauf installiert werden kann. Diese Reihenfolge muss im Build-Plan berücksichtigt werden.

Build-Plans werden in der Regel manuell erstellt. Um dies jedoch zu automatisieren, stellen Breitenbücher et al. [BBK+14a] einen Ansatz vor, der diese Build-Plans automatisch aus der Topologie generiert. Ein Beispiel für einen TOSCA-Build-Plan zum Aufsetzen der modellierten Beispielanwendung ist in Abbildung 2.5 auf der linken Seite dargestellt.

TOSCA-basierte Anwendungen können auf zwei Arten provisioniert werden: deklarativ oder imperativ. In einer *deklarativen* Laufzeitumgebung muss kein Build-Plan bereitgestellt werden, da diese bereits spezifisches Wissen über die Komponenten einer Topologie besitzt und diese somit ohne zusätzlichen Plan selbst aufsetzen kann. Jedoch ist der deklarative Ansatz nicht generisch, da nur Komponenten in der Topologie modelliert werden können, die der Laufzeitumgebung bekannt sind. Im Gegensatz dazu benötigen *imperative* Laufzeitumgebungen die Erstellung eines Build-Plans. Der große Vorteil daran ist, dass sie dafür generische Anwendungen aufsetzen können. Aus diesem Grund wird im Rahmen dieser Dissertation der imperative Ansatz gewählt, wobei der Build-Plan basierend auf den von Breitenbücher et al. [BBK+14a] vorgestellten Konzepten generiert werden kann und somit nicht manuell erstellt werden muss.

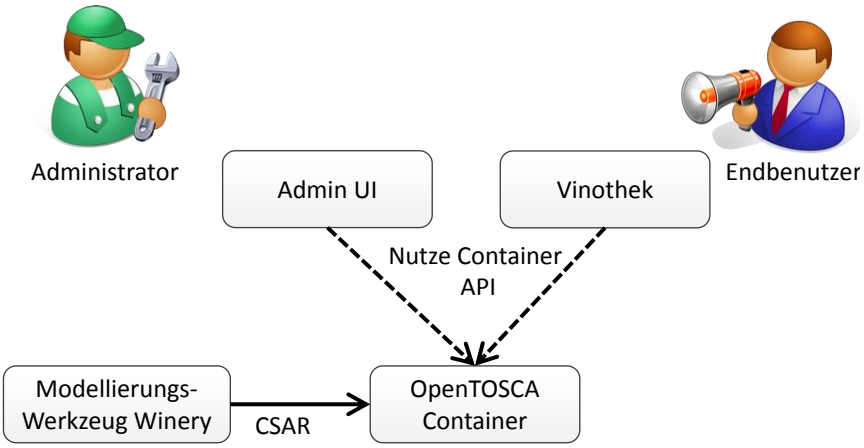

Abbildung 2.7: Architektur des OpenTOSCA-Ökosystems (basierend auf Bildquelle aus: `http://www.opentosca.org/`)

Breitenbücher et al. [BBK+14b; BBKL14a; Bre16] stellen Konzepte vor, um Cloud-Applikationen mittels dem TOSCA-Standard zu verwalten. Hierfür werden sogenannte Management-Pläne vorgestellt, mit denen provisionierte Anwendungen verwaltet werden können. Beispielsweise ermöglichen diese eine Skalierung, um ressourcenintensive Operationen ausführen zu können. Im ihren Arbeiten beschreiben Breitenbücher et al. wie diese Management-Pläne erstellt und ausgeführt werden können.

Im Rahmen dieser Dissertation wurde sich im Gegensatz zu den vielen verfügbaren Tools wie Vagrant [Has] oder Docker [Doc] für TOSCA entschieden, da dies der einzige Standard für die Provisionierung und das Management von Anwendungen ist. TOSCA bietet als offizieller Standard ein gewisses Maß an Zukunftssicherheit.

Um mögliche Werkzeuge für die Provisionierung von Anwendungen aufzufinden und zu vergleichen, wurde außerdem eine Fachstudie durchgeführt. Die Ergebnisse sind in [BRRV16] beschrieben.

2.4.3 OpenTOSCA

OpenTOSCA [BBH+13] ist ein an der Universität Stuttgart entwickeltes Open-Source-Ökosystem für den TOSCA-Standard. Dabei beinhaltet OpenTOSCA verschiedene Tools und Komponenten, mit denen TOSCA-Topologien von Anwendungen modelliert und provisioniert werden können. Abbildung 2.7 zeigt eine Übersicht des OpenTOSCA-Ökosystems.

Das Modellierungstool *Winery* [KBBL13] beinhaltet neben einer grafischen Modellierung von TOSCA-Topologien, basierend auf der Vino4TOSCA-Notation [BBK+12], ein Repository für alle im TOSCA-Standard definierten Elemente, wie zum Beispiel Node- und Relationship-Types oder Artefakte. Hierdurch bietet Winery einfache Möglichkeiten, um Topologien von Anwendungen mitsamt den dazugehörigen Node and Relationship-Types zu erstellen. Die TOSCA-Topologie kann dabei mittels *Drag & Drop* grafisch modelliert werden. Nachdem die Topologie modelliert wurde, bietet Winery einen Export an, der die Topologie, die Typen, sowie die Artefakte in ein CSAR verpackt, das für die Provisionierung verwendet werden kann. Über die *Admin-UI* kann dieses CSAR anschließend importiert werden, um für die Provisionierung der modellierten Anwendung zur Verfügung zu stehen.

Das Herzstück von OpenTOSCA stellt der sogenannte *Container* dar. Dieser steuert die Provisionierung von Anwendungen basierend auf den in Winery erstellten CSARs. Um dies zu ermöglichen, werden folgende Aktionen durchgeführt: (1) ein Build-Plan, basierend auf der Workflowsprache BPEL, wird automatisch erstellt. (2) Die Artefakte werden in passenden Laufzeitumgebungen zur Verfügung gestellt, um vom Build-Plan aufrufbar zu sein. (3) Der Build-Plan wird an eine in den Container integrierte Workflow-Engine übergeben, die diesen bei Instanziierung der Anwendung ausführt.

Die *Vinothek* [BBKL14b] stellt die Schnittstelle zum Endbenutzer dar, der interessiert daran ist, Instanzen von Anwendungen zu erstellen bzw. diese zu provisionieren. Hierfür bietet die Vinothek eine grafische Oberfläche über die Instanzen von Anwendungen mit einem Klick erstellt werden können. Dabei wird deren Build-Plan in der dazugehörigen Ausführungsumgebung des Containers ausgeführt und die Anwendungen aufgesetzt.

2.5 Internet der Dinge

Das Internet der Dinge (IoT) ist ein aufstrebendes Paradigma. Dabei wird unter IoT die Vernetzung verteilter, heterogener Geräte verstanden. Diese Geräte müssen dabei einzeln adressierbar sein [VF13]. Aufgrund der immer günstiger werdenden Hardware nimmt die Anzahl solcher vernetzten Geräte stark zu. Bereits im Jahr 2020 sollen ca. doppelt so viele Geräte existieren wie Menschen auf dem Planeten [Nor16]. Geräte sind heutzutage ausgestattet mit einer Vielzahl an Sensoren, die laufend Daten produzieren. Aufgrund der großen Anzahl der Geräte treten hier die klassischen Big-Data-Probleme Volumen, Vielfalt, und Geschwindigkeit auf. Die Analyse dieser Daten kann zu wertvollen Erkenntnissen führen, um etwa eine Produktionsstraße optimal zu steuern und größtmögliche Fehlervermeidung zu erzielen.

Es gibt verschiedenartige Ansätze, um mit der Komplexität des IoT umzugehen. In sogenannten *Data-Lakes* [Fan15] – dies sind große, skalierbare Datenspeicher – werden alle Daten erst gesammelt bevor diese analysiert werden. Durch Cloud-Technologien kann mit dem hohen Datenvolumen umgegangen werden (vgl. Abschnitt 2.1.1). Data-Lakes ermöglichen es, Analysen basierend auf einer sehr umfangreichen Datenmenge durchzuführen.

Ist es wichtig, aktuelle Daten in Echtzeit zu analysieren, kommen die Data-Lakes jedoch aus Effizienzgründen nicht in Frage. Um mit diesem Problem umzugehen, existieren streaming- bzw. eventbasierte Ansätze wie Data-Stream-Mining und Complex-Event-Processing (CEP). Dabei werden Muster auf dem Datenstrom erkannt, um bspw. Situationen zu bestimmen.

Das Internet der Dinge stellt einen wichtigen Anwendungsbereich für die Konzepte dieser Dissertation dar. Insbesondere die hohen Effizienzanforderungen sowie das große Datenvolumen eignen sich für die Evaluation der im Rahmen dieser Arbeit vorgestellten Beiträge. Aus diesem Grund involvieren viele behandelte Anwendungsfälle das Internet der Dinge.

Kapitel 3

Gesamtübersicht der Dissertation

In diesem Kapitel wird eine Übersicht über die Beiträge der vorliegenden Dissertation gegeben, die in den nachfolgenden Kapiteln detailliert beschrieben werden. Das Ziel der Beiträge ist es, ein holistisches Konzept zu schaffen, um Daten anforderungsgetrieben zu verarbeiten, wobei die Bedürfnisse von Domänennutzern im Vordergrund stehen sollen. Das bedeutet, dass einerseits die Möglichkeit bestehen soll, dass Anwender aus unterschiedlichen Domänen außerhalb der Informatik selbst bestimmen können, welche Daten in welcher Form verarbeitet und analysiert werden sollen. Andererseits soll die Datenverarbeitung abhängig von nichtfunktionalen Anforderungen dynamisch ausgeführt werden, beispielsweise unter Berücksichtigung von Kosten, Sicherheit oder Effizienz. Neben diesen beiden Hauptzielen werden darüber hinaus unterstützende Konzepte vorgestellt wie z.B. die Ressourcen-Management-Plattform, eine Plattform um Datenquellen und -senken automatisch anzubinden. Die unterstützenden Konzepte optimieren das Konzept und erhöhen somit dessen Anwendbarkeit in realen Szenarien.

Um die Konzepte zu evaluieren wurden eine qualitative Evaluation durch Fachexperten sowie eine quantitative Evaluation durch eine Integration der Beiträge in Forschungsprojekte durchgeführt (siehe Kapitel 9).

Die Beiträge der Dissertation umfassen:

B1 Domänennutzerfokussierte Modellierung der Datenverarbeitung

B2 Anforderungsgetriebene Modelltransformation und Auswahl der Ausführungsumgebung

B3 Automatische Provisionierung der Ausführungsumgebung

B4 Datenbereitstellung durch Ressourcen-Management-Plattform

B5 Anforderungserfüllende Ausführung der Datenverarbeitung.

3.1 Gesamtarchitektur

Diese Beiträge gliedern sich in eine Gesamtarchitektur ein, die in Abbildung 3.1 dargestellt ist. Dabei sind die involvierten Komponenten der Beiträge jeweils farblich hervorgehoben.

Die Modellierung der Datenverarbeitung (Beitrag **B1**, blau markiert) stellt den Einstiegspunkt dar. Dabei wird das in dieser Dissertation entwickelte *DVM* (Datenverarbeitungsmodell) erstellt, durch das Datenquellen, Datenverarbeitungsoperationen (Filter, Aggregation, Analyse), sowie Datensenken basierend auf dem Pipes-and-Filters-Modell modelliert werden können. Dabei liegt der Fokus auf der Interaktion mit Domänennutzern, die keine intensiven technischen Kenntnisse über Datenverarbeitung oder IT-Systeme besitzen. Zusätzlich soll es Anwendern mit derartigen Kenntnissen jedoch auch möglich sein, ihr Wissen durch detaillierte Parametrisierung einzubringen. Um die Modellierung weiter zu vereinfachen, wird ein Konzept zur Generierung von Handlungsanweisungen vorgestellt, das Domänennutzer bei der Modellierung unterstützt.

Der nächste Beitrag **B2** befasst sich mit der anforderungsgetriebenen Transformation des DVM (in Abbildung 3.1 orange markiert). Hierfür wurde

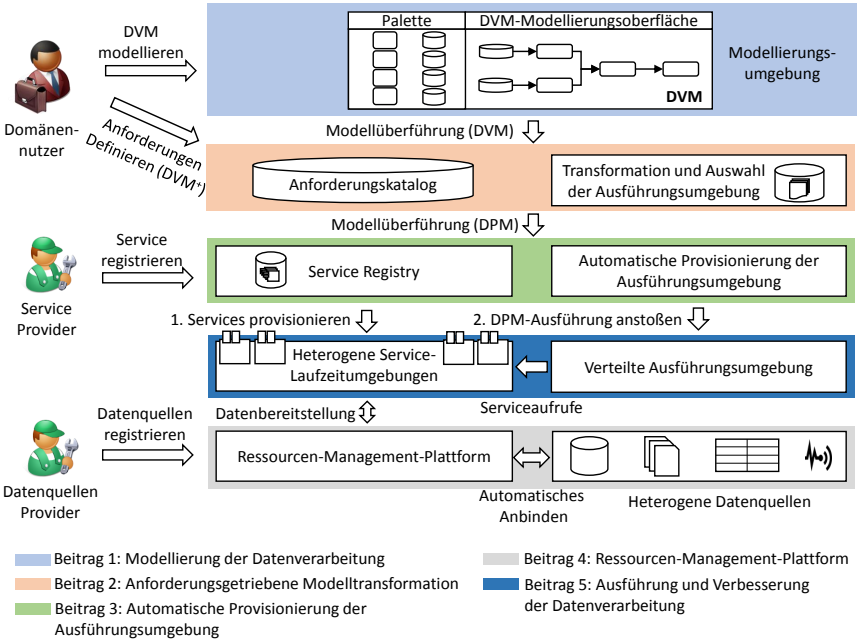

Abbildung 3.1: Beiträge der Dissertation im Überblick

die in Beitrag **B1** beschriebene Modellierung um ein Anforderungskonzept erweitert. Das bedeutet, dass Anforderungen für einzelne Operationen des DVM oder global an das gesamte DVM annotiert werden, die durch einen Anforderungskatalog in verständlicher Art und Weise beschrieben werden. Die Anforderungen umfassen beispielsweise die Kosten der Ausführung, Effizienzanforderungen, oder Datensicherheit. Durch Annotation des DVM mit den Anforderungen entsteht das sogenannte DVM^+-Modell.

Basierend auf dem DVM^+ wird anschließend die Ausführungsumgebung aus modularen Komponenten aufgebaut, damit die enthaltenen Anforderungen bestmöglich erfüllt werden können. Zum Beispiel führt die Definition einer Kostenobergrenze dazu, dass keine Komponenten ausgewählt werden, die diese Grenze übersteigen. Gelten außerdem spezifische Sicherheitsanforderungen, werden bestimmte Sicherheitsmodule hinzugenommen, die

beispielsweise eine Verschlüsselung der Daten bei der Übertragung gewährleisten. Die Auswahl und Zusammenstellung der passenden Ausführungsumgebung, basierend auf einer beliebigen Kombination der Anforderungen, wird in Beitrag **B2** beschrieben. Zusätzlich muss das modellierte DVM$^+$ in eine ausführbare Repräsentation, im Rahmen der Dissertation Datenprozessierungsmodell (kurz DPM, engl.: data processing model) genannt, überführt werden. Hierfür steht ein Repository mit Ausführungsfragmenten zur Verfügung, das als Grundlage eines modularen Aufbaus des DPM dient. Neben diesen Konzepten wird eine Möglichkeit beschrieben, neue Services zu registrieren und automatisch zu provisionieren, um die Services für die Datenverarbeitung nutzen zu können.

Nachdem das DPM automatisch erstellt wurde wird in Beitrag **B3** (in Abbildung 3.1 grün markiert) beschrieben, wie die für eine Ausführung notwendigen Softwarekomponenten dynamisch, bedarfsabhängig aufgesetzt werden können. Dabei kommt der Topology and Orchestration Specification for Cloud Applications-Standard (vgl. Abschnitt 2.4.2) zum Einsatz. Die vorige Auswahl ergibt eine Liste an Komponenten, die mittels TOSCA dynamisch provisioniert werden und somit für die Ausführung der Datenverarbeitung bereitstehen. Ein wichtiger Beitrag ist es, die Provisionierung der Komponenten vollautomatisiert durchzuführen.

Um die Verarbeitung der Daten effektiv zu ermöglichen, werden die Datenquellen im nächsten Schritt angebunden, und es wird ein uniformer Zugriff auf die Daten ermöglicht. Die Datenquellenanbindung wird in Beitrag **B4** (in Abbildung 3.1 grau markiert) beschrieben. Dabei wird ein Konzept vorgestellt, mit dem Daten aus heterogenen, verteilten Quellen vollautomatisiert extrahiert, bereinigt, und transformiert werden können. Dabei kommt unter anderem der TOSCA-Standard [OAS13a] zum Einsatz.

Nachdem die Ausführungsumgebung basierend auf den Anforderungen aufgesetzt und die Datenquellen angebunden wurden, kann die Datenverarbeitung, basierend auf dem in **B1** erstellten DVM und in **B2** transformierten DPM ausgeführt werden. Dies wird in Beitrag **B5** (in Abbildung 3.1 dunkelblau markiert) beschrieben. Die Ausführung wird dabei durch die Orchestrierung von Services mittels Workflow-Technologien ermöglicht. Dabei

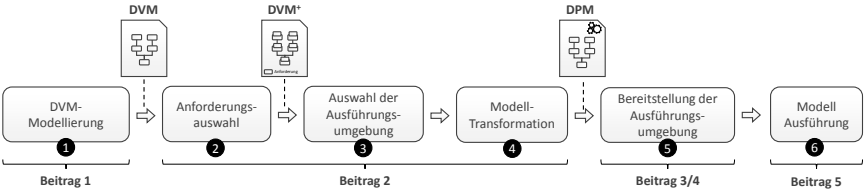

Abbildung 3.2: Methode zur Eingliederung der Beiträge der Dissertation (basierend auf [HB17])

fokussiert sich Beitrag **B5** vor allem darauf, wie die Ausführung des DPM möglichst effektiv umgesetzt werden kann. Hierfür werden verschiedene Konzepte, bspw. durch eine Verteilung der Datenverarbeitung, vorgestellt. Darüber hinaus wird beschrieben, wie eine partielle Ausführung des DVM bzw. DVM$^+$ während der Modellierungszeit umgesetzt werden kann.

3.2 Methodische Vorgehensweise

Die Beiträge lassen sich zu einer umfassenden Methode zusammenführen, die sich von der Modellierung des DVM bzw. DVM$^+$ bis zur Ausführung des DPM erstreckt. Dabei wird jeder der vorgestellten Beiträge einem oder mehreren Schritten zugeordnet. Das Gesamtziel der Methode ist es, durch die Verbindung der Beiträge eine anforderungsbasierte Modellierung und Ausführung der Datenverarbeitung zu ermöglichen. Die Abläufe sollen dabei, bis auf die Modellierung durch Domänennutzer, vollautomatisiert durchgeführt werden. Die methodische Vorgehensweise ist in Abbildung 3.2 dargestellt.

Im ersten Schritt wird das DVM modelliert. Das bedeutet, dass durch ein Pipes-and-Filters-basiertes Modell modelliert wird, welche Datenquellen und Datenoperationen zur Verarbeitung der Daten verwendet werden sollen, um ein bestimmtes Ziel (bspw. eine durchzuführende Analyse) zu erreichen. Das entstandene Modell ist ein gerichteter Graph, in dem die Knoten Datenquellen oder Datenoperationen darstellen und die Kanten den Datenfluss zwischen den Knoten. Im Fokus steht in diesem Schritt die Mo-

dellierung durch Domänennutzer, die durch die entstandenen Konzepte möglichst weitgehend unterstützt werden sollen. Schritt eins wird in Beitrag **B1**, Modellierung der Datenverarbeitung (Kapitel 4), genauer beschrieben.

Anschließend werden im zweiten Schritt Anforderungen an die Datenverarbeitung definiert. Die Anforderungen sind nichtfunktional und beschreiben wichtige Eigenschaften wie Sicherheit, Robustheit oder Effizienz der Ausführung. Hierfür wurde ein Anforderungskatalog geschaffen, mit dem Anforderungen durchsucht, ausgewählt, und kombiniert werden können. Die Anforderungen werden an Knoten des DVM bzw. global für das gesamte Modell annotiert, wodurch das erweiterte Modell DVM^+ entsteht. Die Anforderungen dienen anschließend dazu in Schritt drei die bestmögliche Ausführungsumgebung automatisch zusammenzustellen.

Nachdem die Ausführungsumgebung ausgewählt ist, kann das Datenverarbeitungsmodell in Schritt vier in ein ausführbares Modell, das DPM, überführt werden. Die Generierung des ausführbaren Modells geschieht dynamisch basierend auf den Ergebnissen aus Schritt drei. Die Details von Schritt zwei, drei, und vier werden im Beitrag **B2**, Anforderungsgetriebene Modelltransformationen (Kapitel 5), beschrieben.

In Schritt fünf wird die Ausführungsumgebung für das DPM mittels der Topology and Orchestration Specification for Cloud Applications (TOSCA) in einer virtualisierten Cloud-Umgebung bereitgestellt. Hierbei ist es wichtig, dass die Bereitstellung dynamisch, basierend auf den Ergebnissen aus Schritt drei, der Zusammenstellung der Ausführungsumgebung, geschieht. Die Details von Schritt fünf werden im Beitrag **B3**, Automatische Provisionierung der Ausführungsumgebung (Kapitel 6), beschrieben.

Im letzten Schritt sechs wird die Datenverarbeitung basierend auf dem DPM sowie der, basierend auf den Anforderungen ausgewählten und automatisch aufgesetzten, Ausführungsumgebung ausgeführt. Hierfür sind zwei Konzepte wichtig: die Ressourcen-Management-Plattform, die Datenquellen dynamisch anbindet, um deren Daten für die Verarbeitung zur Verfügung zu stellen sowie Konzepte zur verteilten Verarbeitung von Daten und zur partiellen Ausführung des DVM bzw. DVM^+. Diese werden in Beitrag **B4**, Ressourcen-Management-Plattform (Kapitel 7) und **B5** Ausführung der Da-

tenverarbeitung (Kapitel 8), genauer beschrieben.

In den nachfolgenden Kapiteln werden die Beiträge der Dissertation beschrieben. Dabei enthält jedes Kapitel eine Einführung, eine Konzeptbeschreibung, die Abgrenzung von verwandten Arbeiten, eine prototypische Implementierung sowie eine Evaluation. Anschließend werden die Beiträge zusammengefasst und qualitativ sowie quantitativ evaluiert.

Kapitel 4

Modellierung der Datenverarbeitung

Der erste Hauptbeitrag stellt die Modellierung des in dieser Dissertation entwickelten DVM (kurz für Datenverarbeitungsmodell) dar. Dabei soll es einerseits möglich sein, komplexe Anwendungsszenarien abzubilden, andererseits sollen spezielle Bedürfnisse von Domänennutzern außerhalb des IT-Bereichs berücksichtigt werden. Hierbei muss mit der Gratwanderung zwischen einer hohen Abstraktion technischer Details und einer möglichst hohen Qualität der Ergebnisse umgegangen werden. Dabei gilt die Annahme, dass durch das Einbringen von Expertenwissen die Qualität der Datenverarbeitung erhöht werden kann.

Das DVM soll definieren, welche Datenquellen genutzt werden und wie deren Daten mittels Operationen verarbeitet werden. Typische Datenverarbeitungsszenarien umfassen das Filtern, Aggregieren und Analysieren von Daten. Hierbei soll es möglich sein, beliebige Operationen miteinander zu verknüpfen. Darüber hinaus werden Konzepte vorgestellt, die die Modellierung, z.B. durch Handlungsempfehlungen, vereinfachen.

Um die beschriebenen Anforderungen an die Modellierung umzusetzen, müssen folgende Forschungsfragen beantwortet werden. Die Forschungsfragen stammen aus einer intensiven Literaturrecherche im Bereich von Datenverarbeitungsmodellen, Expertengesprächen, sowie einer Untersuchung etablierter Modellierungswerkzeuge (vgl. Abschnitt 4.5):

- Welches Modell kommt für eine effektive Modellierung von Daten und deren Verarbeitung in Frage?
- Wie können technische Details im Modell abstrahiert werden, um eine Modellierung durch Domänennutzer zu ermöglichen?
- Wie können Domänennutzer bei der Modellierung bestmöglich unterstützt werden?

Basierend auf den Forschungsfragen sind folgende Untersuchungen und Konzepte entstanden, die die gewünschten Eigenschaften der Modellierung der Datenverarbeitung ermöglichen: (1) Auswahl eines geeigneten Modells für das DVM, basierend auf spezifischen Anforderungen, (2) Konzept zur Abstraktion technischer Details, (3) Konzept zur DVM-Modellerstellung, und (4) Konzept zur Unterstützung von Modellierern, insbesondere Domänennutzer. Die Konzepte werden im Folgenden beschrieben und sind zu großen Teilen aus den Veröffentlichungen [HM16a; HRWM15] entnommen. Hierbei enthält [HM16a] ausschließlich von mir geschaffene Konzepte, wohingegen in [HRWM15] auch Konzepte der Koautoren eingeflossen sind.

4.1 Anforderungsdefinition und Auswahl des Datenverarbeitungsmodells

In diesem Abschnitt werden zuerst Anforderungen an das Datenverarbeitungsmodell definiert, die anschließend als Basis für die Auswahl eines passenden Modells dienen.

4.1.1 Anforderungen an das Datenverarbeitungsmodell

Bevor die Auswahl und Konzeption eines für das DVM passenden Modells erfolgen kann, müssen die Anforderungen an dieses definiert werden. Diese Anforderungen stammen einerseits aus den oben beschriebenen Forschungsfragen und andererseits aus einer umfangreichen Literaturrecherche im Bereich "Modellbasierte Datenverarbeitung" (vgl. Abschnitt 4.5). Des Weiteren fanden diverse Gespräche mit Experten, beispielsweise auf Fachtagungen, statt. Die Anforderungen werden im Folgenden beschrieben.

4.1.1.1 Anforderung A1: Flexibilität

Eine wichtige Anforderung ist die Flexibilität des Modells. Das Ziel ist es, beliebige Datenoperationen miteinander verknüpfen zu können. Dabei sollte die Reihenfolge der auszuführenden Datenverarbeitungsoperationen keine Rolle spielen, da dies sonst ungewünschte Einschränkungen bedeuten würde. In klassischen Ansätzen, wie dem KDD-Prozess, ist die Reihenfolge der Schritte über Extraktion, Vorverarbeitung und Analyse klar festgelegt. In diesem Beitrag ist es das Ziel, sich von einer derartigen starren Vorgehensweise zu lösen. Dadurch können beispielsweise Extraktionsschritte an beliebiger Stelle durchgeführt werden, um zusätzliche Daten hinzu zu ziehen. Die angestrebte Flexibilität ermöglicht volle Freiheit bei der Modellierung. Ein für das DVM ausgewähltes Modell muss eine derartige Flexibilität ermöglichen.

4.1.1.2 Anforderung A2: Verständlichkeit

Eine wichtige Eigenschaft des Modells ist dessen Komplexität, sowohl bei der Erstellung, als auch bzgl. der Nachvollziehbarkeit. Die Komplexität soll möglichst gering sein, um eine einfache und schnelle Modellierung zu ermöglichen. Des Weiteren ist die Nachvollziehbarkeit des Modells von großer Bedeutung, da es unter anderem das Ziel ist, das Modell zwischen Domänennutzern auszutauschen, um das enthaltene Wissen zu transferieren und damit wiederzuverwenden. Dies erfordert ein hohes Maß an Intuitivität,

das heißt, das Modell sollte auch ohne spezielle Vorkenntnisse verständlich und nachvollziehbar sein.

4.1.1.3 Anforderung A3: Abstraktionsebenen

Das Modell muss mehrere Abstraktionsebenen unterstützen, da Modellierer wie in Abschnitt 1.3.1 beschrieben unterschiedliche Kenntnisse besitzen. Einige Anwender sind in der Lage, technische Details über die Datenverarbeitung zu spezifizieren, andere hingegen benötigen eine starke Abstraktion von technischen Details, da sie nicht aus der Domäne der Datenverarbeitung stammen. Somit soll es einerseits möglich sein, eine hohe Abstraktion darzustellen und gleichzeitig für Experten die Möglichkeit bieten technische Details festzulegen.

4.1.1.4 Anforderung A4: Visualisierbarkeit

Die letzte Anforderung an das Modell ist dessen Visualisierbarkeit. Unter diesem Begriff wird verstanden, wie gut es sich für den Modellierer grafisch darstellen lässt. Eine grafische Darstellung ist essentiell für eine einfache, intuitive Modellierung. Beispielsweise ist eine Darstellung als Baum oder Graph für Modellierer intuitiver als ein rein formales oder textuelles Modell.

4.1.2 Modellauswahl

Auf Basis der beschriebenen Anforderungen wurde ein passendes Modell ausgewählt, das diese erfüllt. Die Auswahl fiel auf eine Modellierung der Datenverarbeitung basierend auf dem *Pipes-and-Filters*-Modell [Meu95]. Dieses Modell beschreibt einen gerichteten Graphen $G = (V, E)$, bestehend aus einer Menge an Knoten $\{V\}$, den Filtern, und Kanten $\{E\}$, den Pipes. Ein Beispiel eines solchen Modells ist in Abbildung 4.1 dargestellt.

Dabei stellen die Filter modulare Komponenten bzw. Operationen zur Verarbeitung (z.B. verändern, löschen, hinzufügen) von Daten dar, wohingegen die Pipes lediglich dazu dienen Daten zu übertragen. Durch die Kombination von Pipes und Filter kann so ein Datenfluss modelliert werden.

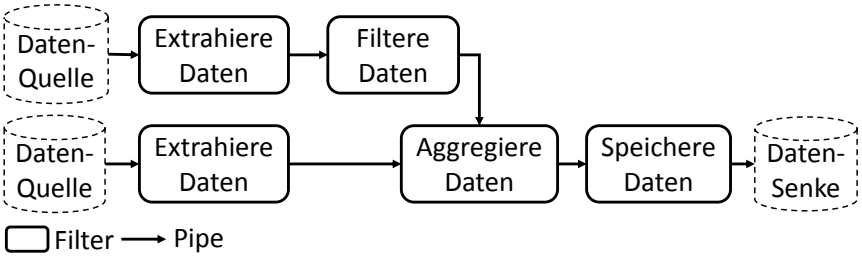

Abbildung 4.1: Beispiel eines Pipes-and-Filters-basierten DVMs

Die Idee des Pipes-and-Filters-Modells ist es, die Filter und Pipes beliebig miteinander verknüpfen zu können. Diese Eigenschaft ist für die Umsetzung der Konzepte der vorliegenden Dissertation notwendig, da es wie in Abschnitt 1.1 beschrieben das Ziel ist, Datenverarbeitungsschritte des KDD-Prozesses in beliebiger Reihenfolge ausführen zu können und nicht an die herkömmliche Reihenfolge Datenextraktion, Transformation, und Analyse gebunden zu sein. Somit kann auch die Anforderung **A1** erfüllt werden.

Eine von Shneiderman [Shn96] durchgeführte Nutzerstudie zur Modellierung boolescher Algebra, basierend auf dem Pipes-and-Filters-Modell, zeigt dessen Vorteile. Dabei nahmen 20 Probanden mit geringen Vorkenntnissen teil. Die durchgeführte Studie zeigt eine signifikante Verbesserung bzgl. Performanz bei der Modellerstellung mittels Pipes-and-Filters gegenüber einer textuellen Schnittstelle. Dabei sprachen sich alle 20 Probanden für die Wahl des Pipes-and-Filters-basierten Modells aus. Auf Basis dieser Studie sowie der weiten Verbreitung des Pipes-and-Filters-Modells (siehe Abschnitt 4.5), kann auch die Anforderung **A2** – die hohe Verständlichkeit bzw. Simplizität des Modells – erfüllt werden.

Um die im vorigen Abschnitt beschriebene Anforderung **A3** zu erfüllen, muss die "klassische" Modellierung des Pipes-and-Filters-Modell angepasst bzw. erweitert werden. Dabei darf das Pipes-and-Filters-Konzept grundsätzlich nicht verletzt werden. Hierfür wurden mehrere Abstraktionsebenen eingeführt. Dabei können einzelne Filter selbst ein Pipes-and-Filters-Modell als Untermenge enthalten. Das bedeutet, dass jeder Knoten aus der Menge

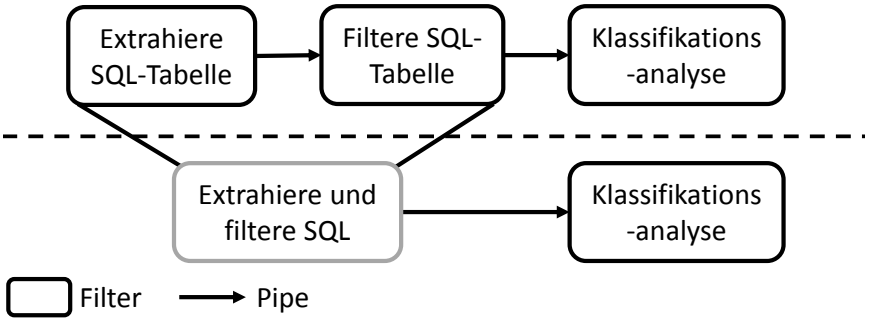

Abbildung 4.2: Verschachtelte Modellierung des Pipes-and-Filters-Modell

$\{V\}$ des Modells selbst ein eigenständiger Graph G sein kann. Diese Verschachtelung ermöglicht es, verschiedene Detailebenen einzuführen. Ein Beispiel für ein derartig verschachteltes Modell ist in Abbildung 4.2 zu sehen. Im dargestellten Beispiel soll die Extraktion, Filterung und Analyse von relationalen aus einer SQL-Datenbank stammenden Daten modelliert werden. Im oberen Bereich der Abbildung ist dies zu sehen. Dabei wurde für jede Datenoperation (Extraktion, Filterung, Analyse) ein einzelner Knoten modelliert. Um eine höhere Abstraktion zu ermöglichen, werden nun die Knoten Extraktion und Filterung in einem neuen Knoten zusammengefasst. Hierdurch kann die Modellierung durch die Reduzierung der notwendigen Knoten vereinfacht werden. Das vereinfachte Modell, unter Verwendung des neu entstandenen verschachtelten Knotens, ist im unteren Bereich der Abbildung zu sehen. Unter Verwendung dieser Technik können technische Details der Modellierung durch Abstraktionsebenen verborgen werden. Dieses Konzept basiert auf Reimann [Rei17], bei dem mehrere Abstraktionsebenen für die Modellierung wissenschaftlicher Workflows eingeführt werden, um technische Details vor Domänennutzern (in diesem Fall Wissenschaftler aus den Naturwissenschaften) zu verbergen. Durch die Abstraktion kann die Anforderung **A3** erfüllt werden.

Die vierte Anforderung **A4**, die eine hohe Visualisierbarkeit des Modells erfordert, kann durch die vorgestellte graphbasierte Modellierung ebenfalls

erfüllt werden. Insbesondere Graphmodelle können besonders gut visualisiert werden, da sie aus einer einfachen Menge an Knoten und Kanten bestehen. Für viele Menschen ist der Umgang mit visuellen, gerichteten Graphen außerdem einfach und intuitiv, da diese aus dem Alltag oder der schulischen Bildung bekannt sind. Die im vorigen Abschnitt beschriebene Studie von Shneiderman [Shn96] kommt ebenfalls zu diesem Ergebnis.

Zusammengefasst erfüllt das Pipes-and-Filters-Modell alle definierten Anforderungen, um die Modellierung des DVM zu ermöglichen. Im Folgenden wird genauer beschrieben, wie die Abstraktion technischer Details basierend auf diesem Modell erreicht werden kann.

4.2 Abstraktion technischer Details

Ein wichtiger Aspekt des angestrebten DVM ist die Abstraktion technischer Details, um eine Modellierung durch Domänennutzer zu gewährleisten. Ein erster Schritt, um eine Abstraktion zu ermöglichen, wird durch die in Anforderung **A3** geforderten und im vorigen Abschnitt 4.1.2 beschriebenen Abstraktionsebenen ermöglicht. Dabei können komplexe Verzweigungen und Details im Modell abstrahiert und somit auch dessen Größe reduziert werden. Die Frage, welche Teilgraphen des Modells abstrahiert werden sollten, kann durch Analyse bestehender Modelle beantwortet werden. Ist eine Basis von Modellen geschaffen, können wiederkehrende Muster bzw. Teilgraphen erkannt werden. Beispielsweise ist die in Abbildung 4.2 dargestellte Extraktion relationaler Daten und deren anschließende Filterung ein oft verwendetes Muster. Hierfür kann daher ein neuer Knoten “Extrahiere und filtere SQL” im Modell eingeführt werden, der von nun an für die Modellierung durch Domänennutzer verwendet werden kann.

Das Erkennen von häufig auftretenden Teilgraphen kann vollautomatisch durch Mustererkennungsalgorithmen erreicht werden. Dabei kommen sowohl klassische Verfahren, wie FP-Growth [HPY00], oder bereits auf Graphen spezialisierte Algorithmen zur Mustererkennung, wie beispielsweise von Inokuchi et. al. [IWM03], Hancock und Wilson [HW12] oder Kuramochi und

Karypis [KK05] zum Einsatz. Hierfür werden alle bisher entstandenen Modelle gespeichert und kontinuierlich analysiert. Das Ergebnis dieser Analysen sind häufig auftretende Teilgraphen, die als Vorschlag für neue Muster angesehen werden können. Ein Experte kann dann basierend auf den Vorschlägen neue Knotentypen zur Modellierung schaffen, die aufgefundene häufige Teilgraphen abstrahieren. Dadurch wird eine vereinfachte Modellierung durch eine Reduktion der Anzahl notwendiger Knoten ermöglicht.

Die Konzepte der Abstraktionsebenen gehen jedoch nicht weit genug. Das Ziel von Domänennutzern ist es, lediglich Elemente zu modellieren, die ihnen bekannt sind. Beispielsweise kennt ein Sachbearbeiter eines Unternehmens zwar ein bestimmtes CRM-Werkzeug (Kundenbeziehungsmanagement, engl.: Customer-Relationship-Management) und die Art der enthaltenen Daten. Jedoch weiß dieser nicht, dass die Daten in einer relationalen Datenbank gespeichert sind, geschweige denn, wie diese Daten mittels SQL-Queries durchsucht werden können. Derartige technische Details sollen daher vor dem Modellierer bestmöglich versteckt werden. Um eine bessere Abstraktion zu ermöglichen, werden im Rahmen dieser Dissertation sogenannte *Modellierungsmuster* [HRWM15] vorgestellt, die von technischen Details einzelner Knoten oder ganzer Subgraphen im Modell abstrahieren können. Die Grundlage hierfür stellen Business-Artefakte (auch Business-Objekte [KM17] genannt) dar. Die Inspiration für die Business-Artefakte stammt dabei von Cohn et al. [Coh+09]. Ein Business-Artefakt stellt eine abstrakte Beschreibung einer Datenquelle oder Datenverarbeitungsoperation dar.

Jedes Modellierungsmuster enthält einen *eindeutigen* Namen, ein Icon, eine Beschreibung, einen Typ (Datenquelle oder Datenoperation) und optional eine beliebige Anzahl an Parametern. Die im Rahmen der vorliegenden Dissertation entstandenen Modellierungsmuster basieren einerseits auf der Arbeit von Reimann et al. [Rei17] und andererseits auf dem Grundprinzip von Mustern (engl.: patterns), das erstmalig von Alexander [AIS+77] eingeführt wurde.

Abbildung 4.3 zeigt zwei Beispiele für derartige Modellierungsmuster. Auf der linken Seite wird eine abstrakte Beschreibung einer Datenquelle dargestellt und auf der rechten Seite einer Datenanalyseoperation. Wie in der

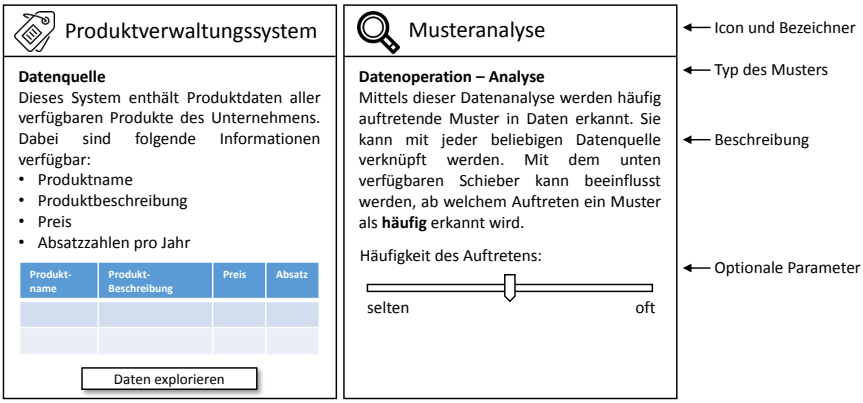

Abbildung 4.3: Beispiele für Modellierungsmuster. Links: Muster einer Datenquelle, rechts: Muster einer Datenanalyseoperation

Abbildung zu sehen, weisen diese Beschreibungen große Gemeinsamkeiten auf. Die Musterbeschreibungen sind in eine Kopfzeile und einen Inhaltsblock aufgeteilt. In der Kopfzeile des Musters steht dessen eindeutiger Name sowie ein dazugehöriges Icon, um die Wiedererkennung durch den Modellierer zu vereinfachen. Im Inhaltsblock folgt, in fett gedruckter Schrift, der Typ des Musters. Grundlegende Typen sind *Datenquelle* und *Datenoperation*, welche jedoch beliebig verfeinert werden können. In Abbildung 4.3 auf der rechten Seite gibt es beispielsweise zusätzlich den Untertyp *Analyse*, der alle Analyseoperationen gruppiert. Nach der Typdeklaration folgt eine textuelle Beschreibung des Musters. Hierbei ist es wichtig, dass die Datenquellen oder Datenverarbeitungsoperationen möglichst abstrakt bzw. in der Fachsprache der Domänennutzer beschrieben werden.

Neben der Beschreibung können außerdem optionale Parameter angegeben werden. Im Beispiel aus Abbildung 4.3 auf der rechten Seite, ist dieser Parameter die Mindesthäufigkeit des Auftretens von Mustern, damit diese bei der Analyse erkannt werden. Im Fall einer Datenquelle können zusätzlich Beispieldaten angegeben oder eine Datenexploration ermöglicht werden. Dies ermöglicht dem Modellierer neben der textuellen Beschreibung der

Datenquelle einen tieferen Einblick in die Daten zu bekommen. Hierfür können beispielsweise Datenexplorationswerkzeuge wie Elasticsearch[1] und Visualisierungen wie Kibana[2] zum Einsatz kommen.

Das Resultat der im vorigen Abschnitt beschriebenen Konzepte ist ein umfassender Katalog mit Modellierungsmustern. Dieser Katalog kann als Grundlage für die Erstellung des Pipes-and-Filters-basierten DVMs verwendet werden. Wichtig ist vor allem, dass technische Details weitestgehend abstrahiert werden und dem Modellierer trotzdem eine Übersicht über die verfügbaren Daten und Datenoperationen gewährt wird.

Die darunterliegenden Konzepte des Katalogs sind inspiriert von Fehling et al. [FBFL15; Feh15], die eine Sammlung an Mustern für das Cloud-Computing entwickelt haben. Modellierungsmuster sind wie folgt definiert:

Definition 4.1 (Modellierungsmuster)
Ein Modellierungsmuster stellt eine abstrakte Beschreibung einer oder mehrerer Datenoperationen dar. Jedes Modellierungsmuster enthält einen aussagekräftigen, eindeutigen Namen, eine textuelle Beschreibung, und ein Icon. Die Modellierungsmuster werden in einem Katalog festgehalten.

Basierend auf den Modellierungsmustern sowie der Modellauswahl kann das DVM wie folgt definiert werden:

Definition 4.2 (Datenverarbeitungsmodell (DVM))
Das Datenverarbeitungsmodell – kurz DVM – basiert auf dem Pipes-and-Filters-Modell und enthält Knoten, die Datenoperationen repräsentieren, sowie gerichtete Kanten zur Definition des Datenflusses. Jeder Knoten stellt dabei eine abstrakte Beschreibung der darunterliegenden Datenoperation per Modellierungsmuster dar. Knoten können beliebig verknüpft werden.

Nachdem in diesem Abschnitt Konzepte vorgestellt wurden, wie die für die Modellierung verwendbaren Elemente abstrahiert werden können, beschäftigt sich der nachfolgende Abschnitt mit der DVM-Modellierung basierend auf den Modellierungsmustern.

[1] https://www.elastic.co/de/products/elasticsearch
[2] https://www.elastic.co/de/products/kibana

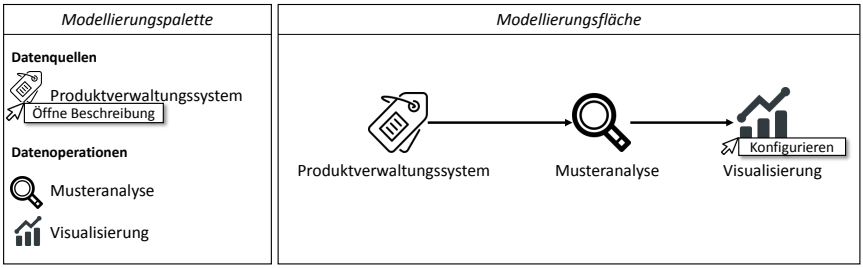

Abbildung 4.4: Beispielhaftes Werkzeug zur DVM-Erstellung

4.3 DVM-Modellierung

Auf Basis der im vorigen Kapitel beschriebenen Modellierungsmuster kann das DVM erstellt werden. Aufgrund dessen, dass die Modellierung mit unterschiedlichen darunterliegenden Datenformaten, wie XML oder JSON, ermöglicht werden kann, wird im Rahmen dieser Dissertation stets eine grafische Modellierung beschrieben. Vor allem für die Modellierung basierend auf dem Pipes-and-Filters-Modell können die entstehenden Modelle dadurch übersichtlich dargestellt werden. Abbildung 4.4 zeigt ein beispielhaftes Modellierungswerkzeug, das für die DVM-Erstellung verwendet wird.

Dabei ist zu sehen, dass alle verfügbaren Modellierungsmuster für Datenquellen und Datenoperationen auf der linken Seite in einer Palette zur Verfügung gestellt werden. Detailinformationen, also die in Abbildung 4.3 dargestellten Einträge des Katalogs für Modellierungsmuster, können über ein Kontextmenü aufgerufen und parametrisiert werden. Die in der Palette enthaltenen Knoten können dann in die auf der rechten Seite dargestellte Modellierungsfläche verschoben und dort miteinander verbunden werden. Um die in Abschnitt 4.2 beschriebene Modellierung von zu abstrahierenden Teilgraphen zu ermöglichen, kann eine Extraansicht, analog zur Graphmodellierung, geschaffen werden. Um eine klare Trennung der Verantwortlichkeiten zu ermöglichen, sollte die Modellierung von Teilgraphen in einer separaten Expertenansicht geschehen. Dadurch können Details vor Domänennutzern versteckt werden.

4.4 Unterstützung des Modellierers

In diesem Abschnitt werden Konzepte vorgestellt, die die Modellierer bei der Erstellung des DVM unterstützen. Die Konzepte umfassen (1) Vorschlagsgenerierung und (2) Wissenstransfer zwischen Modellierern.

4.4.1 Vorschlagsgenerierung

Um dem Modellierer die Erstellung des DVM zu vereinfachen, wurde ein Konzept zur Vorschlagsgenerierung während der Modellierung geschaffen. Dieses Konzept basiert auf den Assoziationen bei der Modellierung von Kanten und Knoten des Graphen. Beispielsweise kann bei der Modellierung einer Datenquelle automatisch vorgeschlagen werden, diese mit einem Filterknoten zu verbinden, um nur die für die Verarbeitung notwendigen Daten zu extrahieren. Dabei liegt die Assoziation zugrunde, dass bei vorigen Modellen diese beiden Knoten oft zusammen modelliert wurden.

Für das Bilden derartiger Assoziationen wurden in der Vergangenheit eine Vielzahl von Algorithmen zur Assoziationsanalyse geschaffen [WFHP16; WKQ+08]. Der Bekannteste ist hierbei der Apriori-Algorithmus [AIS93a; AIS93b]. Mit diesem Algorithmus können Beziehungen, beispielsweise zwischen Spalten der Tabellen relationaler Datenbanken, gefunden werden. Die Eingabe des Algorithmus sind dabei Ursprungsdaten, also in diesem Fall zuvor erstellte Modelle, die der Analyse zugrunde liegen. Die Ausgabe sind *Assoziationsregeln* vom Typ $A \rightarrow B$, wobei A und B Mengen von Items in der Datenbasis darstellen. Die Regel $A \rightarrow B$ besagt also, dass wenn die Itemmenge A in einem großen Teil der Datenbasis vorkommt, kommt mit hoher Wahrscheinlichkeit auch die Itemmenge B vor. Die Assoziationsregeln sind jeweils mit den Kennzahlen *Support* und *Konfidenz* annotiert. Dabei gibt der Support die relative Häufigkeit des Auftretens einer Regel in der Datenbasis an. Die Konfidenz gibt an, in welchem Anteil der Transaktionen in denen A vorkommt auch B vorkommt. Sowohl ein hoher Support als auch eine hohe Konfidenz sind wünschenswert. Der Apriori-Algorithmus wurde in zahlreichen Arbeiten über die Jahre immer weiter optimiert [Bor04; OPP01;

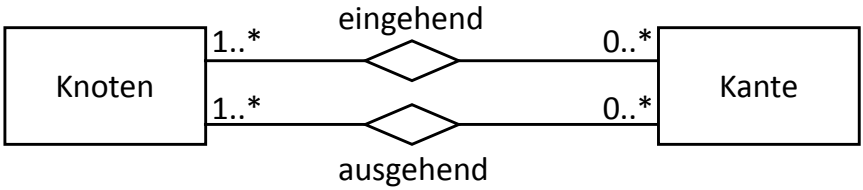

Abbildung 4.5: Entitäten-Relationen-Diagramm des DVM zur Assoziationsanalyse

YC06]. Ein oftmals genutzter Algorithmus zur Assoziationsanalyse ist außerdem FP-Growth [HPY00], der keine Generierung von *Kandidaten* benötigt und dadurch effizienter ist. In Anbetracht dessen, dass Effizienz nicht im Vordergrund steht, wurde der Apriori-Algorithmus zur Umsetzung bei der Vorschlagsgenerierung verwendet.

Hierzu muss zuerst eine umfangreiche Basis an Modellen geschaffen werden, um hohen Support und Konfidenz zu erreichen. Hierfür werden alle bisher erstellten Modelle in einer relationalen Datenbank abgespeichert. Dabei werden einerseits die verwendeten Knoten und andererseits die Kanten in Tabellen gespeichert. Die Verbindungen von Knoten und Kanten werden über eine Relationentabelle in der Datenbank gespeichert. Basierend auf dieser Datenstruktur kann der Apriori-Algorithmus angewendet werden. Dabei beschreiben die Verbindungen zweier Knoten im DVM die sogenannte Transformationsmenge über das der Algorithmus die häufig vorkommenden sog. Items bzw. Itemmengen (die häufigsten Verbindungen) bestimmt sowie den dazugehörigen Support und die Konfidenz berechnet. Das zugrundeliegende relationale Schema ist im Entitäten-Relationen-Diagramm in Chen-Notation [Che76] in Abbildung 4.5 dargestellt. Die Anwendung des Apriori-Algorithmus wird bereits in vielen Arbeiten behandelt und wird daher an dieser Stelle nicht im Detail beschrieben.

Die Ausgabe des Algorithmus sind Assoziationen zwischen den modellierten Knoten. Beispielsweise kann sich die Assoziation ergeben, dass auf die Modellierung einer Datenquelle oftmals eine Filteroperation folgt, was

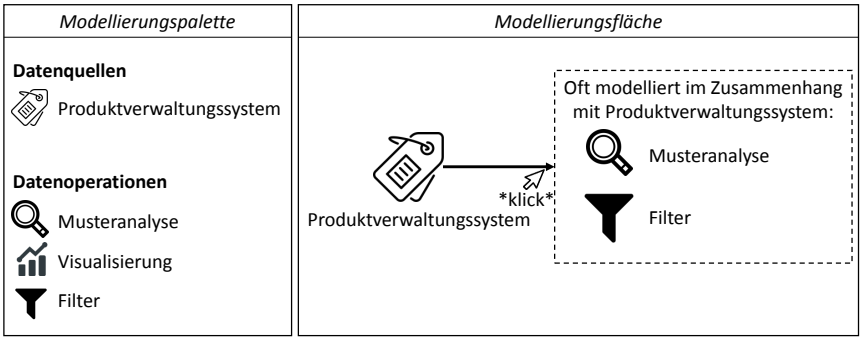

Abbildung 4.6: Beispielhafte Vorschlagsgenerierung bei der DVM-Modellierung

auch durch ein entsprechendes Modellierungsmuster repräsentiert sein kann. Um eine Assoziation als Modellierungsvorschlag zu übernehmen, muss eine hohe Konfidenz erfüllt sein. Die Konfidenz kann je nach Anwendungsfall variieren, sollte jedoch über einem sinnvoll gewählten Schwellenwert wie zum Beispiel 90 % liegen. Die gefundenen Assoziationen können auch nach dem Konfidenzwert priorisiert werden, wobei dem Modellierer beispielsweise die fünf am höchsten bewerteten Vorschläge angezeigt werden, vorausgesetzt deren Konfidenz übersteigt den angegebenen Schwellenwert. Wie eine derartige Vorschlagsgenerierung, integriert in die Modellierung, aussehen könnte, ist in Abbildung 4.6 dargestellt. Dabei ist zu sehen, dass bei der Modellierung einer Kante automatisch Vorschläge in absteigender Priorität angezeigt werden. Aus diesen Vorschlägen kann der Modellierer auswählen.

Das vorgestellte Konzept zur Vorschlagsgenerierung wurde zusätzlich um eine Komponente zur Generierung des gesamten DVMs erweitert. Bei der DVM-Generierung wird der Nutzer schrittweise, mittels eines Wizards, durch die Grapherstellung geführt. Dabei kann in jedem Modellierungsschritt entschieden werden, ob einer der vorgeschlagenen Knoten oder ein manuell ausgewählter Knoten zum Modell hinzugefügt werden soll. Die für die Vervollständigung genutzten Konzepte sind in [HBBL14] beschrieben. Durch die beschriebene Generierung von Vorschlägen und die schrittweise Durchfüh-

rung durch die DVM-Modellierung kann diese deutlich effizienter gestaltet werden. Dies wird auch durch eine von Roy Chowdhury et al. [RCN+13; RRDC12] durchgeführte Evaluation ähnlicher Ansätze belegt.

Der Prototyp der Vorschlagsgenerierung wird zur Zeit der Erstellung der vorliegenden Dissertation in einer von mir betreuten studentischen Masterarbeit implementiert [Das17].

4.4.2 Wissenserhalt bei der Modellierung

In diesem Abschnitt wird beschrieben, wie der Wissenserhalt zwischen Modellierern durch die beschriebenen Konzepte ermöglicht werden kann. Dabei ist der Wissenserhalt insbesondere innerhalb von Firmen von hoher Wichtigkeit, da Wissen durch das Ausscheiden von Mitarbeitern leicht verloren gehen kann. Im Rahmen dieses Beitrags wurden folgende Maßnahmen unternommen, um im Rahmen der Modellierung anfallendes Wissen zu sichern bzw. es zu ermöglichen, dieses zwischen den Modellierern auszutauschen.

- **Persistierung der Modelle**
  Um Analysen zur Vorschlagsgenerierung bzw. zur Abstraktion von Sub-Graphen durchzuführen, werden die erstellten Muster persistiert. Die dadurch geschaffene Basis an Modellen kann darüber hinaus einer ausgewählten Gruppe von Modellierern zur Verfügung gestellt werden. Dadurch können die Modellierer entweder aus bisher erstellten Modellen lernen oder existierende Modelle für gleiche Anwendungsszenarien wiederverwenden. Dies spart Zeit und Kosten bei der Modellierung.
- **Vorschlagsgenerierung**
  Wie in Abschnitt 4.4.1 beschrieben, werden kontinuierlich Analysen ausgeführt, um Vorschläge für die Modellierung zu generieren. Die Vorschläge tragen zur Erhaltung des Wissens bei und können zukünftige, mit der Modellierung noch nicht vertraute, Modellierer dabei unterstützen, DVMs effizient zu erstellen.
- **Subgraphen / Abstraktionsknoten**
  Mittels Analysen oder durch Experten können Abstraktionsknoten er-

stellt werden, die komplexe Subgraphen zur Lösung eines spezifischen Problems enthalten können (vgl. Abschnitt 4.2). Abstraktionsknoten werden ebenfalls im Katalog persistiert und somit bei der Vorschlagsgenerierung mit berücksichtigt. Damit lässt sich Abstraktions- bzw. Erfahrungswissen darüber dauerhaft erhalten.

4.5 Verwandte Arbeiten

In diesem Abschnitt werden verwandte Arbeiten beschrieben, die sich auf die vorgestellten Konzepte zur Modellierung des DVM beziehen.

Fehling et al. [FBFL15; Feh15] stellen ein Repository für Muster vor. Dabei basieren die Muster ebenfalls auf der Definition von Alexander [AIS+77], wobei jedes Muster eine Beschreibung, den Kontext, in dem es angewendet wird, eine Lösungsbeschreibung sowie Hinweise auf verwandte Muster enthält. Die Muster werden in einem Wikipedia-ähnlichen Repository namens *PatternPedia* [FBFL15] abgespeichert. Dies ermöglicht es, die Muster leicht auffinden zu können sowie eine Sammlung an Mustern bereitzustellen, die zwischen verschiedenen Anwendern ausgetauscht werden kann. Dabei fokussieren sich Fehling et al. auf Muster im Bereich Cloud-Computing, zeigen jedoch auch, dass das Repository generisch für beliebige Muster – beispielsweise Entwurfsmuster – eingesetzt werden kann [FBFL15].

Im Rahmen der vorliegenden Dissertation werden die Konzepte des Musterrepositorys von Fehling et al. für die vorgestellten Modellierungsmuster verwendet. Dadurch wird es Modellierern des DVM ermöglicht, schnell einen Überblick über verfügbare Modellierungselemente (Datenverarbeitungsoperationen) zu bekommen und gezielt nach Mustern zu suchen.

Reimann [Rei17] stellt in seiner Dissertation *Patterns* für wissenschaftliche Workflows (engl. scientific workflows), insbesondere im Bereich Simulation, vor. Da es vielen Wissenschaftlern außerhalb des Bereichs der Informatik Schwierigkeiten bereitet, Workflows, beispielsweise in der Workflowsprache BPEL, korrekt zu modellieren, stellt Reimann eine Abstraktion von technischen Details einzelner Knoten in Workflows vor. Dabei erstreckt sich die

Abstraktion auf mehrere Ebenen, wobei auf jeder Ebene verschiedene Rollen involviert sind. Auf oberster Abstraktionsebene ist ein Pattern ein nichtausführbarer, abstrakter Knoten im Workflow. Dessen Funktionalität sowie dessen Parametrisierung ist dem Modellierer bekannt, die konkrete Umsetzung bzw. die Formulierung in einer Workflowsprache jedoch nicht. Dieses Pattern wird in den darunterliegenden Abstraktionsebenen immer weiter konkretisiert. Die Konkretisierung wird dabei von Experten, beispielsweise Mathematiker aus dem Bereich Simulationsverfahren oder Experten aus der Informatik, durchgeführt. Dabei wird eine klare Trennung der Verantwortlichkeiten angestrebt. Eine Konkretisierung umfasst dabei eine detaillierte Parametrisierung des Knotens und kann darüber hinaus auch die Modellierung eines Untergraphen bedeuten. Dabei wird der auf oberster Ebene modellierte Knoten bei einer Konkretisierung in mehrere Knoten aufgeteilt, die wiederum einen Untergraphen bilden.

Das von Reimann vorgestellte Konzept zur Abstraktion technischer Details wurde im Rahmen dieser Dissertation ebenfalls, in ähnlicher Art und Weise, vorgestellt. Dabei liegt der Fokus jedoch auf dem vorgestellten DVM (Datenfluss), anstatt auf wissenschaftlichen Workflows (Kontrollfluss). Des Weiteren gliedern sich die vorgestellten Modellierungsmuster nicht, wie bei Reimann, in einen ausführbaren Workflow ein. Im Gegensatz dazu besteht das gesamte Modell aus nichtausführbaren, abstrakten Knoten – den Modellierungsmustern. Dadurch kann noch weiter von den technischen Details abstrahiert werden. Dieses Modell wird anschließend als Ganzes in eine ausführbare Repräsentation überführt.

Roy Chowdhury et al. [RCN+13; RDC11; RRDC12] und Rodriguez et al. [RCD+14; RDC16] stellen Konzepte vor, um oft auftretende Muster bei der Modellierung sogenannter *Mashups* [DM14] zu finden, die für eine vereinfachte Erstellung zukünftiger Modelle verwendet werden können. Dabei wird einerseits untersucht, wie oft auftretende Muster erkannt werden können und andererseits, wie diese Muster dem Modellierer bei der Modellerstellung automatisch vorgeschlagen werden können. Unter anderem führen Rodriguez et al. eine Studie durch, die untersucht, ob oft auftretende Muster mittels Crowd-Sourcing effektiv gefunden werden können [RDC16].

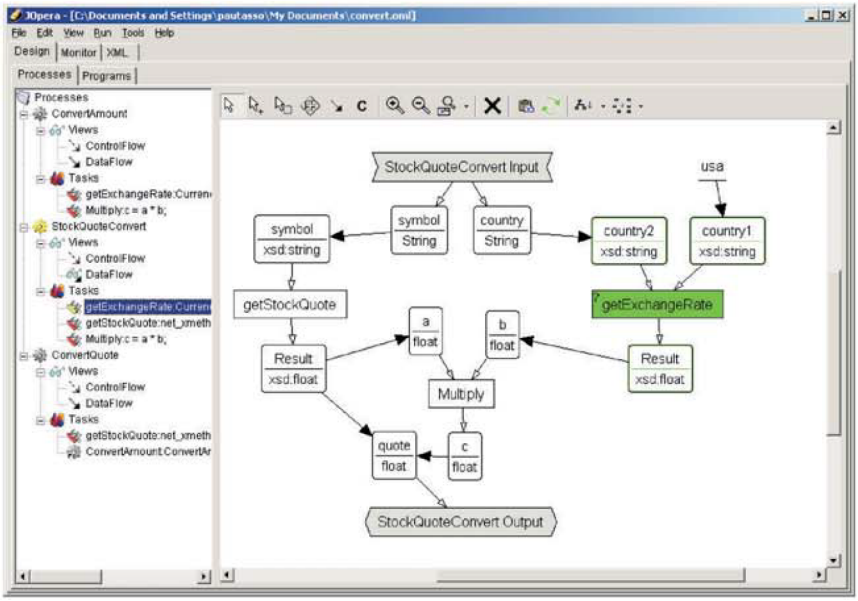

Abbildung 4.7: Screenshot des JOpera Modellierungswerkzeugs

Des Weiteren wird ein Konzept vorgestellt, das die Modellierung durch Vorschläge, basierend auf den Mustern, vereinfachen soll. Implementierung sowie durchgeführte Studien belegen die Einsetzbarkeit dieser Konzepte.

Im Rahmen dieses Beitrags der Dissertation kommen ähnliche Konzepte zum Einsatz. Statt jedoch auf Crowd-Sourcing zu setzen, werden Modellierungsempfehlungen basierend auf Data-Mining mittels einer Assoziationsanalyse auf bereits bekannten Modellen gefunden. Ähnlich wie Roy Chowdhury et al. existieren bei der Modellierung des Datenverarbeitungsmodells oft auftretende Muster, die ebenfalls mittels Data-Mining gefunden werden können. Die Muster beziehen sich im Rahmen dieser Dissertation jedoch vor allem auf das eingeführte Datenverarbeitungsmodell und nicht auf (z.B. UI-)Mashups. Die von Roy Chowdhury et al. durchgeführte Evaluierung der Modellierungsempfehlungen kann jedoch auch auf die Konzepte dieses Beitrags angewendet werden.

Pautasso und Alonso [PA05] stellen *JOpera* vor, eine visuelle Kompositionssprache. In deren Arbeit versuchen sie mit den Herausforderungen bzgl. Abhängigkeiten, die bei der Kombination von Daten- und Kontrollfluss entstehen, umzugehen. Dabei können, ähnlich zu den Konzepten dieser Dissertation, Datenflussgraphen modelliert werden, die Serviceaufrufe zur Datenextraktion und Verarbeitung repräsentieren. Dieses Datenflussmodell wird dann mit dem zugehörigen Kontrollflussgraphen synchronisiert. Das JOpera Flussmodell ist ähnlich zum DVM dieser Dissertation.

JOpera modelliert die Datenverarbeitung analog zum DVM mittel dem Pipes-and-Filters-Modell, jedoch existieren keine Abstraktionskonzepte, die den vorgestellten Modellierungsmustern entsprechen. Es werden Daten auf einer sehr technischen Ebene verarbeitet. Darüber hinaus existieren keine verschachtelten Abstraktionsebenen oder Konzepte zur Vorschlagsgenerierung. Für JOpera wurde ein Modellierungswerkzeug geschaffen, das in Abbildung 4.7 zu sehen ist.

Zusätzlich habe ich eine Untersuchung durchgeführt, bei der ähnliche Modellierungsansätze in kommerziellen oder frei verfügbaren Werkzeugen untersucht und systematisch, basierend auf einem einheitlichen Kriterienkatalog, verglichen wurden. Hierbei standen vor allem ETL- und Datenanalysewerkzeuge im Vordergrund, da diese eine ähnliche Modellierung von Datenverarbeitungsszenarien ermöglichen. Zusätzlich wurden Data-Mashup-Werkzeuge untersucht, die eine ähnliche Funktionalität bieten.

Um einen umfassenden Vergleich zu ermöglichen, wurde eine Marktanalyse durchgeführt, bei der 38 Werkzeuge mit einer Modellierung basierend auf dem Pipes-and-Filters-Modell aufgefunden wurden. Unter diesen 38 Werkzeugen sind sowohl sehr bekannte kommerzielle Modellierungswerkzeuge wie Pentaho[1], als auch in kleineren (Forschungs-)Projekten entstandene Prototypen enthalten. Im ersten Schritt wurden fünf Werkzeuge basierend auf verschiedenen Kriterien, beispielsweise Verfügbarkeit und Bekanntheit,

[1] http://www.pentaho.com/

ausgewählt. Diese sind: RapidMiner[1], Pentaho, Knime[2], Jaspersoft ETL[3] und Advanced ETL Processor Enterprise[4]. Anschließend wurden die ausgewählten Werkzeuge mittels den Kriterien Erlernbarkeit, Bedienbarkeit, Mächtigkeit, Zuverlässigkeit, Interaktivität, Fehlerbehandlung, Erweiterbarkeit und Verfügbarkeit verglichen.

Es wurde insbesondere untersucht, ob die Modellierungskonzepte dieser Dissertation in frei verfügbaren Werkzeugimplementierungen auftreten. Dabei wurde festgestellt, dass sich eine Modellierung basierend auf dem Pipes-and-Filters-Modell auch in der Praxis durchgesetzt hat und somit ein etabliertes Mittel für die Datenverarbeitung darstellt. Die Untersuchung hat gezeigt, dass viele Werkzeuge, wie zum Beispiel Pentaho, zwar eine Abstraktion von technischen Details ermöglichen, diese jedoch nicht weit genug geht, da eine komplexe Parametrisierung weiterhin notwendig ist. Die Unterstützung des Modellierers durch Modellierungsempfehlungen wurde in keinem dieser untersuchten Werkzeuge vorgefunden. Eine Abstraktion des Modells durch Untergraphen war ebenfalls nicht möglich.

Die Konzepte der Modellierungsmuster basieren teilweise auf den von Künzle et al. [Kün+11] sowie von Cohn et al. [Coh+09] vorgestellten artefaktzentrierten Ansätzen, in denen sogenannte *Business-Artefakte* geschaffen werden, deren Ziel ebenfalls eine Abstraktion technischer Details darstellt. Die Artefakte können anschließend, beispielsweise mittels den von Sun et al. [Sun+14] vorgestellten Ansätzen, auf die konkreten, technischen Artefakte abgebildet werden. Jedoch bieten sowohl Künzle et al. [Kün+11] als auch Cohn et al. [Coh+09] keinen Katalog sowie keine festgelegte Struktur für die Business-Artefakte. Die vorgestellten Konzepte sind außerdem sehr abstrakt gehalten, eine Beispielanwendung wird nicht gegeben. Sie dienen daher lediglich als Inspiration und nicht als Grundlage der Modellierungsmuster.

Zusammenfassend ergibt eine Untersuchung verwandter Arbeiten, dass die im Rahmen dieses Beitrags verwendete Modellierung mittels dem Pipes-

[1] https://rapidminer.com/
[2] https://www.knime.com/
[3] https://community.jaspersoft.com/project/jaspersoft-etl
[4] https://www.etl-tools.com/advanced-etl-processor-enterprise/features.html

and-Filters-Modell ein etabliertes Mittel ist, um die Verarbeitung von Daten zu modellieren. Dies bestätigt die Sinnhaftigkeit der Modellauswahl. Des Weiteren wurden zwar ähnliche Konzepte, insbesondere für die Abstraktion technischer Details, als auch für Modellierungsvorschläge, in verwandten Arbeiten vorgestellt. Jedoch wurden die Konzepte zur Modellierungsunterstützung bisher noch nicht auf den Bereich der Datenverarbeitung durch Domänennutzer abgebildet. Somit stellen die in diesem Beitrag erarbeiteten Konzepte ein Alleinstellungsmerkmal dieser Dissertation dar. Dazu gehören insbesondere die Abstraktion technischer Details durch Modellierungsmuster sowie die Vorschlagsgenerierung basierend auf Data-Mining-Techniken.

4.6 Implementierung der Konzepte und Evaluation

Dieser Abschnitt umfasst die Beschreibung einer prototypischen Implementierung der vorgestellten Konzepte sowie deren Evaluation.

4.6.1 Prototypische Implementierung

Die vorgestellten Konzepte wurden prototypisch implementiert und sind Open-Source, auf GitHub[1], verfügbar. Hierfür wurde ein browser-basiertes Modellierungswerkzeug namens *FlexMash* geschaffen, das basierend auf dem Framework Alloy[2] implementiert wurde. Die Implementierung umfasst dabei eine in JavaScript implementierte Benutzeroberfläche, über die eine Modellierung des DVM mittels Drag & Drop ermöglicht wird sowie das Anzeigen von Modellierungsempfehlungen und des Katalogs der Modellierungsmuster. Zusätzlich existiert eine Java-basierte Serverkomponente zum Speichern und Laden des Modells sowie zur Generierung der Modellierungsvorschläge mittels dem Apriori-Algorithmus.

Abbildung 4.8 zeigt einen Screenshot des Werkzeugs. Dabei wird im linken Bereich eine Palette dargestellt, in der die verfügbaren Modellierungsmuster zur Erstellung des DVM aufgelistet sind. Die Modellierungsmuster können

[1] https://github.com/hirmerpl/FlexMash
[2] https://github.com/liferay/alloy-ui

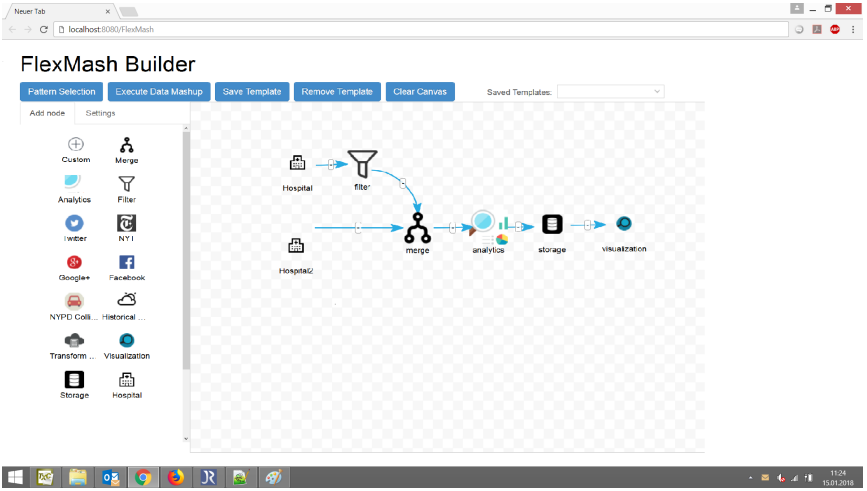

Abbildung 4.8: Screenshot des DVM-Modellierungswerkzeugs FlexMash

mittels Drag & Drop in den Modellierungsbereich auf der rechten Seite verschoben und beliebig miteinander verbunden werden. Die Parametrisierung der Knoten geschieht über den Reiter “Settings” der Palette. Über die Navigationsleiste im oberen Bereich kann auf die Beschreibungen der verfügbaren Modellierungsmuster zugegriffen werden (vgl. Abbildung 4.3). Des Weiteren kann das Modell über die Navigationsleiste abgespeichert werden und ist somit auch für andere Modellierer zugreifbar. Durch den “Ausführen”-Knopf kann die Datenverarbeitung angestoßen werden (siehe ff. Kapitel).

Im abgebildeten Beispiel in Abbildung 4.8 wurde ein Datenverarbeitungsszenario modelliert, mittels dem Daten aus zwei Krankenhäusern vereint und analysiert werden sollen. Die Daten umfassen Patientendaten, die in relationalen Datenbanken vorliegen. Diese sollen vereint und für eine Assoziationsanalyse verwendet werden. Damit kann festgestellt werden, welche Medikamenteneinnahmen zu bestimmten Nebenwirkungen führen.

Dabei wurden folgende Knoten für die Modellierung verwendet: 1) die beiden “Krankenhaus”-Knoten abstrahieren von den tatsächlich verwendeten relationalen Datenbanken, 2) ein Filterknoten, der verwendet werden kann,

um für die Analyse nicht benötigte Zeilen aus den Tabellen zu entfernen, 3) ein "Merge"-Knoten, der die Tabellen beider Datenquellen vereint, 4) ein Analyseknoten für die Durchführung der Assoziationsanalyse, 5) ein Knoten zum Abspeichern der Ergebnisse, und 6) ein Visualisierungsknoten, der die resultierenden Assoziationsregeln grafisch darstellt.

Das Modell des Szenarios kann mit diesem Werkzeug einfach durch Drag & Drop modelliert werden. Die Knoten sind dabei abstrakt gehalten und erfordern keinerlei technisches Wissen. Beispielsweise müssen für die Assoziationsanalyse nicht der maximale Support oder die Konfidenz angegeben werden. Stattdessen kann dies über einen Schieber abstrahiert werden, der die geforderte Häufigkeit des Auftretens angibt (vgl. Abbildung 4.3).

Der vorgestellte Prototyp deckt momentan noch nicht alle vorgestellten Konzepte ab, kann aber für eine Machbarkeitsstudie der vorgestellten Hauptbeiträge verwendet werden. Die Implementierung der Generierung von Modellierungsvorschlägen wird aktuell in einer von mir betreuten Masterarbeit erstellt [Das17]. Im Folgenden werden die Konzepte basierend auf dem Prototypen evaluiert.

4.6.2 Evaluation

Zur Evaluation des Beitrags habe ich im Jahr 2015 in Rotterdam sowie im Jahr 2016 in Lugano an der *Rapid Mashup Challenge* im Rahmen der International Conference on Web Engineering (ICWE) teilgenommen[1]. Die Ergebnisse wurden in [HM16a] für das Jahr 2015 und in [HB17] für das Jahr 2016 veröffentlicht. Bei der Challenge wurden die Konzepte sowie eine Demonstration des Prototypen FlexMash einer Fachjury sowie einem Expertenpublikum aus dem Bereich Web-Entwicklung und Datenverarbeitung vorgestellt. Zur Vorstellung der Konzepte und des Werkzeugs standen hierfür jeweils 10 Minuten zur Verfügung. Hierbei konnten die Modellierungskonzepte qualitativ durch eine Fachjury evaluiert werden.

Die Evaluation fand basierend auf diversen Kriterien statt, die sowohl

[1] Webseite des Jahres 2015: `http://mashup.inf.usi.ch/challenge/2015/`; Webseite des Jahres 2016: `http://challenge.webengineering.org/`

die darunterliegenden Konzepte als auch deren Umsetzung bewerten. Die Details der Evaluation sind in Abschnitt 9.1 beschrieben.

4.7 Zusammenfassung

In diesem Beitrag wurden Konzepte zur Modellierung des DVM beschrieben, bei dem vor allem die Abstraktion technischer Details sowie die Einfachheit der Modellierung im Vordergrund stehen. Im ersten Schritt wurden verschiedene Anforderungen an das Modell definiert, die aus einer umfangreichen Literaturrecherche stammen. Basierend auf den Anforderungen wurde ein passendes Modell ausgewählt. Dieses basiert auf dem Pipes-and-Filters-Modell und ermöglicht die einfache Verknüpfung von Datenquellen und Datenverarbeitungsoperationen.

Um zusätzlich von technischen Details zu abstrahieren wurde einerseits das Konzept von Subgraphen vorgestellt und andererseits von Modellierungsmustern, die eine abstrakte Beschreibung komplexer Datenoperationen darstellen. Die Modellierung wurde darüber hinaus durch die Generierung von Vorschlägen während der Modellierungszeit für Domänennutzer optimiert. Durch die entstandenen Konzepte kann das in den Modellen enthaltene Wissen persistiert und transferiert werden, was in Abschnitt 4.4.2 beschrieben wird. Die entstandenen Konzepte wurden prototypisch implementiert und durch eine Demonstration bei den Rapid Mashup Challenges 2015 und 2016 vor einer Fachjury evaluiert. Dabei konnte die Anwendbarkeit bzw. die Sinnhaftigkeit der Beiträge bestätigt werden.

Das nächste Kapitel baut auf dem DVM-Modellierungskonzept auf und beschreibt, wie DVMs in ein ausführbares Modell überführt werden können.

Kapitel 5

Anforderungsgetriebene Modelltransformationen

Das im vorigen Kapitel 4 beschriebene DVM wird in ein ausführbares Modell, das DPM, transformiert. Die Besonderheit in dieser Dissertation ist es, dass die Ausführungsumgebung des Zielmodells der Transformation dynamisch und anforderungsgetrieben ausgewählt wird und nicht, wie bei bisherigen Lösungen, statisch vorgegeben ist. Um entscheiden zu können, auf welche Art das Datenverarbeitungsmodell ausgeführt werden soll, muss die Modellierung um ein Anforderungskonzept erweitert werden. Hierfür wird das DVM mit Anforderungen annotiert und somit in das erweiterte Modell DVM$^+$ überführt. Dieses wird im Folgenden genauer vorgestellt.

Die in diesem Abschnitt beschriebenen Konzepte sind zum großen Teil den Veröffentlichungen [HB17; Hir15; Hir17; HM16a] entnommen. Dabei stammen die Konzepte der Veröffentlichungen [Hir15; Hir17; HM16a] ausschließlich von mir, wohingegen in [HB17] zusätzlich Konzepte beschrieben werden, wie die Modellierung und Ausführung der Datenverarbeitung interaktiv gestaltet werden kann. Dies ist nicht Teil des Beitrags.

5.1 Anforderungen an die Datenverarbeitung

Um eine passende Art der Datenverarbeitung auswählen zu können, muss das DVM-Modell aus Kapitel 4 mit Anforderungen des Modellierers annotiert werden, wodurch das DVM$^+$ entsteht. Hierbei stehen nichtfunktionale Anforderungen an die Ausführung im Vordergrund, wie zum Beispiel, dass sie möglichst sicher, effizient, robust oder kostengünstig ablaufen soll. Die Anforderungen variieren stark mit den Anwendungsfällen, die durch das DVM ausgeführt werden sollen. Beispielsweise ist es bei einer explorativen Modellerstellung, bei der das Modell mehrfach ausgeführt und angepasst wird, wichtig, dass die Kosten möglichst gering gehalten werden, auch wenn dadurch Robustheit oder Effizienz der Ausführung sinken. Hingegen ist es für langlaufende Datenanalysen, beispielsweise durchgeführt von einem Unternehmen, von großer Wichtigkeit, dass diese möglichst robust ablaufen, was wiederum zu höheren Kosten, beispielsweise durch Replikation, führen kann. Die Beispiele zeigen, dass sich die Anforderungen gegenseitig beeinflussen (z.B. Kosten und Effizienz) und es nicht möglich ist, alle Anforderungen gleichzeitig voll zu erfüllen. Zur Definition der Anforderungen wird daher ein Gewichtungsmodell geschaffen, das deren Einflussfaktoren abbildet.

Bei der Definition von Anforderungen ist außerdem zu beachten, dass die in Abschnitt 1.3.1 beschriebenen Bedürfnisse von Domänennutzern berücksichtigt werden. In Folge dessen sollte die Definition von Anforderungen bei Bedarf auch ohne umfangreiche technische Kenntnisse möglich sein. Um eine abstrakte Anforderungsdefinition zu ermöglichen, wurden verschiedene Abstraktionsebenen für die in Abschnitt 1.3.1 vorgestellten Rollen eingeführt. Domänenexperten *ohne jegliche IT-Vorkenntnisse*, bzw. lediglich mit Kenntnissen aus der Statistik, wird es ermöglicht, Anforderungen an die Ausführung zu definieren, ohne jedoch die technischen Details der Umsetzung zu kennen (z.B. Verschlüsselungstechniken zur Steigerung der Sicherheit).

Durch eine derartige Abstraktion können die Anforderungen jedoch unscharf werden, beispielsweise wenn lediglich definiert wird, dass die Ausführung möglichst *effizient* oder *robust* sein soll, jedoch keine konkreten Zeitgrenzen definiert werden. Domänenexperten *mit IT-Kenntnissen* bekom-

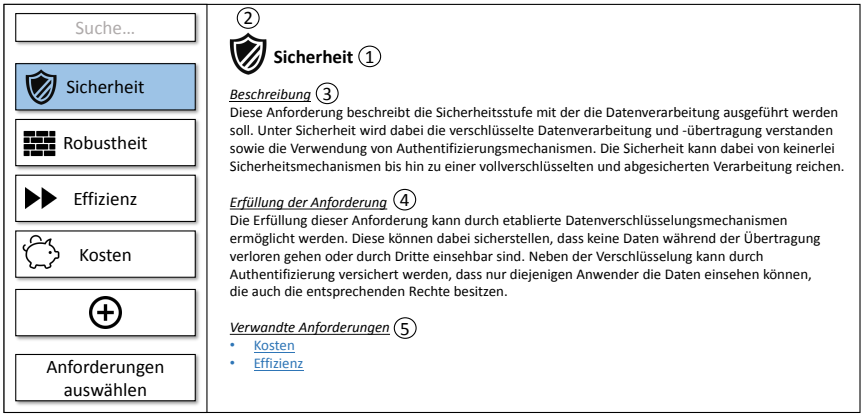

Abbildung 5.1: Beispieleintrag “Sicherheit” im Anforderungskatalog (basierend auf [HM16a])

men daher die Möglichkeit, die Anforderungen auch technisch exakt zu definieren. Hierfür werden *Policys* verwendet, also strukturierte Anforderungsbeschreibungen, die die Detailanforderungen an die Ausführung definieren. In den nachfolgenden Abschnitten werden beide Möglichkeiten zur Anforderungsdefinition genauer beschrieben. Grundlage für beide Ansätze ist ein *Anforderungskatalog*, der als erstes beschrieben wird.

5.1.1 Anforderungskatalog

Um mögliche Anforderungen an die Ausführung zu dokumentieren, wurde ein Anforderungskatalog [HM16a] geschaffen, der eine Navigation, analog zu Sammlungen wie z.B. Wikipedia, ermöglicht. Der Anforderungskatalog ist ähnlich aufgebaut wie der in Kapitel 4 vorgestellte Katalog für die Modellierungsmuster.

Der Anforderungskatalog enthält für jede mögliche Anforderung einen Eintrag, der in Abbildung 5.1 dargestellt und wie folgt aufgebaut ist: (1) ein eindeutiger, ausdrucksvoller Bezeichner, der die Anforderung beschreibt, (2) ein Icon, um den Wiedererkennungswert zu gewährleisten, (3) eine

Beschreibung der Anforderung selbst, (4) eine Beschreibung, wie die Anforderung bei der Ausführung erfüllt werden kann, und (5) verwandte Anforderungen, deren Katalogeintrag direkt über einen Verweis aufgerufen werden kann. Die Einträge im Anforderungskatalog entsprechen gängigen, oft wiederkehrenden Anforderungen, die intensiv in relevanter Fachliteratur diskutiert wurden und für die eine dazugehörige Lösung für deren Umsetzung entwickelt wurde (z.B. bestimmte Verschlüsselungstechniken). Der Anforderungskatalog ist darüber hinaus beliebig erweiterbar.

Ein Beispieleintrag für die nichtfunktionale Anforderung *Sicherheit* ist in Abbildung 5.1 dargestellt. Dabei ist zu sehen, dass auf der linken Seite durch die Anforderungen navigiert sowie neue Einträge eingefügt werden können. Auf der rechten Seite ist die Beschreibung einer Anforderung zu sehen, die angezeigt wird, sobald sie selektiert wurde. Der Katalog dient bisher lediglich zur Dokumentation der möglichen Anforderungen.

Der Anforderungskatalog kann wie folgt definiert werden:

Definition 5.1 (Anforderungskatalog)
Der Anforderungskatalog enthält alle nichtfunktionalen Anforderungen, die bei der Datenverarbeitung auftreten können. Jede Anforderungsbeschreibung besteht aus einem Bezeichner, einem Icon, einer Beschreibung der Anforderung sowie einer Beschreibung, wie sie erfüllt werden kann.

Die Auswahl und Konfiguration der Anforderungen an die Ausführung erfolgt im nächsten Schritt. Für die Definition von Anforderungen an die Ausführung wurden, wie oben beschrieben, zwei Möglichkeiten geschaffen: (1) eine abstrahierte Anforderungsauswahl für Domänennutzer ohne umfangreiche technische Kenntnisse und (2) eine policy-basierte Detailauswahl für technische Experten. Anforderungen können sowohl *global* für das gesamte Datenverarbeitungsmodell definiert werden, als auch *lokal* für einzelne Knoten, beispielsweise wenn bestimmte, sensible Daten in einem Verarbeitungsschritt verschlüsselt werden sollen oder wenn eine, einen Flaschenhals darstellende, Datenoperation beschleunigt werden soll.

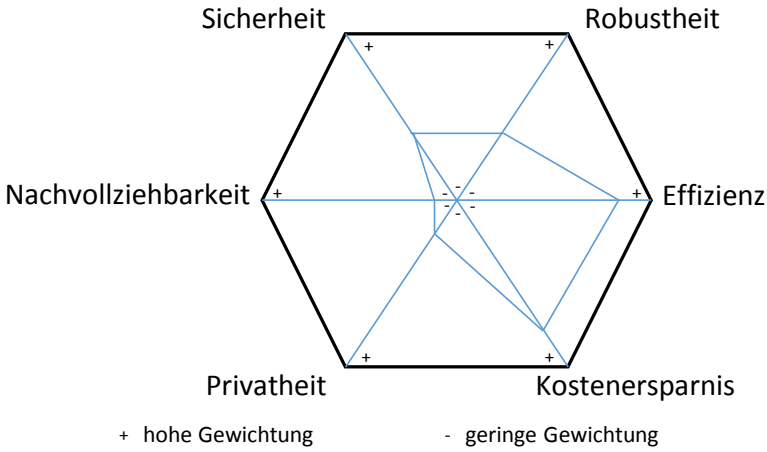

Abbildung 5.2: Mögliche Gewichtungsverteilung bei der abstrahierten Anforderungsauswahl

5.1.2 Abstrahierte Anforderungsdefinition

Um es auch Anwendern ohne umfangreiche technische Kenntnisse zu ermöglichen, möglichst passende Anforderungen zu definieren, wurde eine abstrahierte Möglichkeit zur Anforderungsdefinition geschaffen. Diese basiert auf einem Gewichtungsmodell, welches dem Modellierer anfangs eine gewisse Punktzahl (z.B. 100), global für das Gesamtmodell, zur Verfügung stellt. Anschließend kann der Anwender Punkte auf die Anforderungen im Katalog verteilen. Je nachdem, wie viele Punkte vergeben werden, desto höher wird die Anforderung priorisiert. Dies macht insbesondere deshalb Sinn, da naive Anwender oftmals erwarten, es können alle Anforderungen gleichzeitig erfüllt werden. Jedoch widersprechen sich viele Anforderungen bzw. es müssen Kompromisse gemacht werden (z.B. zwischen Effizienz, Robustheit, Sicherheit und Kosten). Durch das Punktesystem wird das Bewusstsein der Anwender diesbezüglich geschärft. Außerdem können so keine unrealistischen Anforderungsdefinitionen entstehen, die bei der Ausführung nicht erfüllt werden können. Eine mögliche Punkteverteilung könnte wie in Abbildung 5.2 dargestellt aussehen. Hierbei ist zu sehen, dass in diesem

Beispiel eine hohe Effizienz und möglichst geringe Kosten für die Ausführung wichtig sind. Diese Vorgehensweise zur Definition der Anforderungen ist für Anwender ohne umfangreiche technische Kenntnisse intuitiv.

Des Weiteren können auch einzelne Knoten im Modell mit Anforderungen annotiert werden, die lokal, also nur für deren Ausführung, gelten. Hierfür wird ebenfalls eine Punktzahl zur Verfügung gestellt, die auf die Anforderungen des Katalogs für jeden Knoten einzeln verteilt werden kann.

Werden sowohl globale Anforderungen für das Gesamtmodell als auch lokale Anforderungen an Einzelknoten annotiert, werden lediglich die lokalen Anforderungen beachtet. Die globalen Anforderungen gelten dann jeweils nur für Knoten, für die keine lokalen Anforderungen definiert wurden. Dies ist notwendig, um Konflikte zwischen globalen und lokalen Anforderungen zu vermeiden. Beispielsweise könnte in den globalen Anforderungen definiert sein, dass die Kosten eine bestimmte Grenze nicht überschreiten dürfen. In den lokalen Anforderungen könnten hingegen höhere Kosten erlaubt sein. Derartige Konflikte lassen sich nicht automatisch auflösen.

Abbildung 5.3 zeigt, wie die Anforderungsdefinition in die Modellierung integriert werden kann, um das DVM^+ zu erstellen. Es gibt hierbei einerseits die Möglichkeit, den Anforderungskatalog einzusehen (1) sowie globale Anforderungen wie in Abbildung 5.2 dargestellt zu definieren (2). Des Weiteren kann bei der Auswahl eines einzelnen Knotens (3) die lokale Anforderungsdefinition durchgeführt werden. In einer Extraansicht (4) können die Punkte auf die jeweiligen Anforderungen verteilt werden. In diesem Beispiel wird die Anforderungsauswahl durch eine grafische Eingabe realisiert, andere Darstellungen für die Anforderungsauswahl, beispielsweise textuell, sind ebenfalls möglich.

Selbstverständlich kann durch diese Vorgehensweise zur Anforderungsdefinition lediglich eine Tendenz des Anwenders erfragt werden. Die Definition der Anforderung bleibt jedoch unscharf. Eine große Herausforderung ist außerdem, dass Anwender bei manchen Anforderungen klare Vorstellungen haben, beispielsweise maximale Kosten, andere Anforderungen hingegen nicht bewerten können, wie zum Beispiel die Robustheit der Ausführung. Die Anforderungen an die Kosten stellen einen Sonderfall dar, da sie beinahe

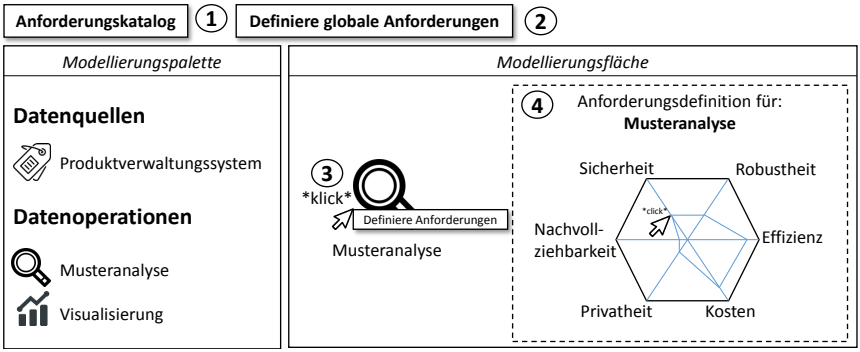

Abbildung 5.3: Abstrahierte Anforderungsdefinition bei der DVM$^+$-Modellierung

immer bekannt sind. Aus diesem Grund wird es, falls gewünscht, dem Modellierer ermöglicht, die Maximalkosten der Ausführung anzugeben. Jedoch birgt dies die Gefahr, dass das Modell nicht mit den angegebenen Kosten ausgeführt werden kann, da Domänennutzer in der Regel keine Vorstellung haben, wie kostenintensiv die Datenverarbeitung sein kann. Dies kann dazu führen, dass eine Ausführung nicht möglich ist. Alternativ könnten die Maximalkosten von einem Experten der Datenverarbeitung angegeben werden, sodass der Modellierer mittels der beschriebenen Punktevergabe, lediglich definiert, wie nah die Kosten an den festgelegten Maximalkosten liegen dürfen.

Zusammenfassend kann die abstrahierte Auswahl von Anforderungen dazu führen, dass die konkreten Anforderungen der Anwender nicht immer voll erfüllt werden. Trotzdem kann davon ausgegangen werden, dass eine Tendenz des Anwenders erkennbar ist. Des Weiteren kann durch eine explorative Vorgehensweise bei mehrfacher Ausführung das gewünschte Ergebnis durch Anpassen der Anforderungen angenähert werden.

Um es darüber hinaus zu ermöglichen, diese Unschärfe zu beseitigen, wurde zusätzlich eine konkrete, policy-basierte Anforderungsauswahl geschaffen, die im Folgenden beschrieben wird.

5.1.3 Policy-basierte Anforderungsdefinition

Neben der abstrahierten Auswahl von Anforderungen wurde für Anwender mit umfangreichen technischen Kenntnissen zusätzlich eine Möglichkeit geschaffen, um Anforderungen, sowohl global als auch lokal, konkret mittels Policys zu definieren. Die Voraussetzung für die Definition einer derartigen Policy ist, dass die Anforderungen quantifizierbar sind. Das bedeutet, dass für jede der Anforderungen ein konkreter Wert angegeben werden kann. Für die Anforderung bzgl. Kosten ist dies einfach, da sich die Kosten in Zahlenwerten ausdrücken lässt. Ähnlich verhält es sich mit Anforderungen bzgl. Robustheit, bei denen typischerweise die Anzahl maximal zugelassener Ausfallstunden (beispielsweise pro Jahr) in sogenannten Service Level Agreements [KT11] angegeben werden sowie Anforderungen bzgl. Effizienz, bei der die maximale Laufzeit der Datenverarbeitung angegeben werden kann. Wichtig bei dem vorgestellten Ansatz ist es, dass alle Anforderungen quantifizierbar gemacht werden, was im Anforderungskatalog, beispielsweise in einer gesonderten Expertenansicht, beschrieben werden kann.

Um die Anforderungen an die Ausführung konkret zu definieren, wurde in [Hir17] ein Policyformat, basierend auf der XML-Schema-Definition, geschaffen. XML ist ein etabliertes Format zur Definition von Policys (vgl. zum Beispiel WS-Policy [WCL+05]). Eine beispielhafte Policy für die Anforderungsdefinition ist in Listing 5.1 dargestellt.

Hierbei ist zu sehen, dass jede Anforderung mit einem *schemaLocation*-Attribut annotiert ist, in dem das strukturgebende Schema referenziert wird. Dadurch wird die Erweiterbarkeit der Policy gewährleistet, indem ein Schema für neue Anforderungen definiert werden kann. Die Schemata sind im Anforderungskatalog bei den entsprechenden Einträgen hinterlegt.

Bei der Definition von konkreten Anforderungen mittels Policys können sich jedoch leicht Widersprüche ergeben, beispielsweise, dass die Anforderungen mit den definierten Kosten nicht umgesetzt werden können. Somit ist eine Anforderungsdefinition durch Policys nur durch Anwender mit umfangreichem Wissen über die Verarbeitung von Daten möglich. Widersprüche in der Policy können zu ungewünschten Ergebnissen führen, zum Beispiel

```
<Policy>
  <Anforderungen>
    <MaximalKosten
      xsi:schemaLocation="http://www.example.com/costs.xsd">
      <Waehrung> Euro </Waehrung>
      <Betrag> 1500 </Betrag>
      <Zeitraum> Jahr </Zeitraum>
    </MaximalKosten>
    <Sicherheit
      xsi:schemaLocation="http://www.example.com/security.xsd">
      <Verschluesselung>
        <Verfahren> Advanced Encryption Standard </Verfahren>
      </Verschluesselung>
      <Authentifizierung> false </Authentifizierung>
    </Sicherheit>
    <!-- Definition weiterer Anforderungen -->
  </Anforderungen>
</Policy>
```

Listing 5.1: Policy zur Anforderungsdefinition in XML

einer zu geringen Effizienz oder zu hohen Kosten.

Die Policys können sowohl global für das gesamte Modell angegeben werden, als auch lokal für einzelne Knoten. Die Möglichkeit der abstrakten Anforderungsdefinition kann darüber hinaus ebenfalls, beispielsweise für einzelne Knoten, genutzt werden. Die Modellierung des DVM$^+$ kann durch eine Annotation der enthaltenen Knoten mittels den vorgestellten Policys umgesetzt werden.

5.2 Anforderungsbasierte Auswahl der Ausführungsumgebung

Die Datenverarbeitung soll durch die Komposition von Softwarekomponenten realisiert werden, die bestimmte Funktionalitäten zur Verfügung stellen. Derartige Softwarekomponenten werden als Services (vgl. Kapitel 2) bezeichnet, die ihre Funktionalitäten über standardisierte Schnittstellen zur Verfügung stellen. Diese Funktionalitäten werden zum Beispiel realisiert durch die Implementierung von Analysealgorithmen oder Datenbankanwendungen, die (Zwischen-)Ergebnisse persistieren. Jeder Knoten im Datenverarbeitungs-

modell wird durch einen oder mehrere Services repräsentiert.

Für die Ausführung eines bestimmten DVM stehen oftmals viele verschiedene Servicekombinationen zur Verfügung, die die modellierte Funktionalität abdecken. Zusätzlich müssen jedoch auch die, im DVM$^+$ annotierten, nichtfunktionalen Anforderungen bei der Auswahl von Services beachtet werden. Beispielsweise kann es für einen Knoten für die Speicherung von Ergebnissen zwei verschiedene Services geben. Dabei verschlüsselt einer der Services die Daten bei der Speicherung, der andere jedoch nicht. Wurde die Anforderung "Sicherheit" an dem Knoten annotiert, sollte ein Service, der eine Verschlüsselung anbietet, für die Ausführung ausgewählt werden. Jedoch ergibt sich hier ein komplexes Problem, da sich die nichtfunktionalen Anforderungen, wie in Abschnitt 5.1 beschrieben, gegenseitig beeinflussen können. Im Folgenden wird beschrieben, wie die am besten geeignete Kombination an Services gefunden werden kann, um die Ausführung zu ermöglichen.

Bei den auszuwählenden Services zur Ausführung des DVM existieren zwei Rollen: (1) der Service Provider, der die Services bereitstellt und (2) der Service Consumer, der die Services aufruft und die Ergebnisse weitergibt. Auf den Service Provider wird an dieser Stelle nicht im Detail eingegangen. Es kann sich beispielsweise um einen Cloudanbieter handeln, der ein Software-as-a-Service-Angebot mit einem dazugehörigen Kostenmodell bereitstellt oder um einen frei verfügbaren Service.

Services bestehen allgemein aus zwei Teilen: einer Implementierung und einer Beschreibung. Die Implementierung kann in einer beliebigen Programmiersprache vorliegen, da die Schnittstellen der Services über standardisierte Protokolle (bspw. HTTP) zugreifbar sind und die konkrete Implementierung verborgen wird. Des Weiteren besitzt jeder Service eine Beschreibung, die dessen Funktionalität sowie dessen nichtfunktionale Eigenschaften (z.B. Kosten, Service Level Agreements, etc.) beschreibt. Die nichtfunktionalen Eingenschaften können beispielsweise mittels Web Service Policys beschrieben [WCL+05] werden. Diese Beschreibungen sind von großer Wichtigkeit, um den zu den angegebenen nichtfunktionalen Anforderungen des DVM$^+$ am besten passenden Service auswählen zu können.

Alle Servicebeschreibungen werden in einem UDDI-ähnlichen Repository

(vgl. Abschnitt 2.3) – der Service-Registry (siehe Abbildung 3.1) – zur Verfügung gestellt. Die Service-Registry stellt die Grundlage für das Auffinden passender Services dar. Für das Auffinden der Services werden die Knoten aus dem DVM$^+$ – also die annotierten Modellierungsmuster – betrachtet. Für jeden Knoten wird ein passender Service in der Service-Registry gesucht. Als Suchkriterien werden einerseits die Eigenschaften der Knoten verwendet, zum Beispiel dessen Typ oder Parameter, also die funktionalen Anforderungen (z.B. Datenanalyse mittels Assoziationsanalysen) und zum anderen auch die nichtfunktionalen Anforderungen (z.B. Kosten, Sicherheit), die entweder global für das gesamte DVM$^+$ oder lokal für einzelne Knoten definiert wurden. Wurde das DVM nicht mit Anforderungen annotiert, findet keine Auswahl bezüglich nichtfunktionalen Anforderungen statt.

Für die Auswahl der Services wurde ein Algorithmus geschaffen, der die Eigenschaften der Services mit den Anforderungen aus dem DVM$^+$ abgleicht. Dieser Algorithmus ist in Listing 5.2 in Java-ähnlichem Pseudocode dargestellt. Im ersten Schritt (Zeile 2) wird das DVM$^+$ "gesäubert", indem alle Subgraphen aufgelöst und darin enthaltene Knoten dem Modell an der entsprechenden Stelle hinzugefügt werden. Somit existiert nach Schritt eins nur noch eine Hierarchieebene.

Daraufhin werden alle Knoten und Kanten aus dem DVM$^+$ extrahiert und in einer Liste gespeichert (Zeile 4). Anschließend wird diese Liste traversiert, um für jeden Knoten einen passenden Service aufzufinden (Zeile 7). Hierfür werden zuerst alle Services des entsprechenden Knotentyps aus der Service-Registry extrahiert. Der Knotentyp entspricht den funktionalen Anforderungen des Knotens, wie z.B. ein bestimmter Analysealgorithmus. Anschließend wird verglichen, welcher der Services auch die nichtfunktionalen Anforderungen des Knotens im DVM$^+$ erfüllen kann. Hierfür werden die Servicebeschreibungen (Zeile 17) mitsamt der Beschreibung ihrer nichtfunktionalen Eigenschaften extrahiert (Zeile 18), und es werden die einzelnen nichtfunktionalen Eigenschaften der Services mit den Anforderungen des jeweiligen Knotens im DVM abgeglichen (Zeile 24). Dies ist für die mittels Policys definierten Anforderungen einfach möglich, da sie bereits dem richtigen Format entsprechen. Beispielsweise können konkrete Kosten oder

```
// Saeuberung des Datenverarbeitungsmodells
DataProcessingModel dpm = clean(inputModel);

// Extrahiere die Knoten aus der Liste
List<Node> nodeList = extractNodes(dpm);
List<Service> foundServices = new List<Services>();

for (Node n: nodeList) {
  // der Typ des Knotens, bzw. dessen funktionale Eigenschaft
  String type = n.getType();
  List<Requirement> requirements = n.getRequirements();

  // Services mit passender Funktionalitaet
  List<Service> suitableServices =
    ServiceRegistry.getServices(type);

  for (Service s: suitableServices) {
    ServiceDescription sd = s.getDescription();
    Policy p = sd.getServicePolicy();

    // Vergleiche Policys der Services mit den Anforderungen
    Boolean fulfilled;
    for (Entry e: p.getEntries()) {
      for (Requirement r: requirements) {
        if (e.satisfies(r)) {
          fulfilled = true;
        } else {
          // Breche bei Nichterfuellung ab
          fulfilled = false;
          break;
        }
      }
    }
    // Nur alles erfuellende Services werden aufgenommen
    if (fulfilled) {
      foundServices.append(s);
    }
  }

  Service mostSuitable = findMostSuitableService(foundServices);
}
```

Listing 5.2: Auffinden passender Services

Verschlüsselungsmechanismen sowohl in der Anforderung des DVM konkret definiert sein, als auch in der Eigenschaftenbeschreibung des Services. Entsprechen Policy und Eigenschaften des Services nicht demselben Format, können Transformationen eingesetzt werden, die neben der Syntax der Policy auch konkrete Werte transformieren können. Beispielsweise können Kosten in den Services in Dollar definiert sein, wohingegen diese in der Policy in Euro angegeben werden. Durch den Einsatz von modularen Transformatoren kann diesem Problem begegnet werden.

Wurden die Anforderungen jedoch abstrakt definiert, muss die Anforderungsdefinition zuerst interpretiert werden. Wie in Abschnitt 5.1.2 beschrieben, wurden Punkte für die jeweiligen Anforderungen im DVM$^+$ vergeben, wobei die Summe aller vergebenen Punkte einen bestimmten Grenzwert nicht überschreiten darf. Wurden Punkte lokal für Einzelknoten vergeben, gelten die Anforderungen nur für die Knoten selbst. Wurden Anforderungen global definiert, dienen sie für alle Knoten, die keine lokale Anforderungsdefinition besitzen.

Die vergebenen Gewichtungspunkte können als erstes für eine Priorisierung der Anforderungen genommen werden. Hierdurch kann bestimmt werden, welche der Anforderungen am wichtigsten ist, z.B. Kosten, Sicherheit oder Effizienz. Anschließend wird versucht, einen passenden Service zu finden, der alle angegebenen Anforderungen erfüllt. Die Kosten sind oftmals bereits konkret definiert und können für den Abgleich der Anforderung mit den Eigenschaften des Services verwendet werden. Bzgl. Sicherheit oder Effizienz reicht es, wenn ein Service Mechanismen bietet, die diese Anforderungen umsetzen. Wird kein Service gefunden, der alle Anforderungen erfüllt, wird die am wenigsten priorisierte Anforderung gestrichen und die Suche wird erneut ausgeführt. Diese Vorgehensweise wird wiederholt bis ein passender Service gefunden wurde.

Erfüllt ein Service die Anforderungen eines Knotens im DVM$^+$ (Zeile 25), wird dieser zur Liste passender Services aufgenommen (Zeile 35). Falls nicht, wird der Service nicht beachtet (Zeile 28/29). Hierbei ist es nicht unüblich, dass mehrere Services in Frage kommen, mit denen die Daten unter Erfüllung der Anforderungen verarbeitet werden können. Ist dies der Fall,

wird überprüft, welcher der Services die Anforderungen am besten erfüllt. Erfüllen alle gefundenen Services die Anforderungen in gleichem Maße, werden die konkreten Werte der Anforderungen verglichen, beispielsweise der Preis des angebotenen Services. Erst wenn alle Werte ebenfalls gleich sind, wird ein Service zufallsbedingt ausgewählt. Dass kein passender Service gefunden werden kann, wird hierbei ausgeschlossen.

Im Rahmen dieses Beitrags wird davon ausgegangen, dass Services mittels einer Beschreibung, beispielsweise in WSDL, versehen sind. Die Beschreibung steht bei Web Services in der Regel zur Verfügung oder kann automatisch generiert werden. Bei REST-Services liegt eine derartige Beschreibung jedoch nicht vor. Mandel [Man08] beschreibt, wie auch REST-Services mittels der Servicebeschreibungssprache WSDL beschrieben werden können. Dies ist notwendig, um deren funktionale und nichtfunktionale Eigenschaften zu definieren. Ohne eine derartige Beschreibung ist eine Auswahl von Services nicht möglich.

Wurden Services für alle Knoten des DVM$^+$ aufgefunden, kann das DVM$^+$ in ein ausführbares Modell –das DPM – überführt werden, wie im nachfolgenden Abschnitt vorgestellt.

5.3 Modelltransformation – DVM$^+$ zu DPM

Nachdem die zur Erfüllung der Anforderungen bestmögliche Ausführungsumgebung, bzw. die zu verwendenden Services, ausgewählt wurde, wird daraufhin das nichtausführbare DVM$^+$ in das ausführbare Modell DPM (Datenprozessierungsmodell) überführt. Das DPM beschreibt, wie die Services orchestriert werden müssen, um die im DVM$^+$ beschriebenen funktionalen und nichtfunktionalen Anforderungen umzusetzen. Dabei muss das DVM$^+$, das eigentlich einen Datenfluss beschreibt, in ein ausführbares Kontrollflussmodell umgewandelt werden, das die Services orchestriert.

Eine Ausführung von Datenflüssen mittels Kontrollflussgraphen wird von Kopp beschrieben [Kop16]. Dabei werden die Kontrollflussgraphen mit Datenflussannotationen versehen. Bei der Ausführung des Kontrollflussgraphen

werden die Daten auf Basis dieser Annotation verarbeitet. Die Anwendbarkeit wird dabei für die Workflowsprache BPEL gezeigt. Im Rahmen der vorliegenden Dissertation wird eine andere Vorgehensweise beschrieben. Statt Kontrollflussgraphen zu annotieren, werden sie basierend auf dem Datenflussmodell erzeugt. Da das DVM$^+$ ein einfaches Pipes-and-Filters-basiertes Modell darstellt, kann die Komplexität der Kontrollflussgrapherstellung gering gehalten werden.

Durch diese Vorgehensweise wird es darüber hinaus ermöglicht, das DVM$^+$ in verschiedenartige Kontrollflussgraphen zu überführen, um es ausführbar zu machen. Die ausführbaren Kontrollflussgraphen können durch verschiedene Formate umgesetzt werden. Typischerweise werden hierfür Workflowbeschreibungssprachen wie Web Services Business Process Execution Language (WS-BPEL) [ACD+03; WCL+05] oder Business Process Model and Notation (BPMN) [Gro06] verwendet. Die auszuwählende Sprache hängt vom Anwendungsfall ab, da deren Ausführungsumgebungen sich stark, beispielsweise bezüglich Robustheit oder Effizienz, unterscheiden. Im Rahmen dieser Dissertation wurden drei Transformationen beispielhaft umgesetzt, eine Transformation auf WS-BPEL, BPMN und auf eine leichtgewichtige, JSON-basierte Workflowsprache, die für die Ausführungsumgebung Node-RED [IBMb] verwendet wird. Da die Transformationen sehr ähnlich funktionieren, wird im Rahmen dieser Dissertation lediglich die Transformation auf WS-BPEL beschrieben, die auch eine der am häufigsten eingesetzten Workflowsprachen darstellt [LS07]. Eine Transformation auf Node-RED wird in [HRWM15] beschrieben, eine Transformation auf BPMN in der von mir betreuten Masterarbeit von Mahrous [Mah17].

Wie oben beschrieben, definiert das DPM die Orchestrierung von Services zur Umsetzung der beschriebenen Funktionalität. Das DPM beschreibt also Serviceaufrufe mittels einem Kontrollflussgraphen.

5.3.1 Automatische DPM-Erstellung

Das DPM wird mittels Workflows umgesetzt. Dabei hängt die Auswahl der Workflowsprache vom Anwendungsfall bzw. von der in Abschnitt 5.2 aus-

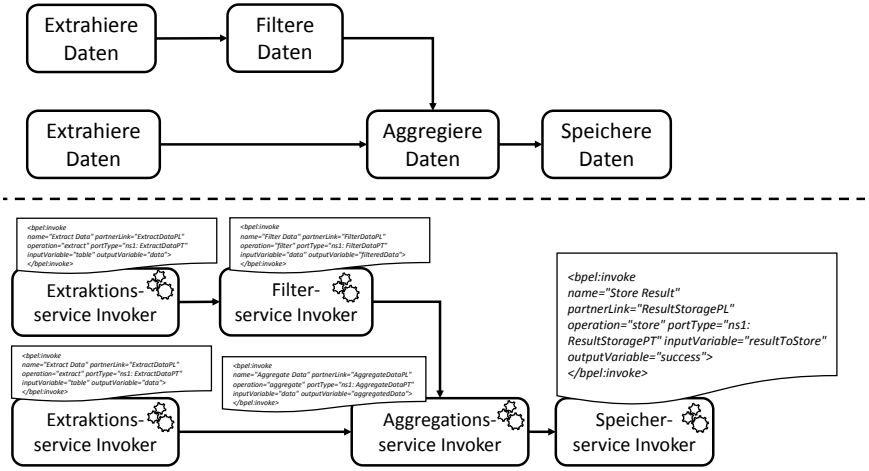

Abbildung 5.4: Beispielhaftes nichtausführbares $^{+}$ (oben), Ausführbarer Workflow des DVM^{+} (unten); Fragmente der Workflowaufrufe sind beispielhaft in BPEL beschrieben

gewählten Ausführungsumgebung ab. Bei der Erstellung des Workflows werden Aufrufe für die in Abschnitt 5.2 aufgefundenen Services automatisch erstellt. Die Reihenfolge dieser Aufrufe wird aus dem DVM^{+} extrahiert, das außerdem parallel auszuführende Datenoperationen definiert.

Als Grundlage für die automatische Workflowerstellung dient ein Fragment-Repository, wie zum Beispiel das von Schumm et al. entwickelte Repository Fragmento [SKLS11]. Ein Fragment stellt die Beschreibung des Aufrufs eines Services dar. Jedes Fragment wird genau einem Service zugeordnet. Durch die Kombination von Workflowfragmenten kann ein vollständiger Workflow erstellt werden. Es wird angenommen, dass für jeden möglichen aufzurufenden Service ein Workflowfragment im Repository hinterlegt ist und zur Erstellung des Workflows gefunden und extrahiert werden kann. Dies vereinfacht die DPM- bzw. die Workflowerstellung, da dadurch nur noch ein einfaches, baukastenartiges Zusammenstecken von Fragmenten notwendig ist.

Abbildung 5.4 zeigt die Erstellung eines Workflows basierend auf dem

DVM$^+$. Im oberen Bereich der Abbildung ist ein beispielhaftes Modell zu sehen, bei dem Daten aus zwei verschiedenen Quellen extrahiert, gefiltert, aggregiert und abgespeichert werden. Im unteren Bereich ist der entsprechende, ausführbare Workflow zu sehen. Jeder Knoten im Workflow entspricht einem Workflowfragment, das in der Abbildung beispielhaft in der Workflowsprache BPEL an den jeweiligen Knoten dargestellt ist. Jedes Fragment entspricht einem *BPEL-Invoke*-Element. Das Invoke-Element repräsentiert einen generischen Serviceaufruf, wobei die auszuführende Operation sowie deren Parameter im Fragment definiert sind. Die aufzurufenden Services wurden im vorigen Schritt ausgewählt.

Für bereits aufgesetzte bzw. provisionierte Services kann die Workflowerstellung wie beschrieben durchgeführt werden. Sind die Services jedoch noch nicht aufgesetzt, da sie lediglich dynamisch bei Bedarf bereitgestellt werden, sind deren Aufrufpfade noch nicht verfügbar. Erst nach dem Aufsetzen (siehe Kapitel 6) sind diese Pfade verfügbar. Um mit diesem Problem umzugehen, gibt es verschiedene Möglichkeiten. Zum einen können Platzhalter eingefügt werden, die nach der Provisionierung der Services ersetzt werden. Hierfür muss der erstellte Workflow erneut durchlaufen, die Pfade an der Service-Registry angefragt, und in den Workflow eingefügt werden. Eine andere Möglichkeit ist es, bereits Zugriffspfade der Services anzugeben, ohne dass sie bereits erreichbar sind. Die Serviceprovisionierung müsste dann die Services so aufsetzen, dass sie unter den im Workflow angegebenen Zugriffspfaden erreichbar sind. Hierfür muss jedoch bekannt sein, auf welcher Ausführungsumgebung sie aufgesetzt werden sollen.

Eine weitere Möglichkeit ist eine Abstraktion der Serviceaufrufe. Durch eine zentrale Middleware-Komponente, die die Serviceaufrufe intern an die Services weiterleitet, kann es ermöglicht werden mit dem beschriebenen Problem umzugehen. In den Workflow werden somit lediglich Aufrufe an die Middleware hinzugefügt, die später bei der Workflowausführung die Services aufrufen. Als Parameter werden zum Beispiel der Typ des aufzurufenden Services sowie dessen Parameter übergeben. Zum Zeitpunkt der Ausführung wurden die Services provisioniert. Auf diese Art müssen die Zugriffspfade zum Zeitpunkt der Workflowerstellung nicht bekannt sein. Daher wird im

Rahmen dieser Dissertation die Verwendung einer Middleware zur Serviceabstraktion gewählt. Wettinger et al. beschreiben eine derartige Middleware in [WBB+14]. Eine Beschreibung einer möglichen Implementierung folgt in Abschnitt 5.5.

Im Folgenden wird im Detail und an einem Beispiel beschrieben, wie eine automatische Workflowerstellung konkret mittels WS-BPEL umgesetzt werden kann.

5.3.2 Automatische Workflowerstellung – Beispiel in BPEL

Abbildung 5.5 zeigt einen automatisch erstellten Workflow in der Sprache BPEL, der basierend auf dem DVM^+ des Beispiels aus Abbildung 5.4 generiert wurde. Die grafische Darstellung dieses Workflows wurde mittels dem BPEL-Designer der Entwicklungsumgebung Eclipse[1] geschaffen. Dabei ist zu sehen, dass die automatische Erstellung des Workflows einfach gehalten werden kann, da dieser lediglich aus den einfachen BPEL-Konstrukten *Receive*, *Assign* und *Invoke* besteht und dessen Komplexität somit gering ist. Zur Unterstützung einer (Pseudo-)Parallelisierung werden außerdem die BPEL-Elemente *Flow* (im Beispiel als weißes Rechteck dargestellt) und *Sequence* verwendet. Weitere BPEL-Elemente treten nicht auf und sind für die Erstellung des Workflows nicht notwendig.

Die automatische Generierung des Workflows nutzt ein Repository, das Workflowfragmente der jeweiligen Services zur Verfügung stellt, die nur noch zusammengefügt werden müssen. Die Workflowerstellung für die Sprache BPEL traversiert alle Knoten im DVM^+ und extrahiert zu den gefundenen Services die passenden Workflowfragmente aus dem Repository. Ein Workflowfragment ist immer ein BPEL-Invoke-Element, das bei der Ausführung des Workflows einen Service aufruft. Die Fragmente müssen für die entsprechenden Services einmalig erstellt werden und können anschließend modular wiederverwendet werden. Durch Vorlagen kann die Fragmenterstellung stark vereinfacht bzw. sogar automatisch realisiert werden. Eine spezielle Parametrisierung oder Anpassung der Fragmente ist bei der Erstel-

[1] http://eclipse.org/

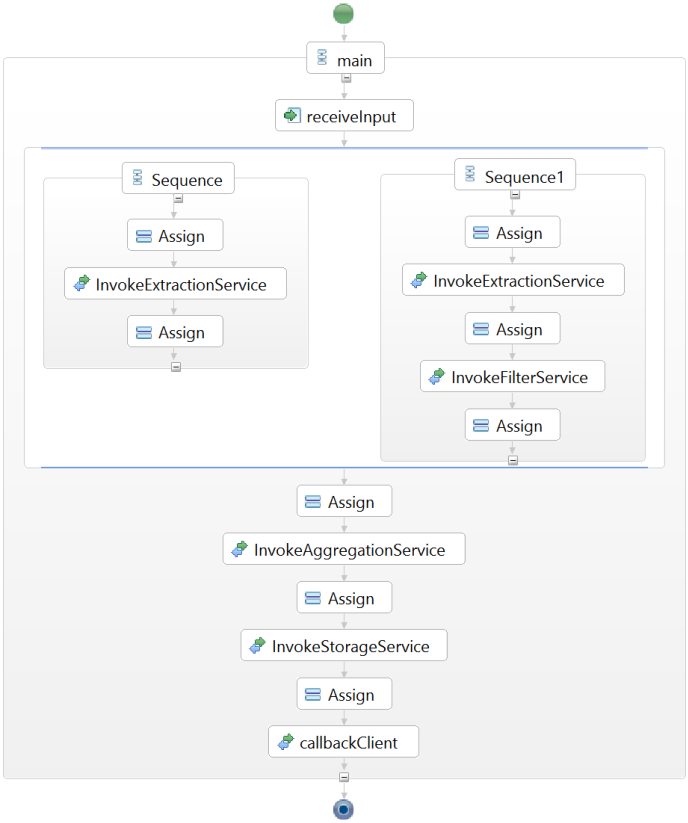

Abbildung 5.5: Transformiertes DVM$^+$ in der Workflowsprache BPEL

lung des Workflows nicht notwendig. Die Fragmente müssen lediglich an der passenden Stelle im Workflow eingefügt werden. Im Fall von BPEL handelt es sich bei den Fragmenten um textuelle XML-Beschreibungen, die einfach maschinell verarbeitet und zusammengefügt werden können.

Die Grundlage für die Erstellung des Workflows ist eine Grundstruktur eines BPEL-Prozesses. Diese enthält das BPEL-*Process*-Element, in das der gesamte Workflow eingebettet ist und eine Definition der Eingabeparameter des Workflows. Die Parameter (z.B. Zugangsdaten einer Datenbank)

```
<bpel:process name="Datenverarbeitungsprozess"
  targetNamespace="http://www.uni-stuttgart.de/bpel"
  xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/
      executable"

  <!-- Importe -->
  <bpel:import>...</bpel:import>

  <!-- Definition der PartnerLinks -->
  <bpel:partnerLinks>...</bpel:partnerLinks>

  <!-- Variablendeklarationen -->
  <bpel:variables>...</bpel:variables>

  <!-- Workflowablauf -->
  <bpel:sequence name="main">

    <!-- Eingabedaten des Prozesses -->
    <bpel:receive name="receiveInput" partnerLink="client"
      portType="tns:SimpleInvokeProcess"
      operation="process" variable="input"
      createInstance="yes"/>

    <!-- Hier werden die Serviceaufrufe ergaenzt -->
  </bpel:sequence>
</bpel:process>
```

Listing 5.3: Grundstruktur des BPEL-Prozesses

können den Knoten des DVM$^+$ entnommen und automatisch den Parameterdefinitionen des Workflows hinzugefügt werden. Sie dienen später als Eingabeparameter für die Serviceaufrufe.

Die Grundstruktur des BPEL-Prozesses ist in Listing 5.3 dargestellt. Dabei sind alle grundlegenden Elemente vorhanden, um einen solchen BPEL-Workflow ausführbar zu machen. Lediglich die Serviceaufrufe bzw. die zugehörigen Workflowfragmente müssen noch ergänzt werden.

Für die Workflowerstellung, basierend auf der Grundstruktur, kann das DVM$^+$ als Baum betrachtet werden. Die Datenquellen bzw. Datenextraktionsoperationen stellen die Blattknoten dar, da sie nur ausgehende Kanten besitzen. Die Datensenken stellen hingegen die Wurzelknoten des Baumes dar. Bei der Workflowerstellung wird dieser Baum mittels Tiefensuche traversiert.

Für jeden gefundenen Knoten werden im Workflow jeweils ein BPEL-Assign- und ein BPEL-Invoke-Element eingefügt. Das Assign-Element ist dafür verantwortlich, dem Invoke-Element die Eingabeparameter für den Serviceaufruf sowie ggf. die zu verarbeitenden Daten zuzuweisen, die aus einem vorigen ausgeführten Serviceaufruf stammen können. Diese Vorgehensweise wird für jeden Knoten im Baum wiederholt. Teilt sich der Baum in mehrere Zweige auf, werden die BPEL-Assign- und Invoke-Elemente pro Zweig innerhalb von separaten BPEL-Sequence Elementen erzeugt. Alle parallelen Zweige bzw. BPEL Sequence-Elemente werden anschließend in BPEL-Flow-Elemente verschoben. Dadurch kann der BPEL-Workflow (pseudo)-parallel ausgeführt werden[1]. Sind die Wurzelknoten des Baumes erreicht, ist die Workflowerstellung abgeschlossen.

Der Ablauf ist in Listing 5.4 als Javabasierter Pseudocode dargestellt:

Dabei ist zu sehen, dass zuerst das DVM$^+$ mittels Tiefensuche traversiert wird und dessen Knoten in sortierter Reihenfolge abgespeichert werden (Zeile 1). Anschließend werden alle vom Modellierer angegebenen Parameter aus dem DVM$^+$ extrahiert und an den BPEL-Prozess mittels einem BPEL-Receive-Knoten übergeben (Zeile 2). Dieser Knoten kann als Eingabeparameter des Prozesses gesehen werden. Hierfür wird als erstes eine neue Prozessinstanz erzeugt (Zeile 4), der Receiveknoten selbst mit den Parametern instanziiert (Zeile 5), und anschließend an den Prozess angehängt. Nachdem dadurch das Grundgerüst des Prozesses geschaffen wurde, können die Serviceaufrufe eingefügt werden. Hierfür wird die sortierte Liste an Knoten durchlaufen (Zeile 8), und es werden für jeden Knoten ein BPEL-Assign und ein BPEL-Invoke eingefügt (Zeile 10-17). Das Assign-Element bekommt jeweils das Ergebnis des Vorgängerknotens sowie dessen Parameter, die anfangs dem Prozess übergeben wurden (Zeile 11). Ein Invoke-Knoten wird anschließend aus dem Fragment-Repository extrahiert und ebenfalls instanziiert.

Hat ein Knoten keine Vorgänger (Zeile 13), werden die Parameter direkt aus dem Receive-Knoten extrahiert – ein Ergebnis des Vorgängers gibt es in diesem Fall nicht. Anschließend werden die Knoten an den Workflow

[1] abhängig von der Ausführungsumgebung findet die Ausführung echt-parallel in mehreren Threads, oder pseudo-parallel in einem Thread statt

```
List<Node> sortedList = getNodesByDepthSearch(tree);
String parameters = getParametersFromDataProcessingModel();

BPELProcess process = new BPELProcess();
Receive receive = new Receive(parameters);
process.appendChild(receive);

for (Node node: sortedList) {
  Node predecessor = node.getPredecessor();
  if (node.getPredecessor() != null){
    Assign assign = new Assign(predecessor.result, predecessor.
        parameters);
  } else {
    Assign assign = new Assign(receive.parameters);
  }
  Invoke invoke = new Invoke(FragmentRepo.getFragment(node));
  process.append(assign);
  process.append(invoke);
}

Package executable = createDeployablePackage(process);
WorkflowEngine engine = deployPackage(executable);
```

Listing 5.4: Workflowerstellung in BPEL und Provisionierung

angehängt (Zeile 16, 17). Im letzten Schritt wird ein Paket mit dem Workflow, den WSDL-Dateien der Services sowie den Workflowartefakten erstellt (Zeile 20), das für eine Ausführung in einer BPEL-Workflowumgebung verwendet werden kann (Zeile 21).

Es existieren viele Services zur Datenverarbeitung, die nicht, wie hier beschrieben, auf SOAP basieren und somit auch keine WSDL-Beschreibung besitzen. Dabei handelt es sich vor allem um REST-basierte Services. Derartige Services sollen durch den hier vorgestellten BPEL-basieren Ansatz ebenfalls unterstützt werden. Um mit REST-basierten Services umzugehen, stellt Pautasso [Pau08] BPEL4REST vor. Mittels einer Erweiterung von BPEL ist es dadurch möglich auch REST-basierte Services auszurufen.

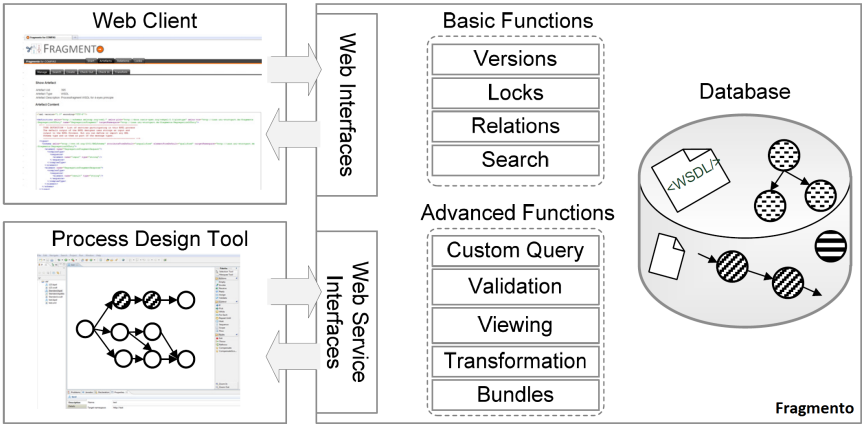

Abbildung 5.6: Architektur des Fragmento-Repositorys (rechte Seite, aus [SKLS11])

5.4 Verwandte Arbeiten

In diesem Abschnitt werden verwandte Arbeiten, insbesondere bezüglich einer automatischen Generierung von Workflows, diskutiert und von den vorgestellten Konzepten abgegrenzt.

Schumm et al. [SKLS11] stellen Fragmento vor, ein Repository für das Archivieren von Workflowfragmenten. Dessen Architektur ist in Abbildung 5.6 dargestellt. Neben den Grundfunktionen, dem Abspeichern und Abrufen bzw. der Suche nach Fragmenten, bietet Fragmento außerdem vielfältige Funktionalitäten für deren Verwaltung an. Darunter eine Versionierung der Fragmente, deren Validierung, Browsingfunktionalitäten, oder Transformationen. Es existiert einerseits eine Webklientenanwendung, über die die Fragmente eingesehen werden können und andererseits ein Design Tool, das die Erstellung von Workflows basierend auf den Fragmenten erlaubt.

Für die Bedürfnisse dieses Beitrags werden lediglich die Persistierung und die Suche nach Fragmenten benötigt. Fragmento bietet darüber hinaus, wie in Abbildung 5.6 dargestellt, viele weitere nicht benötigte Funktionen. Aus diesem Grund wurde in einer von mir betreuten Diplomarbeit [Kal16] eine

leichtgewichtige Variante des Fragment-Repositorys geschaffen.

Képes [Kep13] und Breitenbücher et al. [BBK+14a] stellen ein Konzept zur automatischen Generierung von BPEL-Workflows für die Orchestrierung von Services basierend auf TOSCA vor. Hierfür werden ebenfalls Workflows aus Fragmenten generiert. Es stehen primär Serviceaufrufe zum Aufsetzen von Anwendungen im Vordergrund. Für die Überführung des DVM^+ in eine ausführbare Repräsentation werden ebenfalls Workflows aus Fragmenten zusammengesetzt. Die Arbeiten von Képes [Kep13] und Breitenbücher et al. [BBK+14a] dienen dabei als Inspiration.

Eine Ausführung von Datenflüssen mittels Kontrollflussgraphen wird von Kopp beschrieben [Kop16]. Dabei werden die Kontrollflussgraphen mit Datenflussannotationen versehen. Bei der Ausführung des Kontrollflussgraphen werden die Daten auf Basis dieser Annotation verarbeitet. Die Anwendbarkeit wird dabei für die Workflowsprache BPEL gezeigt. Im Rahmen der vorliegenden Dissertation wird eine andere Vorgehensweise beschrieben. Statt Kontrollflussgraphen zu annotieren, werden sie basierend auf dem Datenflussmodell erzeugt. Da das DVM^+ ein einfaches Pipes-and-Filters-basiertes Modell darstellt, kann die Komplexität der Kontrollflussgrapherstellung gering gehalten werden.

Bezüglich der anforderungsbasierten Auswahl der Ausführungsumgebung sei an dieser Stelle an die Untersuchung etablierter Werkzeuge zur Datenverarbeitung verwiesen, die in Abschnitt 4.5 beschrieben wurde. Hierbei wurde unter anderem untersucht, ob bisherige Ansätze es ermöglichen, Anforderungen an die Datenverarbeitung zu definieren. Dabei hat sich ergeben, dass keines der verfügbaren Werkzeuge einen derartigen Ansatz verfolgt. Die Annotation des DVM mit Anforderungen, das daraus entstehende DVM^+ sowie die Transformation auf das DPM stellen ein Alleinstellungsmerkmal dar.

Zusätzlich sei auf die in Abschnitt 9.1 beschriebene Teilnahme an den Rapid Mashup Challenges 2015 und 2016 verwiesen. Dabei wurde unter anderem auch die prototypische Implementierung der anforderungsbasierten Ausführung vorgestellt. Der Prototyp konnte durch die Fachjury ebenfalls positiv (mit einem zweiten Platz) evaluiert werden (siehe [HB17; HM16a]).

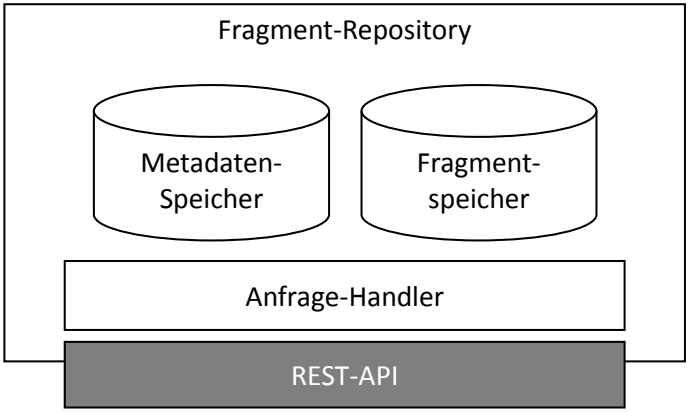

Abbildung 5.7: Architektur des Fragment-Repository (aus [Kal16])

5.5 Implementierung und Evaluation

Die vorgestellten Konzepte wurden zum Zweck der Evaluation prototypisch implementiert. Die Funktionalität wurde teilweise selbst, teilweise mit Hilfe studentischer Arbeiten [Iva17; Kal16] realisiert. Die Implementierung umfasst drei Teile: (1) die Erweiterung des Modellierungswerkzeugs um den Anforderungskatalog sowie die Möglichkeit der Anforderungsdefinition, (2) ein Fragment-Repository, das alle Workflowfragmente enthält sowie eine Suchfunktionalität, um Fragmente aufzufinden, (3) die Service-Registry, in der Services aufgefunden werden können sowie Service-Provider neue Services registrieren können und (4) die Modelltransformation basierend auf BPEL, BPMN und Node-RED.

Die Anpassung des Modellierungswerkzeuges wurde mittels den für dessen Erstellung eingesetzten Technologien und Programmiersprachen umgesetzt. Die Technologien umfassen das JavaScript-Framework Alloy, die Programmiersprache JavaScript und die Markupsprachen HTML und CSS zur visuellen Darstellung. Die Konzepte wurden anhand den in Abbildung 5.1 und Abbildung 5.3 abgebildeten Darstellungen implementiert. Für die Definition der Policys kam das in Listing 5.1 beschriebene Format zum Einsatz.

Das Fragment-Repository bietet einerseits eine Schnittstelle, um neue Fragmente hinzuzufügen, und andererseits, um Fragmente basierend auf Metadaten zu suchen. Das Repository wurde in der von mir betreuten Diplomarbeit "Regelbasiertes Pattern-Mapping von Mashup Plans" [Kal16] basierend auf den Konzepten dieser Dissertation implementiert. Es wurden zwei Komponenten entwickelt: 1) das Fragment-Repository selbst, bestehend aus einer NoSQL-Datenbank zum Speichern der Fragmente und 2) eine relationale Datenbank, um dazugehörige Metadaten zu speichern. Über eine REST-Schnittstelle kann auf diese Daten zentral zugegriffen werden. Die Architektur des Repositorys ist in Abbildung 5.7 zu sehen.

Die REST-Schnittstelle bietet die klassischen Create-Retrieve-Update-Delete (CRUD)-Operationen an:

- **Erstellung eines neuen Fragments** Hierfür müssen das Fragment als Binärdatei sowie die Metadaten des Fragments angegeben werden. Die Metadaten umfassen einen eindeutigen Namen des Fragments, eine Beschreibung dessen funktionalen und nichtfunktionalen Eigenschaften, der assoziierte Service und die Eingabeparameter der Services. Bei der Erstellung eines Fragments werden die Metadaten separat in einer relationalen Datenbank abgespeichert, um leicht auffindbar zu sein. Diese Metadaten enthalten eine Referenz auf das eigentliche Fragment im Fragmentspeicher. Da relationale Datenbanken für das Speichern großer monolithischer Objekte nicht optimal sind, werden Fragmente daher in einer separaten NoSQL-Datenbank persistiert.
- **Auffinden von Fragmenten basierend auf Metadaten** Bei der Erstellung des ausführbaren Workflows müssen die Fragmente aus dem Repository extrahiert werden. Hierfür bietet die REST-Schnittstelle des Repositorys eine *Retrieve*-Methode an. Die Suche nach dem aufzufindenden Fragment kann basierend auf dessen Namen oder basierend auf allen anderen Metadaten geschehen. Beispielsweise kann so eine Suche basierend auf der Parametrisierung der Services erfolgen. Ist ein passender Eintrag gefunden, kann das Fragment über die Referenz aus den Metadaten aufgefunden und zurückgegeben werden.

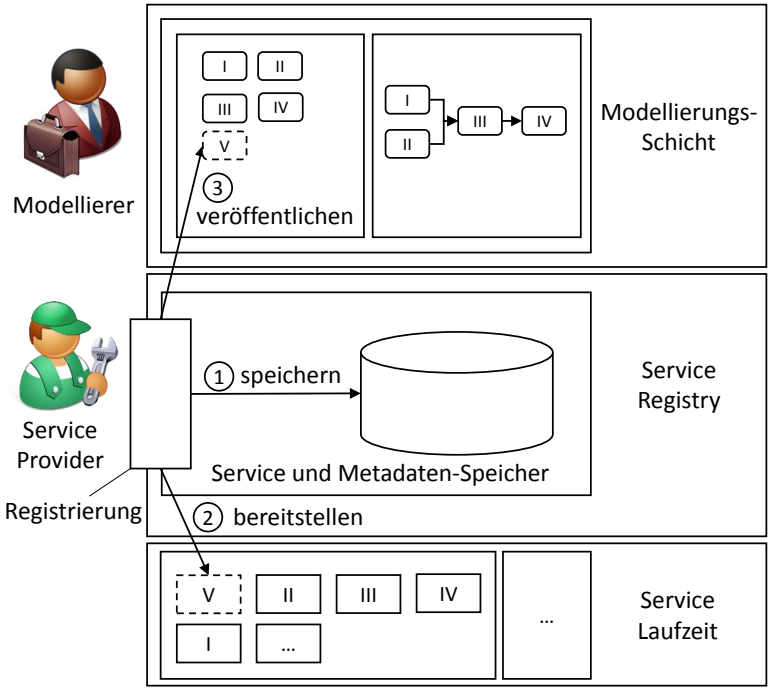

Abbildung 5.8: Architektur des Service-Repositorys (basierend auf [Iva17])

- **Ändern und Löschen der Fragmente** Des Weiteren wurden Funktionen zum Löschen und Ändern der Fragmente implementiert, die entsprechende Operationen auf Metadaten- und Fragmentspeicher ausführen.

Die Implementierung der beschriebenen Komponenten wurde mit einer MySQL-Datenbank[1] sowie einer MongoDB-NoSQL-Datenbank[2] umgesetzt. Die REST-Schnittstelle wurde mittels dem Framework Java-Spring[3] implementiert.

Des Weiteren wurde im Rahmen zweier studentischer Arbeiten ein Service-

[1] https://www.mysql.com/de/
[2] https://www.mongodb.com/
[3] https://spring.io/

Repository implementiert, um die von den Fragmenten aufgerufenen Services bereitzustellen und zugreifbar zu machen [Iva17; Mah17]. Das Service-Repository bietet die Möglichkeit neue Services zu registrieren (Abbildung 5.8, 1), die anschließend als Basis für eine erweiterte Modellierung des DVM dienen. Die Architektur des Service-Repository ist in Abbildung 5.8 dargestellt. Dabei ist zu sehen, dass eine neue Komponente, die Service-Registry, geschaffen wurde. Die Service-Registry implementiert einerseits die gängigen Servicekonzepte und -technologien wie UDDI, als auch eine Möglichkeit, Services automatisiert mittels dem TOSCA-Standard bereitzustellen (Abbildung 5.8, 2). Bei der Registrierung von Services, die anschließend vom System für die Verarbeitung von Daten verwendet werden können, gibt es die Möglichkeit, bereits gehostete Services, beispielsweise auf einem dedizierten Server, anzugeben oder, falls die Services noch nicht gehostet sind, diese automatisch provisionieren zu lassen.

Häufig kommt es vor, dass Services bereits auf einer Laufzeitumgebung gehostet sind oder von einem Cloud-Anbieter bereitgestellt werden. In diesem Fall benötigt die Service-Registry die URI, unter der der Service erreichbar ist sowie eine (z.B. WSDL-)Beschreibung des Services, um einen neuen Eintrag in der Service-Registry für den Service anzulegen. Um neben Java-Web-Services auch andere Typen von Services zu unterstützen, gibt es außerdem die Möglichkeit bereits gehostete REST-Services zu registrieren. Hierfür müssen die URI sowie eine Beschreibung des Services (z.B. mittels WSDL) angegeben werden (vgl. [Man08; Pau08]).

Nachdem bereits gehostete Services registriert wurden, können sie mit einem neuen oder bestehenden Modellierungsmuster der Oberfläche verknüpft werden. Bei der Erstellung eines neuen Modellierungsmusters muss ein neuer Eintrag, wie in Abschnitt 4.2 beschrieben, für den Service verfasst werden. Anschließend erscheint in der Modellierungsoberfläche das entsprechende Modellierungsmuster in der Palette, damit es für die Modellierung des DVM verwendet werden kann (Abbildung 5.8, 3). Beispielsweise könnte ein Service Provider einen neuen Analysealgorithmus implementieren, den er (bspw. kostenpflichtig) den Modellierern zur Verfügung stellen möchte. Hierfür stellt er die Implementierung als Service zur Verfügung, erstellt eine

Servicebeschreibung inklusive Eigenschaften und Nutzungsbedingungen des Services mittels Policys und erstellt das Modellierungsmuster für diesen Analysealgorithmus im Katalog. Anschließend kann das Modellierungsmuster von den Modellierern verwendet werden.

Ist ein Service noch nicht provisioniert, kann er von der Service-Registry automatisch in einer eigenen Laufzeitumgebung bereitgestellt werden. Die Registrierung verläuft wie bei den bereits gehosteten Services, mit dem Unterschied, dass zusätzlich die Implementierung des Services sowie der Pfad unter dem dieser erreichbar sein soll bereitgestellt werden muss. Unterstützt werden Services in den Programmiersprachen Java, Python 3.6 bzw. 2.7 und Node.js. Um solche Implementierungen mittels TOSCA provisionieren zu können, wird ein generisches TOSCA-CSAR pro unterstützter Programmiersprache bereitgestellt, das anschließend mit den servicespezifischen Details parametrisiert werden muss. Die Topologie des generischen CSARs für Services in der Programmiersprache Java ist in Abbildung 5.9 dargestellt. Es ist zu sehen, dass in diesem Prototypen Java-Web-Services ausschließlich auf einem Apache-Tomcat-Applikationsserver bereitgestellt werden, der wiederum auf einem Ubuntu-16.04-Betriebssystem läuft. Eine Anpassung, beispielsweise auf ein anderes Betriebssystem, ist im TOSCA-Standard durch den Austausch des Node-Templates möglich.

Das parametrisierte CSAR kann anschließend für die Provisionierung mittels einer TOSCA-Laufzeitumgebung verwendet werden. Analog zur Registrierung eines bereits gehosteten Services, muss der Service mit einem Modellierungsmuster verknüpft werden oder, falls noch kein entsprechendes Muster existiert, muss ein neues erstellt werden.

Selbstverständlich bietet die Service-Registry die Möglichkeit, Services aufzufinden, wie es auch in UDDI üblich ist. Services können über ihre Metadaten, Parametrisierung, deren Namen, oder deren Beschreibung durchsucht werden. Ist ein Service gefunden, wird entweder dessen WSDL-Beschreibung zurückgegeben – vorausgesetzt es handelt sich um einen Java-Web-Service – oder die URI und Parametrisierung eines REST-Services in beliebiger Programmiersprache mittels einer Beschreibung im JSON Format.

Die Fragment- und Service-Repositorys dienen als Grundlage für die Imple-

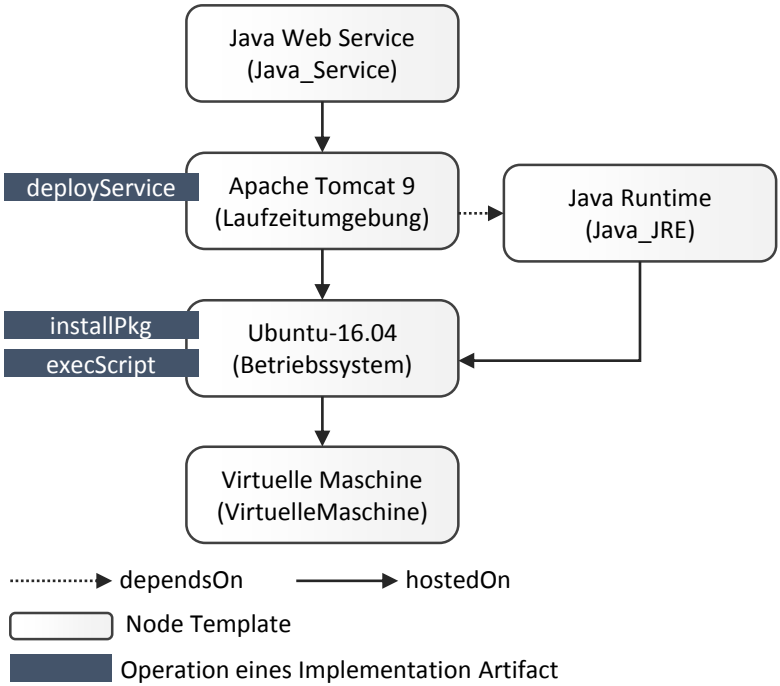

Abbildung 5.9: Beispieltopologie für das Aufsetzen eines Java Web Services mittels TOSCA. Die dazugehörigen Node-Types sind in Klammern angegeben

mentierung der Modelltransformation, die basierend auf den vorgestellten Konzepten umgesetzt wurde. Die Überführung des DVM in Workflows wurde in Abschnitt 5.3.2 beispielhaft für die Workflowsprache BPEL beschrieben.

In der von mir betreuten und von Ziegler durchgeführten [Zie17] Masterarbeit "Ein Sicherheitskonzept für verteilte Datenverarbeitungssysteme" wurde eine Implementierung für die Realisierung der nichtfunktionalen Anforderung "Sicherheit" geschaffen, die im Rahmen dieses Beitrags beschrieben wurde. Hierfür wurden verschiedene Sicherheitsmodule implementiert, die in der Lage sind, Daten bei der Übertragung und Zwischenspeicherung zu

verschlüsseln. Des Weiteren wurde eine Verwaltungskomponente geschaffen, die sich um die Verteilung der Schlüssel an die berechtigten Services kümmert. Die Sicherheitsmodule stehen als Services zur Verfügung und können bei der Zusammenstellung einer sicheren Ausführungsumgebung hinzugezogen werden. Eine Publikation der Ergebnisse dieser Masterarbeit steht noch aus.

5.6 Zusammenfassung

In diesem Kapitel wird die dynamische Transformation des DVM$^+$ in das ausführbare DPM beschrieben. Hierfür wurde in einem ersten Schritt ein Katalog geschaffen, der Anforderungen an die Datenverarbeitung definiert. In dem erweiterbaren Katalog ist es Domänennutzern möglich, vorhandene Anforderungen zu durchsuchen, sie zu parametrisieren und zu kombinieren. Da Domänennutzer dazu neigen, zu hohe Anforderungen an die Datenverarbeitung zu stellen (beispielsweise, dass sie sowohl sehr effizient als auch sicher sein soll), wurde ein Punktesystem geschaffen, mit dem die Anforderungen priorisiert werden können.

Bei der Definition von Anforderungen wurde zwischen zwei Arten unterschieden: (1) eine abstrahierte Anforderungsdefinition, und (2) eine policy-basierte Anforderungsdefinition. Die abstrahierte Variante zielt auf Domänennutzer ab, die nicht in der Lage sind, klare Anforderungen zu benennen und eine genaue Spezifizierung durchzuführen (bspw. eine bestimmte Datenverschlüsselungstechnik). Daher können in dieser Variante Anforderungen nur abstrakt definiert werden. Bei der policy-basierten Anforderungsdefinition hingegen können die Anforderungen an die Datenverarbeitung sehr genau spezifiziert werden. Es können komplexe Parameter gesetzt werden sowie Expertenwissen (bspw. über Verschlüsselung) mit eingebracht werden. Diese Variante ist besonders für Expertennutzer geeignet.

Nachdem die Anforderungen definiert sind, wird die Auführungsumgebung dynamisch, basierend auf den Anforderungen, ausgewählt. Hierfür wurde ein Algorithmus vorgestellt, mit dem passende Services gefunden

werden können. Anschließend wird das abstrakte DVM^+ in eine ausführbare Repräsentation – das DPM – überführt, die Aufrufe von Datenverarbeitungsservices enthält. Das DPM wird mittels Workflowtechnologien umgesetzt. Die in Beitrag **B2** (vgl. Kapitel 3) vorgestellten Konzepte wurden durch eine prototypische Implementierung validiert. In Kapitel 9 wird der Gesamtansatz anhand ausführlicher Beispiele evaluiert.

Kapitel 6

Automatische Provisionierung der DPM-Ausführungsumgebung

Mit der anforderungsbasierten Auswahl der Ausführungsumgebung ist auch bekannt, welche Services bzw. sonstige Softwarekomponenten zur DPM-Ausführung benutzt werden sollen. Manche der Komponenten können bereits aufgesetzt sein, beispielsweise wenn sie durch Cloud-Provider zur Verfügung gestellt werden. Andere Komponenten müssen noch für die Ausführung zur Verfügung gestellt werden. Die Softwarekomponenten umfassen bspw. Services zur Datenverarbeitung, Frameworks (wie Apache Spark), Betriebssysteme, Webserver, Datenbanken für Zwischenspeicher oder Datensenken.

Die Provisionierung der Ausführungsumgebung basiert in dieser Dissertation vollständig auf dem TOSCA-Standard. Die in diesem Abschnitt vorgestellten Konzepte gründen auf den Veröffentlichungen [HBBL14; HM16b]. Dabei stammen die in [HM16b] entstandenen Konzepte vollständig von mir, wohingegen in [HBBL14] zusätzlich Inhalte der Koautoren, bezüglich der

Integration in das TOSCA-Ökosystem OpenTOSCA, eingeflossen sind.

Im Folgenden werden die für das Aufsetzen von Softwarekomponenten oft verwendeten Begriffe “Deployment” und “Provisionierung” als Synonyme verwendet. Es gilt folgende Definition:

Definition 6.1 (Provisionierung)
Unter der Provisionierung von Softwarekomponenten wird verstanden, dass sie auf lokaler oder externer Infrastruktur aufgesetzt werden. Das Aufsetzen umfasst die Installation, Konfiguration und das Starten der Softwarekomponenten und deren Abhängigkeiten, sodass diese lauffähig sind.

Ausgangspunkt für das automatische Aufsetzen, bzw. die Provisionierung der Ausführungsumgebung, ist eine Liste an Softwarekomponenten und deren Abhängigkeiten (Laufzeitumgebungen, Bibliotheken, Betriebssysteme, etc.). Die Liste stammt aus der in Abschnitt 5.2 beschriebenen Auswahl der Ausführungsumgebung. Dabei wurde definiert, welche Services für die Ausführung benötigt werden. Einige dieser Softwarekomponenten sind bereits provisioniert, beispielsweise durch einen Cloudanbieter, andere müssen erst aufgesetzt werden. Ein dynamisches Aufsetzen spart insbesondere bei selbst gehosteten Komponenten Kosten, da sie sonst auch ohne Nutzung Ressourcen in Anspruch nehmen. Werden die Komponenten nicht mehr benötigt, können sie wieder deprovisioniert werden. Ähnlich verhält es sich mit Komponenten, die von externen Anbietern als Dienste zur Verfügung gestellt werden. Wird die Buchung der Dienste lediglich für die Zeit der Datenverarbeitung durchgeführt, können die Kosten stark reduziert werden.

Die Liste der zur Ausführung benötigten Services wird durch das in Abschnitt 5.2 beschriebene Konzept lediglich textuell zur Verfügung gestellt und kann daher nicht direkt für eine Provisionierung verwendet werden. Daher wird diese Liste im ersten Schritt in eine TOSCA-Repräsentation überführt, die für eine anschließende automatisierte Provisionierung genutzt werden kann. Die notwendigen Schritte hierfür sind in der in Abbildung 6.1 dargestellten Methode zu sehen. Die Liste an Ausführungskomponenten sowie das transformierte, ausführbare DPM dienen als Eingabe und werden verwendet, um eine TOSCA-Topologie zu erstellen, die alle zur Ausführung

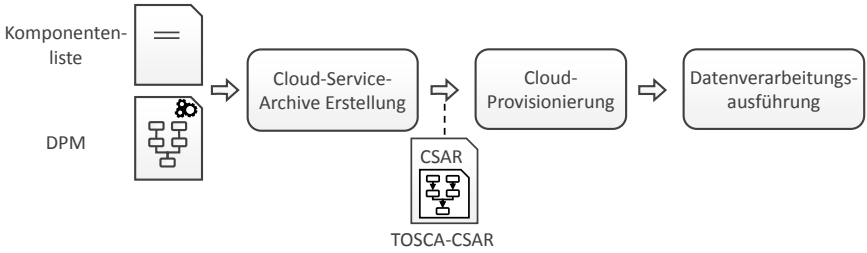

Abbildung 6.1: Methode zur Provisionierung der Ausführungsumgebung (basierend auf [HM16b])

benötigten Services enthält. Basierend auf der Topologie wird dann ein TOSCA-Cloud-Service-Archive (CSAR) erstellt, das zur automatisierten Provisionierung mittels einer TOSCA-Ausführungsumgebung verwendet werden kann. Die Schritte der Methode werden im Folgenden genauer beschrieben. Als erstes wird beschrieben, wie ein TOSCA-CSAR für das Aufsetzen der Ausführungsumgebung erstellt werden kann.

6.1 Erstellung der Topologie und des Cloud-Service-Archives

Die Erstellung eines TOSCA-Cloud-Service-Archives erfordert zwei Voraussetzungen. Zum einen müssen *Node-Types* und *Relationship-Types* für die zu provisionierenden Services und deren Abhängigkeiten (Laufzeitumgebungen, Bibliotheken, etc.) existieren. Zum anderen müssen Artefakte, also Skripte und Binärdateien, vorhanden sein, die für die eigentliche Provisionierung benötigt werden. Zur Definition nichtfunktionaler Anforderungen an die Provisionierung von Softwarekomponenten (z.B. Kosten, Sicherheit) wurden in [WWB+13] außerdem TOSCA-Policys vorgestellt. TOSCA-Policys müssen ebenfalls bereits in den TOSCA-Typen definiert sein, um verwendet werden zu können.

Sind diese Voraussetzungen erfüllt, kann das TOSCA-CSAR erstellt werden. Grundbestandteil eines jeden CSAR ist eine TOSCA-Topologie, die alle zu provisionierenden Komponenten und ihre Abhängigkeiten mittels *Node-*

Templates und *Relationship-Templates* beschreibt. Die Topologie beschreibt sowohl die Struktur der zu provisionierenden Softwareumgebung als auch die zur Provisionierung und Management verwendeten Artefakte (Skripte, Installationsdateien, etc.).

6.1.1 Topologieerstellung

Als Grundlage für die Topologieerstellung dient ein Repository, das alle verfügbaren TOSCA-Elemente (Types, Artefakte, etc.) enthält. Beispielsweise kann das von Kopp et al. [KBBL13] vorgestellte Repository des TOSCA-Modellierungswerkzeugs Winery verwendet werden. Um die Topologie erstellen zu können, werden als erstes Node-Types für jeden Eintrag der Komponentenliste in dem Repository gesucht. Die Suche kann beispielsweise basierend auf den Namen der Services stattfinden. Da es die Grundvoraussetzung ist, dass für jede Komponente ein passender Node-Type existiert, können hier keine Fehler auftreten. Jedoch können verschiedene Versionen der Softwarekomponenten existieren. Wir gehen davon aus, dass die Versionsbezeichnungen ebenfalls bereits in der Liste der Softwarekomponenten enthalten sind. Das Ergebnis ist eine Liste von Node-Types, die für die Ausführung der Datenverarbeitung benötigt werden. Hierbei ist zu beachten, dass zu diesem Zeitpunkt lediglich anwendungsspezifische Komponenten (Datenverarbeitungsservices, Datenbankservices, etc.) mittels Node-Types abgebildet sind. Diese Komponenten sind in den meisten Fällen noch nicht lauffähig, da zusätzliche Plattformkomponenten (z.B. Web-Server, Laufzeitumgebungen) und Infrastrukturkomponenten (z.B. virtuelle Maschinen) benötigt werden. Fehlende Komponenten müssen in der Topologie ergänzt werden, um eine lauffähige Ausführungsumgebung zu erreichen.

Im ersten Schritt der Topologieerstellung wird das Gerüst der Topologie erzeugt. Das bedeutet, dass eine "leere" TOSCA-Topologie (in TOSCA: "Topology-Template") angelegt wird. Anschließend werden Node-Templates für die im vorigen Schritt gefundenen Node-Types instanziiert. Um die Node-Template-Instanziierung vollautomatisiert zu ermöglichen, müssen die Eigenschaften (TOSCA-Properties) der Templates bekannt sein, damit

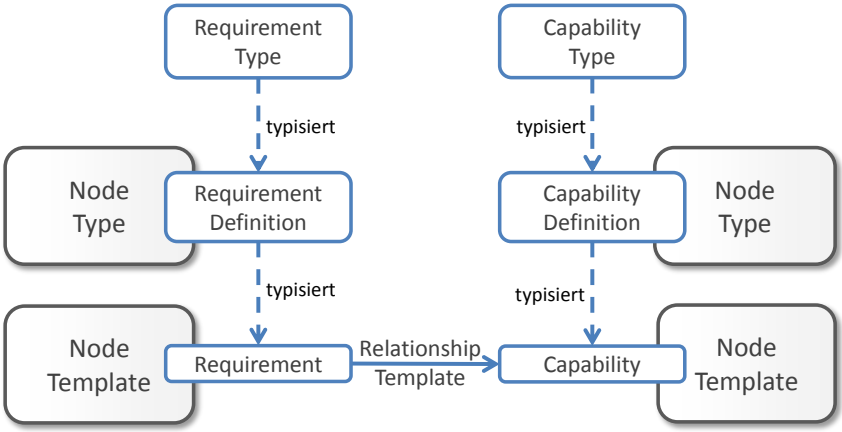

Abbildung 6.2: Verwendung von Requirements und Capabilities in TOSCA (basierend auf [HBBL14])

sie bei der Instanziierung befüllt werden können. Beispielsweise muss beim Aufsetzen einer virtuellen Maschine angegeben werden, wie viele Rechenressourcen sie zugewiesen bekommen soll. Falls möglich, werden an dieser Stelle Defaultwerte für die Eigenschaften eingefügt. Weitere für eine Instanziierung notwendige Angaben (z.B. ein eindeutiger Bezeichner des Node-Templates) können generiert werden. Nach diesem Schritt enthält die Topologie alle Node-Templates, die auf der Serviceliste aufgeführt sind. Jedoch kann die Topologie in den meisten Fällen noch nicht für eine Provisionierung der Ausführungsumgebung verwendet werden, da notwendige Abhängigkeiten (Laufzeitumgebungen, Bibliotheken, etc.) fehlen, die über eine Topologievervollständigung ergänzt werden.

6.1.2 Topologievervollständigung

Erste Arbeiten zur automatischen Vervollständigung von TOSCA-Topologien finden sich in [HBBL14]. In [HM16b] habe ich das von mir entwickelte Konzept nochmals für die vorliegende Aufgabenstellung weiterentwickelt.

Der Ansatz zur automatischen Vervollständigung basiert auf TOSCA-

Requirements und *-Capabilities*. Abbildung 6.2 beschreibt wie Requirements und Capabilities mittels TOSCA modelliert werden können. Ein Requirement kann an einen Node-Type, als *Requirement-Definition*, oder an ein Node-Template angeheftet werden. Wird das Requirement an einen Node-Type angeheftet, wird es bei dessen Instanziierung als Node-Template automatisch auf das Template übertragen. Das bedeutet, dass das Requirement für alle instanziierten Node-Templates des dazugehörigen Node-Types gilt. Wird das Requirement hingegen direkt an ein Node-Template angeheftet, gilt es nur für das einzelne Template. Sowohl Requirement-Definitions als auch Requirements sind typisiert durch Requirement-Types, die deren Inhalt und Struktur vorgeben. Analog verhält es sich mit Capabilities.

TOSCA Requirements definieren, mit welchen Node-Templates ein Template verbunden werden muss, um provisionierbar zu sein. Beispielsweise kann ein Javabasierter Service nur in einer entsprechenden Laufzeitumgebung (wie einem Java-Applikationsserver) betrieben werden. Um dies sicherzustellen, kann das Node-Template der Anwendung beispielsweise mit dem Requirement “requiresApplicationServer” annotiert werden.

Jedes Requirement kann durch eine oder mehrere Capabilities erfüllt werden. Im Beispiel wären dies Node-Templates für einen Apache Tomcat Applikationsserver[1] oder einen Glassfish Server [2], die die Capability “isApplicationServer” besitzen. Welche Requirements und Capabilities zusammenpassen, wird in TOSCA mittels des Attributs “requiredCapabilityType” definiert, das Teil eines Requirements ist. Ein Requirement eines Node-Templates gilt als erfüllt, wenn das Node Template, wie in Abbildung 6.2 gezeigt, über ein Relationship-Template mit einem Node-Template verbunden ist, das eine passende Capability besitzt.

Abbildung 6.3 zeigt ein Beispiel der Topologievervollständigung basierend auf den beschriebenen TOSCA Requirements.

Der grundlegende Ansatz aus [HBBL14] zur Vervollständigung der Topologie sucht nach Node-Types, die passende Capabilitys anbieten, um Requirements zu erfüllen. Aufgrund dessen, dass hierfür mehrere Node-

[1] http://tomcat.apache.org/
[2] http://glassfish.java.net/

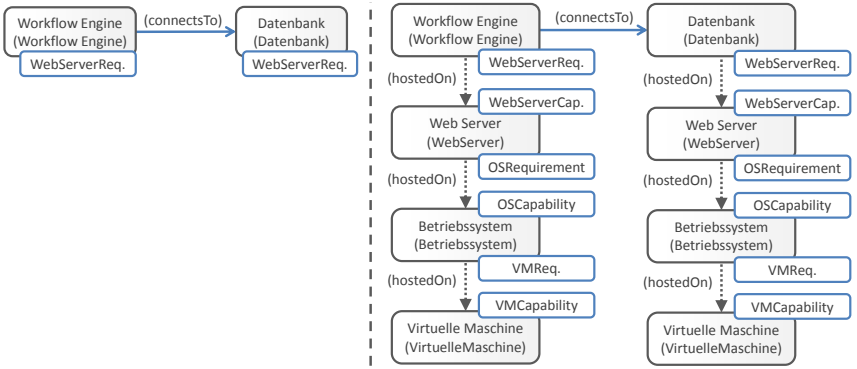

Abbildung 6.3: links: Beispiel einer unvollständigen Topologie mit Requirements, rechts: vollständige Topologie. Die dazugehörigen Node-Types sind in Klammern angegeben (basierend auf [HBBL14])

Types in Frage kommen können, muss eine Auswahl getroffen werden. Da es das Ziel ist, den Vervollständigungsprozess vollautomatisiert auszuführen, wird im Rahmen dieser Dissertation zufällig ein passender Node-Type ausgewählt. Ist ein Node-Type gefunden, dessen Capability das Requirement eines Node-Templates der Topologie erfüllt, wird es instanziiert und mittels eines passenden Relationship-Templates verbunden. Die Instanziierung kann vollautomatisch erfolgen. Hierfür wird als erstes, basierend auf dem Namen des Node-Types, ein Name für das neu erzeugte Node-Template generiert. Anschließend werden die im Node-Type definierten TOSCA-Properties im Node-Template angelegt und mit Standardwerten befüllt. Für eine automatische Provisionierung fehlen daraufhin nur noch die Artefakte.

In [HM16b] wurde eine Erweiterung der Topologievervollständigung aus [HBBL14] vorgestellt, mit der Artefakte automatisch an die Node-Templates angeheftet werden können. Die Artefakte, beispielsweise Binärdateien und Installationsskripte, müssen für jedes Node-Template in einem Repository vorliegen. Bei der Erstellung des CSARs müssen diese Artefakte hinzugefügt werden, um dessen Vollständigkeit zu gewährleisten. In TOSCA werden *Artifact-Definitions* verwendet, um die Artefakte in einem Node-Template

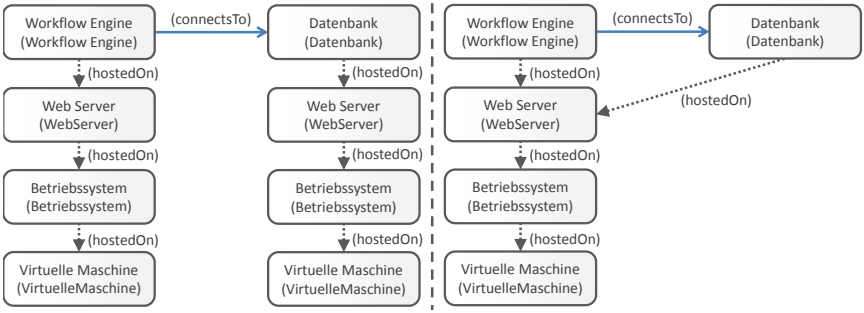

Abbildung 6.4: links: Provisionierung auf mehreren Stacks, rechts: Provisionierung auf einzelnem Stack. Die dazugehörigen Node-Types sind in Klammern angegeben

zu referenzieren. Derartige Artifact-Definitions können einfach automatisch erzeugt werden, da die Referenz innerhalb des Cloud-Service-Archives bei dessen Erstellung bekannt ist. Nach dem Ergänzen der Artefakte wird das Node-Template zur Topologie hinzugefügt.

Nach dem Einfügen eines Node-Templates in die Topologie ist es oftmals der Fall, dass die neu eingefügten Node-Templates weitere Requirements besitzen, die erfüllt werden müssen, damit die Topologie provisionierbar ist. Aus diesem Grund ist eine iterative Vorgehensweise notwendig. Das bedeutet, dass die gesamte Topologie erneut auf unerfüllte Requirements überprüft wird und diese ggf. durch das Einfügen von Node-Templates erfüllt werden. Diese Vorgehensweise wird so lange wiederholt, bis die Topologie frei von Requirements ist und somit als vollständig bzw. provisionierbar bezeichnet werden kann.

Definition 6.2 (Provisionierbarkeit/Vollständigkeit einer Topologie) *Eine Topologie gilt genau dann als provisionierbar bzw. als vollständig, wenn sie keine Requirements in den enthaltenen Node-Templates besitzt.*

Mittels der beschriebenen Vorgehensweise zur Topologievervollständigung werden alle Komponenten auf einzelnen Stacks provisioniert (vgl. Abbildung 6.3), d.h., dass jeweils Plattformkomponenten (z.B. Applikations-

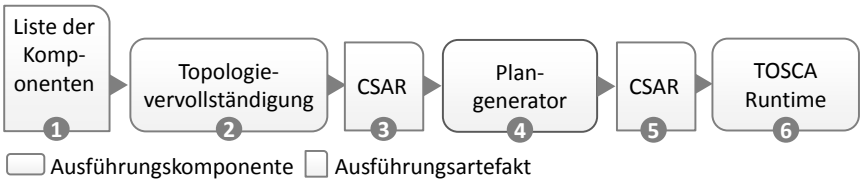

Abbildung 6.5: Ablauf der automatischen Provisionierung (basierend auf [HBBL14])

server) oder Infrastrukturkomponenten (z.B. Betriebssysteme oder virtuelle Maschinen) zur Verfügung gestellt werden. Dies ist in Abbildung 6.4 auf der linken Seite dargestellt. Die Provisionierung auf mehreren Stacks sorgt für hohe Kosten und ist nicht praktikabel, vor allem wenn Anwendungen die Ressourcen nicht vollständig aufbrauchen. Eine Provisionierung aller Komponenten auf einer gemeinsamen Infrastruktur hingegen (in Abbildung 6.4 rechts) kann zu Skalierbarkeitsproblemen führen. Die Entscheidung, welche Komponenten auf gemeinsamer Infrastruktur provisioniert werden sollten und welche nicht, stellt ein separates Problem dar und ist außerhalb des Fokus dieser Dissertation.

6.1.3 CSAR-Erstellung

Nachdem die Topologie vervollständigt ist, kann das CSAR erstellt werden. Hierfür werden die erstellte Topologie sowie alle verwendeten Typen, Policys und Artefakte in das CSAR-Format verpackt, um dann als Eingabe einer TOSCA-Ausführungsumgebung für die automatische Provisionierung verwendet zu werden.

6.2 Cloud-Provisionierung

Abbildung 6.5 zeigt den Ablauf der automatischen Provisionierung mittels einer TOSCA-Ausführungsumgebung. Grundlage der Provisionierung ist das zuvor erstellte CSAR, das alle notwendigen Komponenten enthält. Eine

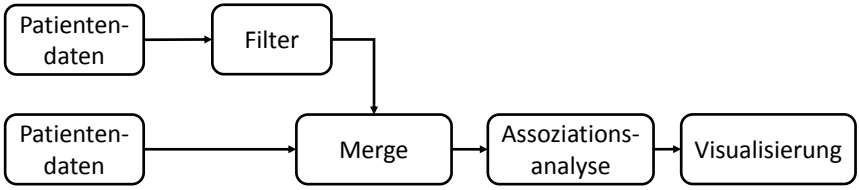

Abbildung 6.6: Beispiel-DVM zur Demonstration der Provisionierungskonzepte

deklarative TOSCA-Ausführungsumgebung kann das CSAR direkt für eine Provisionierung verwenden, da wie in Kapitel Abschnitt 2.4.2 beschrieben kein Ausführungsplan (TOSCA: Build-Plan) benötigt wird. Bei einer imperativen Ausführungsumgebung, wie beispielsweise OpenTOSCA [BBH+13], muss der Build-Plan jedoch bereitgestellt werden. Im Rahmen dieser Dissertation wurde sich für eine imperative Ausführungsumgebung entschieden. Da die Ausführungsumgebung sehr heterogen sein kann und ständig neue Datenverarbeitungsservices hinzukommen, ist die Flexibilität einer imperativen Umgebung notwendig.

Zur automatischen Generierung des Ausführungsplans wird die von Breitenbücher et al. [BBK+14a] vorgestellte Plangenerierungserweiterung von OpenTOSCA verwendet. Auf Basis der Topologie werden die enthaltenen Abhängigkeiten analysiert, um die Reihenfolge des Aufsetzens der Komponenten herauszufinden. Anschließend kann der Ausführungsplan generiert werden, indem in jedem Schritt Aufrufe für die jeweiligen Artefakte der Komponenten erstellt werden. Durch Ausführung des Plans werden die Komponenten schrittweise durch die Artefakte des CSARs in der gewählten Zielumgebung aufgesetzt.

6.3 Provisionierung der Ausführungsumgebung – Beispielanwendung

Im Folgenden wird ein Beispiel der Provisionierung einer möglichen Ausführungsumgebung gezeigt, um die Nachvollziehbarkeit der Konzepte zu

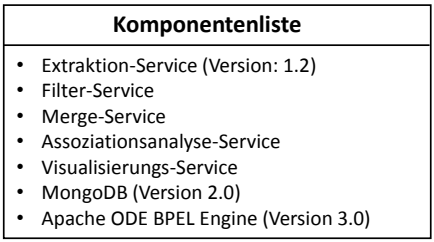

Abbildung 6.7: Komponentenliste für das Beispiel-DVM

verbessern. Dabei liegt das in Abbildung 6.6 dargestellte DVM zugrunde. Dieses wurde einfach gehalten, da die Provisionierungskonzepte im Vordergrund stehen. Ziel des modellierten Szenarios ist es, Patientendaten zweier Datenbanken zu analysieren, um zu bestimmen, welche Medikamente zu bestimmten Nebenwirkungen führen. Hierfür stehen zwei Datenbanken, von zwei verschiedenen Krankenhäusern, zur Verfügung. Da eine der Datenquellen zusätzlich Daten über die Kosten der Medikamente enthält, sollte dies gefiltert werden, um die Datenmenge zu begrenzen. Anschließend wird eine Assoziationsanalyse durchgeführt, die die Zusammenhänge zwischen Medikamenteneinnahmen und auftretenden Nebenwirkungen untersucht. Die Ergebnisse werden daraufhin visualisiert.

Der Algorithmus aus Kapitel 5 bestimmt nach der Modellierung passende Services, durch deren Komposition das Beispiel-DVM bzw. das transformierte DPM ausgeführt werden kann. Für das modellierte Beispiel könnte eine Komponentenliste wie in Abbildung 6.7 dargestellt aussehen. Hierbei wurde für jeden der modellierten Knoten im DVM ein passender Service gefunden. Des Weiteren wird für die Ausführung in diesem Beispiel eine NoSQL-MongoDB-Datenbank als Zwischenspeicher benötigt sowie eine Apache-ODE-Ausführungsengine für einen BPEL-Workflow zur Orchestrierung der Services.

Auf Basis dieser Liste wird anschließend eine Initialtopologie erstellt, die alle aufgeführten Komponenten enthält. Die Liste ist in Abbildung 6.8 dargestellt. Es ist zu sehen, dass für jeden Eintrag der Liste ein passender

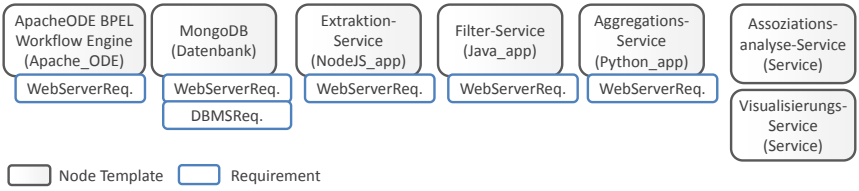

Abbildung 6.8: Initialtopologie des Anwendungsbeispiels. Die dazugehörigen Node-Types sind in Klammern angegeben

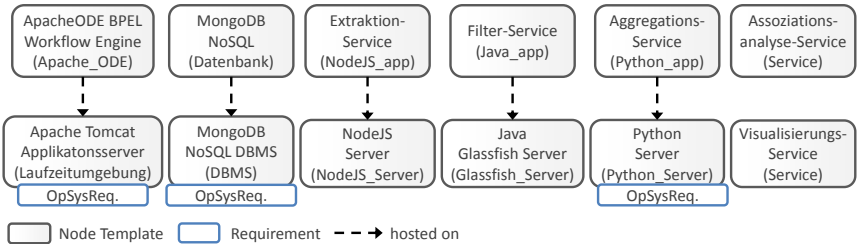

Abbildung 6.9: Topologie nach dem ersten Durchlauf des Vervollständigungs-Algorithmus. Die dazugehörigen Node-Types sind in Klammern angegeben

Node-Type im Repository aufgefunden und initialisiert wurde. Die Services können sehr heterogen sein. In diesem Beispiel existieren Services in den Programmiersprachen Java, NodeJS und Python, welche jeweils unterschiedliche Ausführungsumgebungen benötigen.

Wie in Abbildung 6.8 zu sehen ist, wurden beim Einfügen von instanziierten Node-Templates weitere Requirements hinzugefügt. Das bedeutet, dass die Topologie noch nicht vollständig und damit provisionierbar ist. Beispielsweise benötigt die ApacheODE-Worfklowengine einen Applikationsserver, die Datenbank ein Managementsystem, und die Services eine jeweils passende Laufzeitumgebung. Andere Node-Templates, in diesem Beispiel die Services zur Assoziationsanalyse und Visualisierung, besitzen hingegen keine Requirements, da die Services bspw. von einem Fremdanbieter gehostet werden, der die notwendige Infrastruktur zur Ausführung bereitstellt.

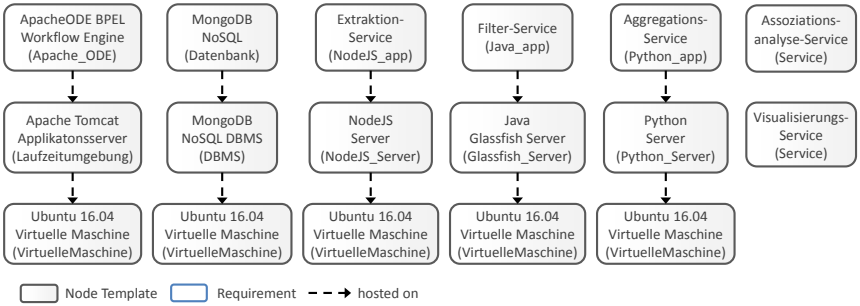

Abbildung 6.10: Vollständige Topologie des Beispielszenarios. Die dazugehörigen Node-Types sind in Klammern angegeben.

Diese Initialtopologie wird anschließend mittels dem vorgestellten Algorithmus zur Topologievervollständigung vervollständigt. Nach dem ersten Durchlauf des Algorithmus sieht die Topologie wie in Abbildung 6.9 dargestellt aus. Die bisher enthaltenen Requirements wurden durch das Einfügen von passenden Node-Templates erfüllt. Die neu eingefügten Node-Templates können, wie dargestellt, wiederum neue Requirements enthalten, die erfüllt werden müssen. Daher wird der Vervollständigungs-Algorithmus erneut ausgeführt, bis keine Requirements mehr in der Topologie vorhanden sind. Im gezeigten Beispiel könnten die Laufzeitumgebungen für die NodeJS- und Java-Services bereits von einem PaaS-Anbieter zur Verfügung gestellt werden, da die dazugehörigen Node-Templates keine Requirements besitzen.

Die vollständige Topologie, nach erneuter Erfüllung der Requirements, ist in Abbildung 6.10 dargestellt. Die vollständige Topologie enthält keine Requirements mehr und ist somit provisionierbar, im vorliegenden Fall durch die TOSCA-Ausführungsumgebung.

6.4 Verwandte Arbeiten

In diesem Abschnitt werden verwandte Arbeiten beschrieben, die einen ähnlichen Ansatz zur automatisierten Provisionierung verfolgen.

Eilam et al. [Eil+06] präsentieren einen Ansatz für modellgetriebene Pro-

visionierung und Management von Applikationen. Mittels Transformationen werden Applikationstopologien auf verschiedene Abstraktionsebenen abgebildet, bis diese vollständig bzw. provisionierbar sind. Ähnlich zu dem in dieser Dissertation vorgestellten Konzept können die entstehenden, vollständigen Topologien mittels einer Ausführungsumgebung automatisch provisioniert werden. Jedoch erfordert der von Eilam vorgestellte Ansatz die Vormodellierung von Teilen der Topologien. In meinem vorgestellten Ansatz ist keine Vormodellierung notwendig, es reicht eine textuelle Beschreibung der zu provisionierenden Komponenten. Außerdem führen Eilam et al. [Eil+11] eine Kombination eines Modellbasierten und Workflowbasierten Ansatzes für die Provisionierung von Anwendungen ein. Dabei wird ein Provisionierungsmodell für den finalen, provisionierbaren Zustand der Zieltopologie erstellt. Auf Basis des Provisionierungsmodells wird anschließend ein Workflowmodell geschaffen, das eine Initialtopologie in den Zielzustand überführt. Dieses Workflowmodell kann durch eine entsprechende Ausführungsumgebung automatisiert ausgeführt werden. Im Gegensatz zu den von mir vorgestellten Konzepten, benötigt der Ansatz von Eilam et al. ein Provisionierungsmodell für die Überführung, was wiederum manuell erstellt werden muss.

Arnold et al. [Arn+07] präsentieren einen musterbasierten Ansatz zur Provisionierung von serviceorientierten Architekturen. Hierfür werden Muster für bestimmte Szenarien vordefiniert, zum Beispiel zur Ermöglichung von hoher Verfügbarkeit. Es ist jedoch notwendig, ein derartiges Muster für alle möglichen Szenarien anzufertigen. Der von mir vorgestellte Ansatz ist generischer und nicht abhängig von bestimmten Szenarien. Nachdem eine Sammlung an Node-Types erstellt wurde, können diese beliebig für die Vervollständigung von Topologien verwendet werden.

Binz et al. [Bin+12] schaffen einen Ansatz, um mit Topologien komplexe IT-Systeme von Unternehmen abzubilden. Derartige Topologien werden als Enterprise-Topology-Graphs (ETG) bezeichnet, welche die Systemkomponenten und alle ihre Verbindungen ähnlich zu den TOSCA-Topologien beschreiben. Im Gegensatz zu TOSCA wird jedoch der Ist-Zustand der IT-Systeme abgebildet, nicht der Soll-Zustand nach einer Provisionierung.

Breitenbücher et al. [BBK+12] beschreiben Vino4TOSCA, eine visuelle

Notation für den TOSCA-Standard. Vino4TOSCA wird im grafischen TOSCA-Modellierungswerkzeug Winery [KBBL13] verwendet, in das die von mir vorgestellten Konzepte vollständig integriert wurden [HBBL14].

Wettinger [Wet17] stellt TOSCAfy vor, einen Ansatz zur einfachen Erstellung von CSARs. Dabei basiert die Erstellung auf sogenannten CSAR-Spezifikationen, ein maschinenlesbares Format, aus dem CSARs generiert werden können. Hierdurch ist es nicht länger notwendig, das vollständige CSAR abzuspeichern und zu verwalten. Es reicht, lediglich dessen Spezifikation zu speichern und die CSARs dynamisch bei Bedarf zu generieren. Im Gegensatz zu Wettinger steht in meinem Ansatz anfangs nicht fest, wie das CSAR aussehen wird, da die Auswahl der Komponenten basierend auf den Anforderungen im DVM$^+$ sowie auf den TOSCA-Requirements und -Capabilities getroffen wird. Es wäre denkbar, in zukünftigen Arbeiten, statt CSARs, CSAR-Spezifikationen zu generieren und lediglich die Spezifikationen abzuspeichern. Wird dann exakt dieselbe Ausführungsumgebung benötigt, kann das CSAR dann einfach aus der Spezifikation generiert werden.

Ich habe mich für die Verwendung des TOSCA-Standards entschieden, da es der einzige Standard für die Provisionierung und das Management von Anwendungen darstellt. Technologien wie Docker [Doc], Vagrant [Has] oder Ansible [Hat] sind zwar weit verbreitet, jedoch nicht standardisiert, was deren Zukunftssicherheit mindert. Des Weiteren können die Provisionierungsbeschreibungen nicht einfach zwischen mehreren Partnern ausgetauscht werden. Ein weiteres Argument für TOSCA ist die von Breitenbücher et al. [BBK+12] vorgestellte grafische Notation für den TOSCA-Standard Vino4TOSCA sowie das auf dieser Notation basierende von Kopp et al. [KBBL13] entwickelte Modellierungswerkzeug Winery. Die grafische Darstellung schafft nicht nur eine einfache Modellierung von Topologien, sondern auch eine gute Nachvollziehbarkeit, wie die zu provisionierenden Komponenten aufgebaut sind. Skriptbasierte Ansätze wie Docker, Vagrant, usw. führen bei komplexen Anwendungen schnell zu Unübersichtlichkeiten und damit zu Schwierigkeiten hinsichtlich der Nachvollziehbarkeit.

6.5 Implementierung und Evaluation

Um die Konzepte zu validieren, wurde ein Prototyp erstellt. Der Prototyp wurde teils eigens, teils durch die von mir betreute studentische Arbeit von Del Gaudio [Del16] entwickelt. Der Prototyp basiert auf der XML-Darstellung des TOSCA-Standards in der Version 1.0 und ist Open-Source verfügbar im Eclipse-Projekt Winery [KBBL13]. Wir nutzen bei der Implementierung die imperative TOSCA-Ausführungsumgebung OpenTOSCA, die einen Ausführungsplan benötigt. Die Ansätze und auch der Prototyp sind auf andere TOSCA-Ausführungsumgebungen übertragbar.

Der Prototyp basiert vollständig auf der Programmiersprache Java und stellt eine eigene Komponente dar, die einfach in das bestehende OpenTOSCA-Ökosystem, genauer in das Modellierungswerkzeug Winery, integriert wurde, ohne dass große Anpassungen notwendig waren. Winery dient neben der Funktion als Modellierungswerkzeug auch als Repository für Node-Types, Relationship-Types und Artefakte. Die Eingabe für die Komponente ist wie in den vorigen Abschnitten beschrieben die textuelle Beschreibung der zu provisionierenden Komponenten. Zuerst wird das Repository Winery nach vorhandenen Node- und Relationship-Types durchsucht. Winery speichert TOSCA-Typen als XML-Beschreibungen. Diese XML-Beschreibungen werden anschließend mittels der Java Architecture for XML Binding (JAXB) in Java-Objekte transformiert, um programmatisch mittels Java verarbeitbar zu sein. Basierend auf diesen Java-Objekten können nun die Eigenschaften der Node- und Relationship-Types extrahiert und auf deren Basis die Node-Templates instanziiert werden. Es ist zu beachten, dass in den Node-Types Defaultwerte für die Eigenschaften der zu instanziierenden Node-Templates hinterlegt sein müssen. Die Defaultwerte können in der XML-Schema-Definition der TOSCA-Properties festgelegt werden. Mittels JAXB können neue Node-Template-Elemente einfach erzeugt werden. Dabei werden eindeutige Namen basierend auf den Node-Type-Namen generiert und die Eigenschaften mit den Defaultwerten eingefügt. Außerdem werden alle Requirements der Node-Types in die Templates übernommen.

Nachdem das Node-Template instanziiert wurde, müssen die Artefakte,

also die Codefragmente die die Softwarekomponenten aufsetzen, zu den Templates hinzugefügt werden. Hierfür gibt es in TOSCA Artifact-Templates, die über die Implementierung der Node-Types referenziert werden. Jedes Artifact-Template verweist auf ein konkretes Codeartefakt. Das Artifact-Template-Element kann einfach mittels JAXB dem neu instanziierten Template hinzugefügt werden. Eine TOSCA-Ausführungsumgebung kann das Artefakt dann für die Provisionierung der Komponente verwenden.

Die Instanziierung der Relationship-Templates wird analog durchgeführt. Anschließend werden alle Node und Relationship-Templates in eine TOSCA Topologie eingefügt.

Die vorliegende Topologie wird dann auf in ihr enthaltene Requirements überprüft. Ist kein Requirement vorhanden, kann die Topologie bereits als provisionierbar bezeichnet werden. Sind Requirements vorhanden, wird für jedes Requirement ein Node-Type gesucht, dass eine passende Capability besitzt. Hierfür werden die XML-Beschreibungen der Node-Types über die REST-Schnittstelle des Winery-Repositorys angefragt. Diese Node-Types werden traversiert, bis ein oder mehrere passende Node-Types mit entsprechender Capability gefunden wurden. Sollten mehrere Node-Types in Frage kommen, wird im Prototyp jeweils der erste passende Typ ausgewählt. Anschließend wird mittels JAXB ein Node-Template instanziiert. Um das Requirement zu erfüllen, müssen die Node-Templates verbunden werden. Hierfür muss ein in Frage kommender Relationship-Type gefunden werden. Welche Relationship-Types für die Verbindung zweier Node-Templates in Frage kommen, kann mittels deren Elemente *Valid-Source* bzw. *Valid-Target* festgelegt werden. Valid-Source gibt an, welche Typen von Node-Templates als gültige Quelle für den Relationship-Type verwendet werden können, und Valid-Target legt fest, welche Node-Templates als Ziel des Relationship-Types gültig sind.

Analog zur Auswahl der Node-Templates wird bei mehreren möglichen Typen jeweils der erste ausgewählt. Anschließend kann das Relationship-Template instanziiert werden, indem das Quell- und Ziel-Node-Template für das Relationship Template angegeben wird sowie ein eindeutiger Name generiert wird. Das Relationship-Template wird dann in die Topologie ein-

gefügt und das Requirement am Node-Template entfernt, da es durch eine passende Capability erfüllt wurde.

Die Suche und Instanziierung von Node- und Relationship-Types wird für jedes Requirement wiederholt. Anschließend wird die neu entstandene Topologie erneut auf vorhandene Requirements überprüft, da durch die Instanziierung neuer Node-Templates weitere hinzukommen können. Sind Requirements vorhanden, werden sie wie oben beschrieben erfüllt. Diese Vorgehensweise wird so lange wiederholt, bis die Topologie keine Requirements mehr besitzt und somit als vollständig bzw. provisionierbar definiert werden kann.

Anschließend wird die Schnittstelle des Werkzeugs Winery genutzt, um basierend auf der vervollständigten Topologie ein CSAR zu erstellen. Dabei werden die Topologie sowie alle involvierten Typen und Artefakte in eine ZIP-Datei verpackt, die als Eingabe für die OpenTOSCA-Ausführungsumgebung verwendet werden kann. Diese Funktionalität ist in Winery vorhanden und musste nicht im Prototypen bereitgestellt werden.

Damit die Provisionierung durch die imperative OpenTOSCA-Umgebung durchgeführt werden kann, muss ein Ausführungsplan generiert werden. Eine derartige Generierung wurde von Képes [Kep13] implementiert und vollständig in OpenTOSCA integriert. Bei der Eingabe des CSAR in OpenTOSCA wird automatisch überprüft, ob das CSAR bereits einen Ausführungsplan enthält. Ist dies nicht der Fall, wird die Plangenerierung gestartet. Anschließend wird der erstellte Plan dem CSAR hinzugefügt und das CSAR der OpenTOSCA-Ausführungsumgebung übergeben. Über die Schnittstellen der Vinothek kann der Ausführungsplan gestartet werden, woraufhin die Komponenten aufgesetzt werden.

Zusätzlich zur vorgestellten Vorgehensweise zur automatisierten Topologievervollständigung wurde eine Möglichkeit geschaffen, die Topologie schrittweise, nutzergesteuert durchzuführen. Dies ist in Abbildung 6.11 dargestellt. Hierbei wird der Nutzer, nach Modellierung der anwendungsspezifischen Komponenten, schrittweise über einen Wizard durch die Vervollständigung geführt. In jedem Schritt wird für ein Requirement der Topologie abgefragt, durch welches Node-Template mit passender Capability es erfüllt

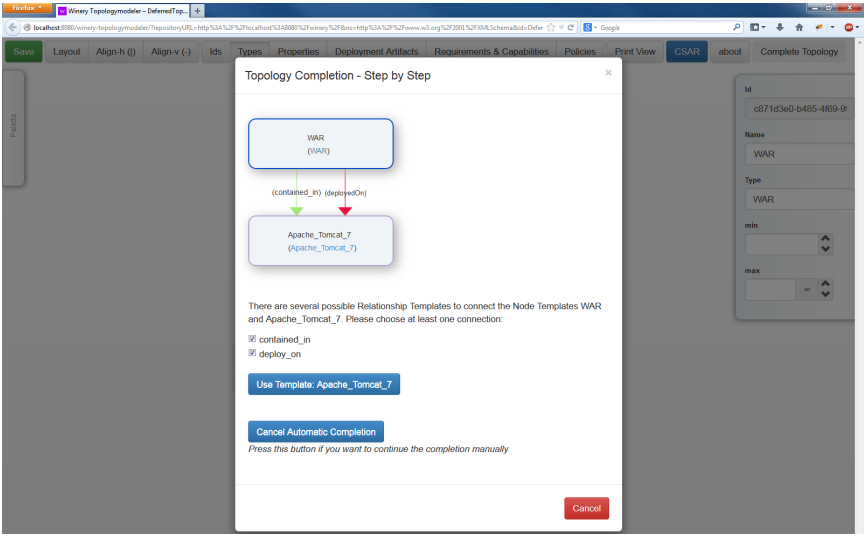

Abbildung 6.11: Implementierung der nutzergesteuerten Topologievervollständigung (aus [HBBL14])

werden soll. Zusätzlich können die zu verwendenden Relationship-Templates angegeben werden. Die manuell gesteuerte Vervollständigung macht vor allem dann Sinn, wenn der Modellierer der Topologie klare Vorstellungen hat, wie die Anwendung aufgesetzt werden soll. Zum Beispiel könnte der Modellierer bereits einen Zugang eines bestimmten Cloudanbieters besitzen, der für das Aufsetzen der Anwendung verwendet werden soll.

Die beschriebene Implementierung ist Open-Source und integriert in das Eclipse-Projekt Winery. Eine umfassende Dokumentation der Topologievervollständigung sowie ein Tutorial sind online verfügbar[1].

6.6 Zusammenfassung

In diesem Beitrag wird vorgestellt, wie komplexe Ausführungsumgebungen für die Datenverarbeitung vollautomatisiert basierend auf dem TOSCA-

[1] http://eclipse.github.io/winery/user/TopologyCompletion

Standard aufgesetzt werden. Als Grundlage hierfür dient eine Beschreibung der für die Datenverarbeitung benötigten Services, die automatisch durch die in Kapitel 5 vorgestellten Konzepte erzeugt wurde. Diese Konzepte umfassen neben den für die Datenverarbeitung benötigten Services auch Laufzeitumgebungen, um das transformierte DVM$^+$ auszuführen oder Datenspeicher, um Zwischenergebnisse zu persistieren. Abhängig davon, ob die Services selbst zur Verfügung gestellt werden oder ob sie von Fremdanbietern (z.B. PaaS-Anbietern) stammen, müssen zusätzliche Komponenten wie Applikationsserver, Datenbanken, Betriebssysteme oder virtuelle Maschinen aufgesetzt werden, damit diese lauffähig sind.

Die Beschreibung der aufzusetzenden Komponenten wird verwendet, um eine TOSCA-Topologie automatisch zu erstellen. Hierfür wurde ein Konzept einer automatisierten Topologieerstellung geschaffen, die, basierend auf einem Repository, alle für eine Provisionierung notwendigen Node- und Relationship-Types sucht, instanziiert und in die Topologie einfügt.

Zusammenfassend wird in diesem Beitrag ein Ansatz beschrieben, um TOSCA-Topologien nur basierend auf einer Komponentenliste zu erstellen und die Topologie für eine automatische Provisionierung von Anwendungen zu verwenden. Dabei existieren keine Domänenabhängigkeiten. Der vorgestellte Ansatz ist allgemein einsetzbar, wie im Beispiel aus Abschnitt 6.3 oder in Kapitel 9 beschrieben.

Kapitel 7

Ressourcen-Management-Plattform

Die Ressourcen-Management-Plattform (RMP) stellt einen wichtigen Beitrag dieser Dissertation dar. Der Hauptzweck der RMP ist das automatisierte Anschließen von Datenquellen zur Datenextraktion und -bereitstellung. Darüber hinaus bietet die RMP Funktionen, um Daten zu transformieren und zu reinigen, wodurch deren Qualität erhöht werden kann. Dabei kann die RMP als Zwischenschicht zwischen den Daten und deren Verarbeitung gesehen werden, die von den konkreten Datenquellen, mit all ihren Details über Schnittstellen und Zugriffsprotokollen, abstrahiert. Um eine derartige Abstraktion zu ermöglichen, bietet die RMP eine uniforme Schnittstelle zu den darunterliegenden Daten an. Die Datenkonsumenten, beispielsweise die in Kapitel 8 beschriebene Datenverarbeitung, greifen also nur noch auf die RMP und nicht auf die Datenquellen direkt zu. Das hat große Vorteile, vor allem, wenn sich die Schnittstellen zu den Datenquellen ändern.

Die Hauptbeiträge der RMP sind: (1) automatisches Anbinden von Datenquellen und Datensenken, (2) effektive Datenzwischenspeicherung, um

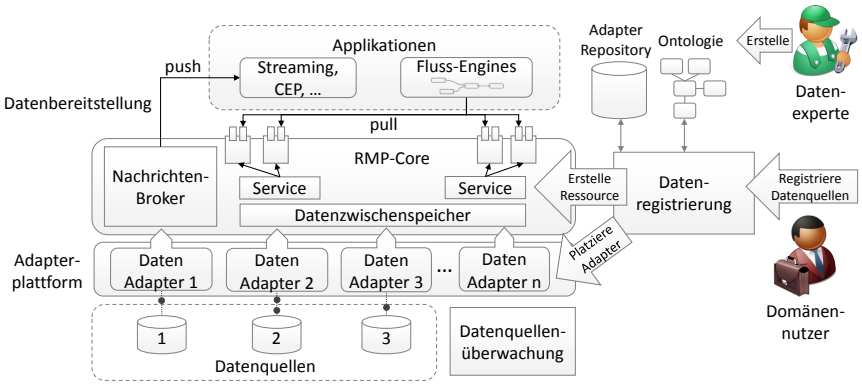

Abbildung 7.1: Architektur der Ressourcen-Management-Plattform (aus [HWBM16b], die gestrichelten Komponenten sind nicht Teil dieser Architektur)

Ausfälle von Datenquellen zu kompensieren, (3) Datentransformation und Bereinigung zur Steigerung der Datenqualität, und (4) eine Überwachung der Datenquellen und Datensenken, um auf Fehlersituationen, etwa deren Wegfallen, reagieren zu können. Des Weiteren weist die RMP aufgrund ihres stark modularen Aufbaus eine hohe Verteilbarkeit und Skalierbarkeit auf. Aufgrund dessen, dass jede Komponente durch standardisierte Schnittstellen lose mit den anderen Komponenten gekoppelt ist, können sie dupliziert und auf verschiedenen Laufzeitumgebungen verteilt betrieben werden.

Die Architektur der RMP ist in Abbildung 7.1 dargestellt. Die Architektur besteht aus drei Hauptkomponenten: 1) die Datenregistrierungskomponente, 2) die Adapterplattform und 3) die Überwachungskomponente.

Mit der *Datenregistrierungskomponente* können neue Datenquellen registriert werden. Diese Komponente enthält ein Repository für *Adapter*, die für das Anbinden von Datenquellen verwendet werden sowie eine Ontologie mit semantischen Informationen zu den Datenquellen. Ein Adapter stellt eine Softwarekomponente dar, die in der Lage ist, zu den Datenquellen zu verbinden, deren Daten zu extrahieren und weiterzugeben. Meist handelt es sich hierbei um leichtgewichtige Skripte.

Zusätzlich zur Registrierungskomponente existiert eine *Adapterplattform*, auf der die Adapter automatisch bereitgestellt werden. Hierfür kommt ebenfalls der TOSCA-Standard zur Anwendung. Bei der Adapterplattform kann es sich um ein verteiltes Ökosystem handeln, das verschiedene Laufzeitumgebungen für die Adapterimplementierungen zur Verfügung stellt.

Die Adapter senden die extrahierten Daten an die Hauptkomponente der RMP, genannt *RMP-Core*, die für die uniforme Datenprovisionierung zuständig ist sowie für das Zwischenspeichern, die Transformation und das Bereinigen der Daten. Neben Funktionen, um Daten zwischen zu speichern sowie zu transformieren und zu bereinigen, stellt der Hauptzweck der RMP-Core-Komponente die Datenprovisionierung bzw. Datenbereitstellung für darüber liegende Anwendungen dar. Hierfür wurden zwei Möglichkeiten geschaffen: einerseits ein anfragebasierter Ansatz, bei dem Daten über eine REST-Schnittstelle zur Verfügung gestellt werden und andererseits ein Publish-Subscribe-basierter Ansatz [EFGK03], bei dem sich Applikationen auf Daten registrieren können und bei Änderungen benachrichtigt werden.

Letztendlich sorgt eine Überwachungskomponente dafür, dass der Zustand der Datenquellen jederzeit überprüft werden kann. Beispielsweise kann durch ein rechtzeitiges Bemerken des Wegfalls von Datenquellen entsprechend reagiert werden, sodass Fehler oder das Arbeiten auf veralteten Daten vermieden werden können.

Bei der Nutzung der Architektur existieren zwei Rollen: der Datenexperte, der umfangreiches technisches Wissen über die Datenquellen besitzt, und Domänennutzer, die daran interessiert sind, Datenquellen zu registrieren, um deren Daten zu nutzen. Die Trennung der beiden Rollen sorgt dafür, dass es einem breiten Kreis von Anwendern erlaubt wird, Daten zur Verfügung zu stellen. Dabei werden Detailinformationen der Datenquellen (z.B. MySQL) sowie die dazugehörigen Adapter durch den Datenexperten zur Verfügung gestellt. Die Bereitstellung der Adapter ermöglicht es, die Registrierung durch Domänennutzer einfach zu gestalten.

Im Rahmen dieser Dissertation wird die RMP für das Anbinden der im DVM modellierten Datenquellen verwendet. Darüber hinaus kann die RMP jedoch auch eigenständig, in unterschiedlichen Domänen (Internet der Din-

ge, Businessapplikationen, etc.) für das Verwalten von Datenquellen und Datensenken, angewendet werden.

Die RMP wird wie folgt definiert:

Definition 7.1 (Ressourcen-Management-Plattform (RMP))
Die Ressourcen-Management-Plattform stellt eine Komponente zur Abstraktion von Datenquellen dar. Neben dem reinen Datenzugriff über eine uniforme Schnittstelle bietet die RMP die Möglichkeit Datenquellen dynamisch anzuschließen, diese zu überwachen sowie Daten zu transformieren und zu säubern.

Die Konzepte im Zusammenhang mit der RMP werden im Folgenden genauer beschrieben. Diese Konzepte basieren auf den Veröffentlichungen [HBS+16; HWBM16a; HWBM16b]. Dabei wurden in diesen Veröffentlichungen neben den in diesem Kapitel vorgestellten Beiträgen weitere Konzepte der Koautoren etwa zur Modellierung eines digitalen Abbilds der realen Welt sowie der einfachen Provisionierung mittels TOSCA beschrieben.

7.1 Automatisches Anbinden von Datenquellen

IT-Umgebungen zeichnen sich durch eine sehr hohe Dynamik aus. Es kommen häufig neue Datenquellen hinzu, z.B. durch die Einführung neuer Softwaresysteme, durch die Durchführung von Simulationen oder dem Kauf neuer Hardware, ausgestattet mit datenproduzierenden Sensoren [HHM17a]. In ähnlichem Maße fallen jedoch auch Datenquellen weg, entweder durch Systemausfälle oder durch absichtliches Entfernen. Zusätzlich ändern sich die Datenquellen ständig. Dies betrifft zum einen die Daten selbst, also ihre Struktur, zum anderen aber auch die Schnittstellen zu den Datenquellen, beispielsweise durch das Austauschen von Datenbanksystemen.

Um mit der beschriebenen Dynamik umzugehen, wurde ein Konzept entworfen, um Datenquellen automatisch und dynamisch anzubinden. Dieses Konzept basiert auf der Topology and Orchestration Specification for Cloud Applications (TOSCA). Das Anbinden von Datenquellen wird durch die in Abbildung 7.2 dargestellten Methode durchgeführt. Die Methode nutzt

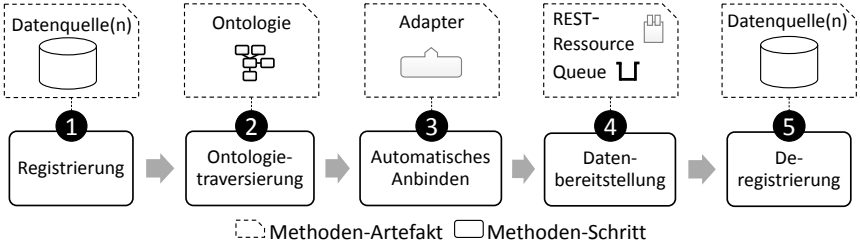

Abbildung 7.2: Methode zur automatischen Anbindung von Datenquellen (basierend auf [HWBM16b])

die Komponenten der dargestellten Architektur (siehe Abbildung 7.1). Der Methode liegt das in Kapitel 4 beschriebene DVM bzw. DVM$^+$ zugrunde, insbesondere die darin enthaltene Beschreibung der Datenquellen.

7.1.1 Schritt 1: Registrierung von Datenquellen

Im ersten Schritt der Methode werden die im DVM enthaltenen Datenquellen an der Ressourcen-Management-Plattform registriert. Dabei können der Typ der Datenquellen sowie deren Zugriffsdaten dem Modell entnommen werden. Die Zugriffsdaten umfassen beispielsweise Anmeldedaten für die Authentifizierung oder Zugriffspfade. Detailinformationen über die Datenquellen sind in einer Ontologie abgespeichert, die im nächsten Schritt genauer beschrieben wird. Daher reicht es, lediglich wenige Informationen über die Datenquelle im Modell anzugeben. Wir gehen im Folgenden davon aus, dass alle modellierten Datenquellen in der Ontologie repräsentiert sind. Ist dies nicht der Fall, muss bei der Registrierung eine passende, schemakonforme Ontologie der Datenquelle angegeben werden. Die Erstellung der Ontologie wird von Experten vorgenommen, die spezifisches Wissen über die Datenquellen besitzen.

7.1.2 Schritt 2: Ontologietraversierung

Im zweiten Schritt werden Detailinformationen über die registrierte Datenquelle aus der Ontologie extrahiert. Die Auswahl eines ontologiebasierten Formates, beispielsweise im Gegensatz zu einem relationalen Format, wurde vor allem aufgrund der hohen Erweiterbarkeit sowie einer einfachen Verknüpfbarkeit mit weiteren Ontologien getroffen. Die aus der Ontologie extrahierten Detailinformationen enthalten beispielsweise Informationen über die Schnittstellen der Datenquelle, deren Protokolle und Zugriffssprachen (z.B. SQL). Des Weiteren enthält die Ontologie eine Referenz eines oder mehrerer zur Datenquelle zugehöriger Adapter, die im oben genannten Adapterrepository (vgl. Abbildung 7.1) hinterlegt sind. Der Adapter dient als Grundlage für das Anbinden der Datenquelle. Die Ontologie ist beispielhaft in Abbildung 7.3 mittels der Web-Ontology-Language (OWL) dargestellt. Dabei ist zu sehen, dass eine Datenquelle als Unterklasse eines Datenobjekts gesehen werden kann. Eine Datenquelle erbt die Attribute: 1) *ID*, ein eindeutigen Bezeichner, 2) *name*, ein nicht eindeutiger Name, und 3) *type*, der Typ der Datenquelle. Weitere, datenquellenspezifische Attribute können im Objekt *DataSource* hinzugefügt werden. Zusätzlich verweisen ein oder mehrere Adapter auf das *DataSource*-Objekt.

7.1.3 Schritt 3: Automatisches Anbinden der Datenquelle(n)

Nachdem der Adapter sowie zusätzliche Informationen über die Datenquellen aus der Ontologie extrahiert wurden, können die Datenquellen im nächsten Schritt automatisch angebunden werden. Das Anbinden geschieht über das parametrisierte Provisionieren eines Adapters. Hierfür kommt TOSCA zum Einsatz. Ein Adapter bezeichnet ein Softwareprogramm, das in der Lage ist, Daten aus der Quelle zu extrahieren und an die RMP weiterzugeben. Um das Anbinden umzusetzen, wird als erstes pro Datenquelle ein Adapter aus dem Adapterrepository extrahiert. Der Typ des Adapters kann der Ontologie entnommen werden. Ein Adapter ist jeweils als TOSCA-Cloud-Service-Archive (CSAR) im Adapterrepository hinterlegt. Dabei enthält jedes

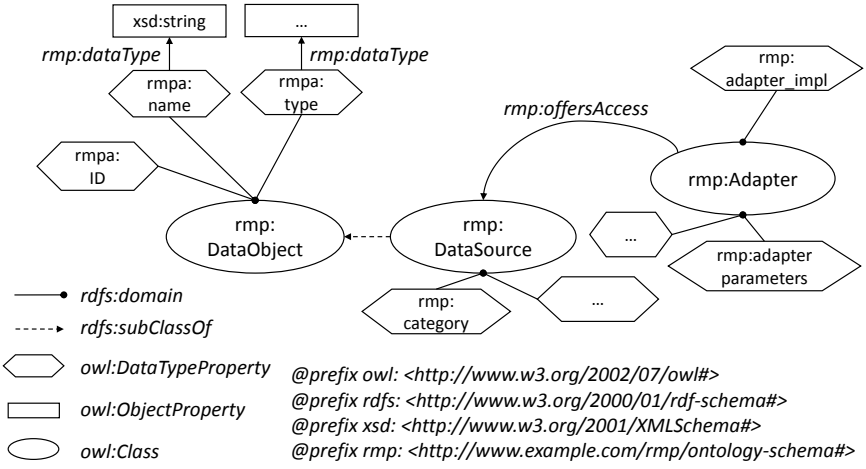

Abbildung 7.3: Datenquellen-beschreibende Ontologie – Auszug zur Beschreibung einer Datenquelle (basierend auf [HWBM16b])

CSAR eine Topologie, die den Aufbau und die Abhängigkeiten der Adapter beschreibt. Die Abhängigkeiten umfassen beispielsweise die verwendeten Bibliotheken (Datenbanktreiber etc.) sowie Anforderungen an die darunterliegende Infrastruktur (Laufzeitumgebungen etc.). Im Regelfall werden alle Adapter-CSARs in einer gemeinsamen Infrastruktur provisioniert, da die Adapter sehr leichtgewichtig sind und dadurch Kosten gespart werden können. Jedoch besteht auch die Möglichkeit, jedes CSAR auf einer eigenen Infrastruktur, zum Beispiel auf einer eigenen virtuellen Maschine, zu provisionieren. Hierdurch können die verfügbaren Ressourcen einfacher skaliert werden. Durch die Verwendung von TOSCA ist es außerdem möglich, mehrere Instanzen eines Adapters aufzusetzen, um etwa für Lastverteilung oder für Ausfallsicherheit zu sorgen.

Abbildung 7.4 zeigt ein Beispiel einer Adaptertopologie, um eine MySQL-Datenbank anzuschließen. Die Topologie enthält fünf Node-Templates für die Infrastruktur-, Plattform- und Anwendungskomponenten, die für die Provisionierung des Adapters notwendig sind. Dabei ist zu sehen, dass dieser Adapter auf einem bereits existierenden Betriebssystem in einer virtuellen

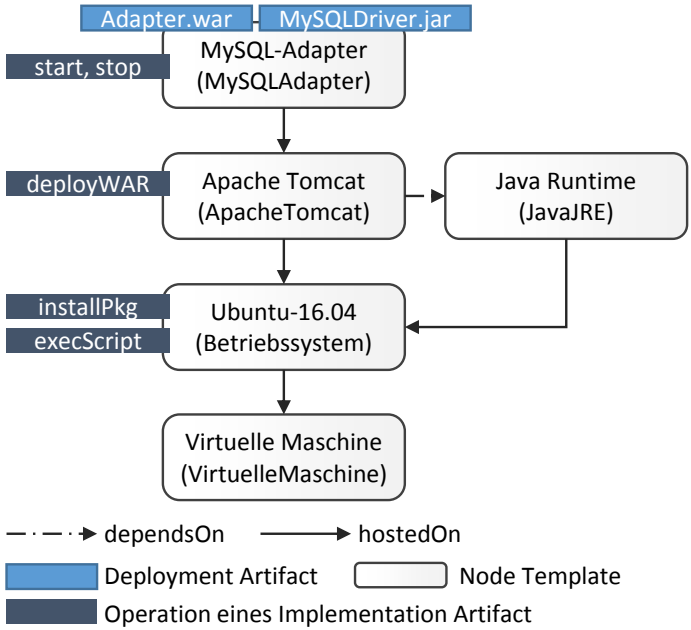

Abbildung 7.4: TOSCA-Topologie eines Adapters zum Anschließen einer MySQL-Datenbank. Die dazugehörigen Node-Types sind in Klammern angegeben

Maschine provisioniert werden soll. Damit der in Java geschriebene Adapter lauffähig ist, müssen sowohl ein Java-Applikationsserver (in diesem Fall Tomcat) als auch eine Java-Laufzeitumgebung (Java JRE) existieren bzw. installiert werden. In der Topologie ist zu sehen, dass der Applikationsserver die Java-Laufzeitumgebung benötigt, um lauffähig zu sein. Diese Abhängigkeit wird durch das *dependsOn* Relationship-Template ausgedrückt.

Das Node-Template des Adapters enthält zwei TOSCA-Deployment-Artefakte: (1) der Adapter selbst als Java-Web-Archive (WAR) und (2) die Bibliothek für den Zugriff auf die MySQL-Datenbank, die von der *Adapter.war* genutzt wird. Des Weiteren besitzt das Node-Template ein Implementation-Artifact, das für das Starten und Stoppen des Adapters verwendet werden kann. Diese Ad-

aptertopologie kann anschließend für die Provisionierung in einer passenden TOSCA-Laufzeitumgebung verwendet werden. Dadurch wird der Adapter automatisch bereitgestellt und gestartet. Sobald diese Schritte durchgeführt wurden, verbindet sich der Adapter zur Datenbank, liest die Daten aus, und schickt sie an die Hauptkomponente der Ressourcen-Management-Plattform weiter. Die Bereitstellung der Daten wird im nächsten Schritt beschrieben.

7.1.4 Schritt 4: Datenprovisionierung

Im vierten Schritt der Methode werden die Daten der angebundenen Datenquelle der DPM-Ausführungsumgebung bereitgestellt. Hierfür bietet die RMP zwei Möglichkeiten: (1) ressourcenbasierte Datenbereitstellung und (2) nachrichtenbasierte Datenbereitstellung. Bei der ressourcenbasierten Datenbereitstellung werden Ressourcen generiert, die dem REST-Prinzip folgen und die Daten repräsentieren. Es ist wichtig, dass jeweils nur lesender Zugriff auf die Daten möglich ist. Das Einfügen, Manipulieren und Löschen der Daten darf nicht möglich sein. Ein derartiger Datenzugriff wird als *pull*-basierter Ansatz bezeichnet, bei dem Anwendungen Daten bei Bedarf abfragen können. Zusätzlich wurde mittels dem *Hypermedia As The Engine Of Application State* (HATEOAS)-Ansatz [VB15] die Navigation durch die REST-Ressourcen ermöglicht, ohne dass Anwendungen die genaue Schnittstellenspezifikation kennen müssen.

Darüber hinaus wurde eine zweite Möglichkeit geschaffen, um auf die Daten zuzugreifen. Im Gegensatz zum pull-basierten Ansatz mittels Ressourcen, verfolgt die nachrichtenbasierte Datenbereitstellung einen *push*-basierten Ansatz. Das bedeutet, dass die RMP aktiv Daten zur Verfügung stellt sobald eine Datenquelle angeschlossen wird bzw. sobald sich Daten ändern. Diese Vorgehensweise ist insbesondere für Anwendungen von Vorteil, die Daten strombasiert verarbeiten, wie zum Beispiel Complex-Event-Processing-Systeme. Ein derartiger Ansatz wird als *Publish-Subscribe* [EFGK03] bezeichnet. Die Anwendungen registrieren sich auf die Daten über sogenannte *Topics*. Topics sind hierarchische Pfade, die Daten eindeutig identifizieren. Beispielsweise könnte ein Topic für das Beispiel einer MySQL-Datenbank, die eine Tabelle

mit Produktinformationen anbietet, wie folgt aussehen: *MeineMySQLDatenbank/Produkte*. Enthält diese Datenbank außerdem Kundeninformationen könnten diese unter einem Topic *MeineMySQLDatenbank/Kunden* verfügbar sein. Registriert sich eine Anwendung auf eines der Topics, sendet die RMP automatisch Daten, sobald sie entstehen. Im Rahmen dieser Dissertation ist vor allem der pull-basierte Ansatz zur Datenabfrage relevant.

Damit Applikationen, die auf Daten über die RMP zugreifen möchten, einen Überblick bekommen, welche Datenquellen zugreifbar sind bzw. welche Daten diese enthalten, bietet die RMP einen *Datenkatalog* (aus Gründen der Übersichtlichkeit nicht in Abbildung 7.1 dargestellt). Dieser Katalog enthält eine Auflistung aller registrierten und angeschlossenen Datenquellen sowie deren Detailinformationen (Datenformat, Zugriffsrechte, etc.), extrahiert aus der Ontologie (vgl. Schritt 2). Somit können sich Applikationen einen Überblick der verfügbaren Datenquellen verschaffen. Darüber hinaus ermöglicht die RMP eine Exploration der Daten über den Datenkatalog oder eine Datenvorschau. Dadurch können Applikationen die am besten geeigneten Daten finden und auf diese anschließend über die oben genannten Schnittstellen zugreifen.

7.1.5 Schritt 5: Deregistrierung der Datenquelle

Im letzten Schritt der Methode können Datenquellen wieder deregistriert werden. Der Deregistrierungsschritt funktioniert ähnlich wie der Registrierungsschritt. Es wird lediglich der eindeutige Identifikator der angeschlossenen Datenquelle angegeben. Hierfür kommen in TOSCA sogenannte *Termination-Plans* [BBLS12] zum Einsatz, wobei es sich um Workflows handelt, die das Herunterfahren der Softwarekomponenten umsetzen. Daraufhin wird der dazugehörige Adapter von der Adapterplattform entfernt, die dazugehörigen Ressourcen werden gelöscht, und die entsprechenden Topics, falls vorhanden, entfernt. Das Deregistrieren von Datenquellen sorgt für Übersichtlichkeit und spart kostenintensive Ressourcen.

7.2 Datenzwischenspeicherung und Überwachung

Wurden Datenquellen mit der beschriebenen Methode angeschlossen, können deren Daten über die Schnittstellen der Ressourcen-Management-Plattform abgefragt werden. Werden die gleichen Daten wiederholt bei jeder Anfrage aus den Datenquellen extrahiert und gegebenenfalls transformiert und bereinigt, sorgt das für Effizienzeinbusen. Aus diesem Grund ermöglicht es die RMP, Daten in Caches zwischenzuspeichern. Der Zwischenspeicher wird in regelmäßigen Abständen aktualisiert, woraufhin Daten extrahiert, transformiert und bereinigt sowie anschließend abgespeichert werden. Manche Datenquellen bieten die Möglichkeit zu überprüfen, ob bzw. was sich an den Daten geändert hat. Ist dies der Fall, wird der Zwischenspeicher nur mit den geänderten Daten aktualisiert. Bietet die Datenquelle eine derartige Möglichkeit nicht an, werden die enthaltenen Daten regelmäßig im Zwischenspeicher aktualisiert. Das Caching kann für jede Datenquelle einzeln (de-)aktiviert werden, per Default ist Caching aktiviert.

Neben der Datenzwischenspeicherung bietet die RMP die Möglichkeit den Zustand der Datenquellen zu überwachen, also ob diese erreichbar sind oder nicht. Hierfür wurde eine Überwachungskomponente integriert. Diese Komponente sendet in regelmäßigen Abständen Nachrichten, sogenannte *Heartbeats*, an die Datenquellen, um zu überprüfen, ob sie noch verfügbar sind. Ist dies nicht der Fall, werden die dazugehörigen Dateneinträge im Cache als "veraltet" markiert. Ist das Caching deaktiviert, wird eine entsprechende Fehlermeldung beim Ressourcenzugriff ausgegeben.

7.3 Datentransformation und Datenqualität

Eine weitere Funktion der RMP ist die Datentransformation. Hierbei werden Daten in ein uniformes Format überführt. Aufgrund der Heterogenität der Datenquellen, die von unstrukturierten Textdaten über strukturierte relationale Datenbanken reichen, müssen die Daten für eine effektive Verarbeitung und insbesondere für deren Aggregation zur Analyse in ein einheitliches Format überführt werden. Dieses Format ist strukturiert, was dazu führt,

dass unstrukturierte Daten erst in eine strukturierte Repräsentation überführt werden müssen. Um eine Strukturierung der Daten zu ermöglichen, kommen *Natural-Language-Processing (NLP)*-Techniken [FS06] zum Einsatz. Diese ermöglichen Tokenisierung – das Aufspalten von Sätzen in Wörter und Satzzeichen – sowie das Erkennen von Satzelementen. Dadurch wird es ermöglicht, Daten in eine strukturierte Form zu überführen, um diese besser interpretieren zu können. Die RMP integriert derartige Techniken zur Verarbeitung unstrukturierter Daten. Da der Fokus dieser Dissertation nicht die Verarbeitung unstrukturierter Daten ist, wird an dieser Stelle auf weiterführende Literatur, etwa Kassner [Kas17] verwiesen.

Ein weiterer wichtiger Aspekt ist die Datenqualität. Eine geringe Datenqualität entsteht vor allem durch fehlerhafte, fehlende oder unvollständige Daten. Kiefer et al. [Kie16] diskutieren, welche Eigenschaften für die Messung der Qualität, insbesondere von unstrukturierten Daten, wichtig sind. Dabei sind vor allem die Relevanz, Interpretierbarkeit und Genauigkeit der Daten wichtige Eigenschaften, um die Datenqualität bestimmen zu können. Ein erster Ansatz, um die Datenqualität messbar zu machen, wird ebenfalls von Kiefer et al. beschrieben. Weitere Ansätze beschreibt Sebastian-Colem [Seb13].

In einem ersten Schritt können derartige Metriken in die RMP integriert werden. Dadurch kann gemessen werden, wie hoch die Qualität der vorliegenden Daten ist. Dabei können diese Metriken beispielsweise eine prozentuelle Angabe machen wie hoch die Datenqualität ist, wobei 100% eine sehr hohe Qualität und 0% eine sehr geringe bzw. keine Qualität ausdrückt. Auf Basis dieser Werte können Datenanbieter die Qualität ihrer, über die RMP bereitgestellten, Daten abfragen bzw. Maßnahmen ergreifen, um diese zu steigern. Durch eine erneute Berechnung dieser Metriken kann anschließend evaluiert werden, ob die Maßnahmen zur Steigerung der Datenqualität beigetragen haben.

Jedoch geht dies noch nicht weit genug. Wünschenswert wäre eine automatische Maßnahmenergreifung, um die Datenqualität zu steigern. Jedoch birgt die automatische Veränderung von Daten zur Erhöhung der Datenqualität auch diverse Risiken. Beispielsweise können leere Dateneinträge für manche Szenarien einen wichtigen Indikator darstellen, in anderen Sze-

narien jedoch störend sein. Automatisch zu entscheiden, ob leere Einträge gelöscht werden sollen birgt dass Risiko, dass Informationen verloren gehen. Auch die Erkennung von Rechtschreibfehlern bzw. deren automatische Verbesserung birgt die Gefahr, dass die Daten in falscher Weise verändert werden. Beispielsweise werden Abkürzungen oftmals als Rechtschreibfehler erkannt, obwohl diese wertvolle Informationen bieten können.

Aufgrund der Risiken bei der automatischen Änderung von Daten zur Steigerung der Datenqualität wurde dies nicht in der RMP umgesetzt. Stattdessen können Daten mittels dem DVM gefiltert und transformiert werden, um die Datenqualität zu erhöhen. Dabei nimmt der Modellierer eine zentrale Rolle ein und kann abhängig vom Anwendungsfall entscheiden, welche Maßnahmen zu einer tatsächlichen Steigerung der Datenqualität führen. Aus diesem Grund kann die RMP zwar durch die beschriebenen Metriken eine Möglichkeit bieten eine Tendenz zu erkennen, wie hoch die Qualität der Daten ist. Automatische Maßnahmen zur Verbesserung werden jedoch nicht getroffen.

7.4 Implementierung der Konzepte und Evaluation

Die beschriebenen Konzepte wurden zu großen Teilen prototypisch implementiert und mittels Messungen evaluiert. Dies wird in den folgenden Abschnitten beschrieben.

7.4.1 Konzeptimplementierung

Die Implementierung der RMP wurde unter Mithilfe der von mir betreuten studentischen Arbeit von Grumbein [Gru16] durchgeführt. Dabei wurden folgende Komponenten geschaffen: 1) eine Ontologie zum Speichern von Datenquellenbeschreibungen, 2) eine Registrierungskomponente, mittels der Datenquellen registriert werden können, und 3) eine Provisionierungskomponente, die Adapter provisioniert und Datenquellen somit anschließt.

Die *Ontologie* wurde mittels der Web-Ontology-Language (OWL) 2 formuliert. Der Zugriff auf die Ontologie wurde über eine REST-Schnittstelle

ermöglicht. Hierfür kommt das Ontologie-Framework Apache Jena[1] zum Einsatz. Dieses Framework bietet Methoden, um die Ontologie mittels der Querysprache SparQL zu durchsuchen. Somit wird es ermöglicht, Elemente der Ontologie abzufragen, zu erstellen, zu verändern, oder zu löschen. Zusätzlich wurde eine rudimentäre, PHP-basierte, Nutzeroberfläche zum Anzeigen des Ontologieinhalts implementiert.

Die *Registrierungskomponente* wurde vollständig in der Programmiersprache Java geschaffen und nutzt die REST-Schnittstelle der Ontologie, um Detailinformationen der Datenquellen, z.B. der zu provisionierende Adapter, abzufragen. Die Registrierungskomponente ruft die Provisionierungskomponente auf, die die Datenquelle anschließt.

Die *Provisionierunskomponente* wurde ebenfalls in der Programmiersprache Java geschrieben. Statt jedoch die Provisonierung mittels TOSCA durchzuführen, wurde im Rahmen der Implementierung SSH verwendet. Hierfür wird auf die Adapterplattform verbunden, der Adapter kopiert und gestartet. Eine derartige Vorgehensweise erfordert es jedoch, dass alle vom Adapter abhängigen Softwarekomponenten bereits vorinstalliert sind. Aufgrund dessen, dass es sich bei der Implementierung um eine Machbarkeitsuntersuchung handelt, wurde an dieser Stelle aus Gründen der Einfachheit auf TOSCA verzichtet. Eine Implementierung in TOSCA wurde von Képes durchgeführt und ist in [HBS+16] beschrieben.

7.4.2 Messungen des Prototyps

Basierend auf dem vorgestellten Prototyp wurden Laufzeitmessungen für die einzelnen Schritte der Methode (Abbildung 7.2) durchgeführt. Die Messungen wurden auf einem Rechner mit einem Core i5-3750K @3.4GHz Prozessor sowie 8 GB RAM durchgeführt. Die Messergebnisse sind in Tabelle 7.1 zu sehen. Es wurden 10 Messungen vorgenommen, um mögliche Ausreißer zu kompensieren. Die anzuschließenden Datenquellen sowie die Adapterplattform waren im selben Netzwerk verfügbar.

[1] https://jena.apache.org/

Tabelle 7.1: Laufzeit des Prototyps

#	Gesamtlaufzeit	Registrierung	Ontologietrav.	Adapterprov.
1	184,37 ms	1,64 ms	14,37 ms	165,62 ms
2	130,52 ms	1,74 ms	7,33 ms	120,97 ms
3	131,80 ms	1,52 ms	6,93 ms	122,99 ms
4	119,88 ms	2,61 ms	5,20 ms	111,68 ms
5	130,43 ms	2,16 ms	5,40 ms	122,50 ms
6	165,72 ms	1,47 ms	4,69 ms	159,14 ms
7	152,45 ms	1,75 ms	5,02 ms	145,32 ms
8	160,27 ms	2,12 ms	6,99 ms	150,60 ms
9	163,06 ms	1,82 ms	6,57 ms	154,27 ms
10	150,67 ms	2,29 ms	4,75 ms	143,23 ms
⌀	148,92 ms	1,91 ms	6,73 ms	139,63 ms

Hierbei wurden die Schritte Registrierung (Schritt 1), Ontologietraversierung (Schritt 2), und Adapterprovisionierung (Schritt 3) einzeln gemessen. In der ersten Spalte der Tabelle ist die Summe dieser Messungen in Millisekunden dargestellt. Dabei ist zu sehen, dass das Anschließen von Datenquellen in diesem Versuchsaufbau in unter 200ms möglich ist.

Betrachtet man die Laufzeiten der einzelnen Schritte im Detail, fällt auf, dass der Registrierungsschritt den kleinsten Anteil an der Gesamtlaufzeit einnimmt. Dieses Ergebnis ist erwartungsgemäß, da die Registrierung lediglich das Anstoßen der weiteren Schritte bzw. die Weiterreichung der Datenquelleninformationen übernimmt. Die Traversierung der Ontologie braucht bei der ersten Ausführung etwa doppelt so lange wie danach. Dies kommt daher, da bei erstmaliger Ausführung die Ontologie in den Hauptspeicher geladen wird und dort verbleibt. Anschließend kann bereits direkt auf die Ontologie zugegriffen werden. Die Traversierungsdauer hängt von der Größe der Ontologie ab. In dem hier gemessenen Beispiel ist die Ontologie sehr klein und enthält lediglich die Beschreibung von drei Datenquellen. Um deren Informationen auszulesen sind im Schnitt 6,73 ms notwendig. Der zeitintensivste Schritt ist die Provisionierung der Adapter. Die Provisionierung benötigt im Schnitt 143,23 ms und nimmt daher fast die gesamte Laufzeit

ein. Die verhältnismäßig hohe Laufzeit stammt fast ausschließlich von dem Aufbau einer SSH-Verbindung zur Adapterplattform. Diese SSH-Verbindung benötigte im Schnitt ca. 100 ms zum Aufbau der Verbindung. Das Übertragen und Starten des Adapters kann hingegen in kurzer Zeit durchgeführt werden (ca. 40 ms).

Die Messungen zeigen, dass die RMP Datenquellen effizient anschließen kann. Dies ist notwendig, da in dynamischen Umgebungen neu hinzugekommene Datenquellen so schnell wie möglich anzuschließen sind. In zeitkritischen Anwendungen sollte die Adapterprovisionierung über schnelle Kommunikationswege stattfinden, beispielsweise mittels UDP statt TCP.

7.5 Verwandte Arbeiten

Es existieren diverse Middleware-Lösungen, die eine ähnliche Funktionalität zum Anbieten oder Bereitstellen von Daten bieten. Darüber hinaus existieren Servicebusse sowie spezielle Datenverarbeitungsplattformen. Die verwandten Arbeiten werden in diesem Abschnitt beschrieben und abgegrenzt.

In der Vergangenheit wurden diverse Middleware-Plattformen entwickelt, die von konkreten Datenquellenzugriffen abstrahieren [BPM02; GAH+08; LWB08]. Jedoch bietet keine dieser Plattformen die Dynamik, die durch die RMP ermöglicht wird. Vor allem das vollautomatische, standardbasierte Anbinden von Datenquellen stellt ein Alleinstellungsmerkmal der RMP dar. Des Weiteren ist die RMP generisch, was bedeutet, dass sie jegliche Typen von Datenquellen unterstützt, vorausgesetzt, es existiert ein entsprechender Adapter, mit dem die Datenquellen angeschlossen werden können.

Ähnlich zu dieser Dissertartion verwenden Li et al. [LVCD13] TOSCA, um Datenquellen (insbesondere im Internet der Dinge) zu spezifizieren und automatisiert anzuschließen. Diese Vorgehensweise soll es erleichtern, darüberliegenden Applikationen Zugriff auf die entstehenden Daten zu gewähren. Jedoch ist hierfür die Erstellung von TOSCA-Modellen notwendig. In den Konzepten dieser Dissertation werden Datenquellen vollautomatisiert angebunden. Die TOSCA-Modelle können wie in Kapitel 6 beschrieben

vollautomatisch erstellt und vervollständigt werden.

Vögler et al. [VSID16] stellen LEONORE vor, ein skalierbares Framework, um Anwendungslogik (zum Beispiel Adapter) auf verschiedenartiger Infrastruktur zu provisionieren und zu betreiben. Um dies zu ermöglichen, müssen jedoch vorinstallierte Agenten auf der Zielinfrastruktur vorhanden sein. Diese Agenten registrieren Datenquellen mittels einer ID. Eine derartige Vorgehensweise kann als Push-Ansatz gesehen werden. Im Gegensatz dazu benötigt der von mir beschriebene Ansatz keinerlei vorinstallierte Software. Die Informationen, wie Datenquellen angeschlossen werden können, sind bereits im Datenverarbeitungsmodell enthalten und müssen nicht von Agenten zur Verfügung gestellt werden.

Der Forschungsbereich der *Datenintegration* nutzt Techniken wie Schema-Mapping oder Schema-Matching, um uniform auf heterogene, verteilte Daten zugreifen zu können [LN07]. Im Gegensatz dazu ist es nicht das Ziel der RMP, Daten zu integrieren, sondern lediglich die Datenquellen einzeln anzubinden. Eine gemeinsame Anfragesprache, mit der die Daten adressiert werden können, ist nicht vorgesehen. Die Integration der Daten findet vielmehr auf einer höheren Ebene durch die Modellierung des DVM statt.

Ishaq et al. [IHR+12] haben eine REST-basierte Schnittstelle geschaffen, um auf Datenquellen im Internet der Dinge, insbesondere auf Sensoren, zuzugreifen. Es wird jedoch davon ausgegangen, dass die Datenquellen bereits angebunden sind. Im Gegensatz dazu bindet die RMP die Datenquellen dynamisch an, was den Aufwand für die Datenbereitstellung stark vermindert. Die Datenbereitstellung über die REST-Schnittstelle wird von Ishaq et al. und dem hier vorliegenden Ansatz analog umgesetzt. Zusätzlich bietet die RMP auch eine push-basierte Datenbereitstellung mittels *Publish-Subscribe*.

Darüber hinaus wurden in der von mir betreuten Fachstudie "Analyse und Vergleich von IoT-Plattformen" [BKL16] verschiedene Middleware-Lösungen für das Internet der Dinge untersucht, da es besonders in dieser Domäne von hoher Wichtigkeit ist, Datenquellen (Sensoren) dynamisch anschließen zu können. Dabei lag der Fokus insbesondere darauf festzustellen, ob die untersuchten Systeme die Funktionalitäten der RMP ebenfalls unterstützen. Das Ergebnis dieser Untersuchung war, dass es zwar vereinzelt Ansätze

gibt, um Datenquellen dynamisch anzuschließen, dies jedoch begrenzt auf spezifische Infrastrukturen ist. Eine generische Lösung, wie sie die RMP ermöglicht, unterstützen diese Plattformen jedoch nicht.

7.6 Zusammenfassung

In diesem Beitrag wird die Ressourcen-Management-Plattform beschrieben, ein Konzept zur automatischen Anbindung und Abstraktion heterogener, verteilter Datenquellen. Der Beitrag teilt sich in: (1) automatisches Anbinden von Datenquellen, (2) effektive Datenzwischenspeicherung, um Ausfälle von Datenquellen zu kompensieren, (3) Datentransformation und Bereinigung zur Steigerung der Datenqualität und (4) eine Überwachung von Datenquellen, um auf deren Wegfallen schnell reagieren zu können.

Hierfür wurden Konzepte entworfen und gegenüber bestehenden Ansätzen abgegrenzt. Alleinstellungsmerkmale sind die automatische Anbindung mittels TOSCA sowie die Ontologien zur Datenquellenverwaltung.

Die Ressourcen-Management-Plattform dient als zentrale Komponente zur Datenbereitstellung für die DPM-Ausführung. Über Caching kann außerdem die Effizienz und über Datenbereinigung die Qualität der Datenverarbeitung erhöht werden.

Kapitel 8

DPM-Ausführung

In diesem Kapitel wird die DPM-Ausführung beschrieben. Des Weiteren werden zwei Verbesserungen beschrieben, mit der die Ausführung effizienter bzw. nutzerfreundlicher durchgeführt werden kann. Die Verbesserungen umfassen die verteilte Ausführung mittels Rechenclustern und die partielle Ausführung der Datenverarbeitung während der Modellierungszeit.

Die in diesem Kapitel vorgestellten Konzepte basieren auf [HBM17; Hir17]. Dabei stammen die in [Hir17] beschriebenen Konzepte vollständig von mir, wohingegen in [HBM17] zusätzlich Inhalte der Koautoren, bezüglich der Darstellung des momentanen Ausführungszustandes sowie der Zwischenergebnisse, eingeflossen sind.

8.1 Ausführung

Nachdem die Ausführungsumgebung automatisiert aufgesetzt wurde, können die Daten basierend auf dem transformierten DPM verarbeitet werden.

Wie in Kapitel 5 beschrieben, findet die Datenverarbeitung durch Services statt. Nach der Generierung des ausführbaren Workflowmodells für das DPM, dem Anschließen der Datenquellen an die RMP sowie dem automatischen

Provisionieren der Ausführungsumgebung, kann die Datenverarbeitung angestoßen werden. Die Ausführungsumgebung besteht sowohl aus selbst aufgesetzten Komponenten, als auch aus Komponenten, die von externen Anbietern als Dienste bereitgestellt werden. Die Orchestrierung dieser Dienste wird mittels dem transformierten Workflowmodell durchgeführt. Die Datenbereitstellung erfolgt wie in Kapitel 7 beschrieben über die RMP.

In Anbetracht dessen, dass das Workflowmodell bereits definiert, welche Services in welcher Reihenfolge zur Datenverarbeitung aufgerufen werden, genügt es, dieses Modell an eine passende Ausführungsumgebung (eine Workflowengine) zu übergeben. Die Workflowengine wurde entweder im Provisionierungsschritt automatisch aufgesetzt oder als Dienst bei einem externen Anbieter gebucht. Bei der Workflowausführung werden nacheinander die Services aufgerufen. Typischerweise bieten Workflowengines die Möglichkeit parallel verlaufende Pfade auch parallel in verschiedenen Threads auszuführen. Manche der Engines bieten jedoch lediglich Pseudoparallelismus, wobei nur ein einzelner Thread verwendet wird.

Existiert wie in Abschnitt 5.3.1 beschrieben eine Servicemiddleware, die die konkreten Serviceaufrufe abstrahiert, gehen alle Aufrufe des Workflows zuerst and diese Middleware und werden anschließend an die Services mit der entsprechenden Parametrisierung weitergeleitet. Auch die Ergebnisrückgabe erfolgt dann über die Middleware.

Bei der Ausführung des DPM werden zuerst die Services zur Datenextraktion aufgerufen. Die Services nutzen die Schnittstellen der RMP, um die Daten aus den Quellen zu extrahieren. Aufgrund dessen, dass die Daten durch die RMP bereits bereinigt werden können, bzw. die verschiedenartigen Formate vereinheitlicht werden, müssen die Extraktionsservices keine komplexen Datenverarbeitungsoperationen durchführen. Anschließend werden die extrahierten Daten an die im DPM definierten Services weitergereicht, die die modellierten Datenoperationen ausführen (Filter, Analyse, Speicherung, Visualisierung, etc.).

Untersuchungen im Rahmen dieser Dissertation haben ergeben, dass viele Workflowengines (bspw. Apache ODE) Probleme damit haben, große Datenmengen zwischen den Serviceaufrufen zu transferieren. Daher kann bei der

Implementierung ein Zwischenspeicher eingesetzt werden, der verarbeitete Daten abspeichert und lediglich einen Verweis auf die zwischengespeicherten Daten an den nächsten Service zur Weiterverarbeitung übergibt.

Nach der Ausführung des Workflows wurden die Daten, wie im DPM beschrieben, verarbeitet. Die Transformation des DVM bzw. DVM$^+$ in das DPM, das Aufsetzen der Ausführungsumgebung und der Ablauf der Datenverarbeitung werden vor dem Modellierer verborgen. Lediglich das Ergebnis wird angezeigt. Durch die somit erreichte Vollautomatisierung kann eine menschliche Interaktion vermieden und damit auch eine wichtige Fehlerquelle ausgeschlossen werden. Darüber hinaus hilft die Vollautomatisierung besonders Domänennutzern, die eine auf ihre Bedürfnisse zugeschnittene Datenverarbeitung zur Verfügung gestellt bekommen, ohne technische Details kennen zu müssen.

8.2 Maßnahmen für eine verbesserte Ausführung

In diesem Abschnitt werden zwei Konzepte vorgestellt, die die Ausführung der Datenverarbeitung verbessern. Die Konzepte umfassen die Verteilung der Datenverarbeitung und die Teilausführung während der Modellierungszeit.

8.2.1 Verteilung der Datenverarbeitung

Eine Möglichkeit, Daten effizient zu verarbeiten, ist es, diese auf mehrere Rechenknoten zu verteilen und die Datenverarbeitung parallel auszuführen. Diese Vorgehensweise bietet sich insbesondere bei großen Datenmengen an. Bei kleineren Datenmengen kann der durch die Verteilung entstehende Kommunikationsoverhead von Nachteil sein bzw. sogar zu Effizienzeinbusen führen. Viel sinnvoller ist es in diesem Fall, die Daten direkt im Service, also im Hauptspeicher des jeweiligen Hosts, zu verarbeiten. Aus diesem Grund ist eine pauschale Verteilung aller Daten, auch aus Kostengründen, nicht sinnvoll. Vielmehr muss eine intelligente, individuelle Entscheidung getroffen werden, welche Datenverarbeitungsschritte verteilt werden sollten und welche nicht. Diese Entscheidung ermöglicht eine optimale Datenverar-

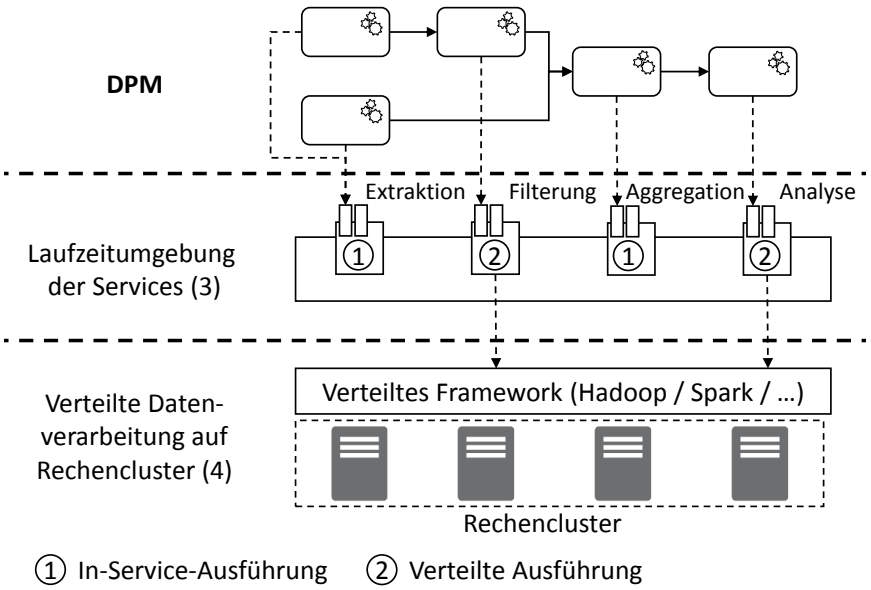

Abbildung 8.1: Konzepterweiterung zur Verteilung der Datenverarbeitung (aus [Hir17])

beitung, sowohl bzgl. Effizienz als auch bzgl. Kosten. Die Entscheidung, ob Daten verteilt verarbeitet werden sollten oder nicht, hängt außerdem von den auf Modellierungsebene festgelegten Anforderungen des Modellierers im DVM$^+$ zum Beispiel hinsichtlich Kosten oder Effizienz ab.

Um die Entscheidung treffen zu können, ob Daten im Service oder verteilt verarbeitet werden, müssen außerdem viele Informationen über die Daten selbst bekannt sein. Diese Informationen betriffen beispielsweise die Datenmenge und Komplexität sowie Wissen über die Art der Datenverarbeitung, beispielsweise deren Eignung zur Verteilung. Dieses Wissen ist bei der Modellierung des DVM vorhanden und sollte an dieser Stelle mittels den beschriebenen Anforderungen an die Knoten des DVM$^+$ annotiert oder automatisch durch Metriken berechnet werden.

Um eine Verteilung generell zu ermöglichen, wurde in einem ersten Schritt

die in Kapitel 3 beschriebene Schichtenarchitektur um eine weitere Ausführungsschicht erweitert. Die Erweiterung ist in Abbildung 8.1 zu sehen. Bisher gab es neben der auf oberster Ebene angesiedelten Modellierungsschicht die Ausführungsschicht für das in das DPM transformierte DVM^+, das entsprechende Services zur Datenverarbeitung aufruft. Nun kommt eine weitere Ausführungsschicht hinzu, die eine verteilte Datenverarbeitung berücksichtigt. Hierfür können Konzepte wie Map-Reduce und Technologien bzw. Frameworks wie Apache Spark zum Einsatz kommen.

Es ist nicht immer sinnvoll die Datenverarbeitung verteilt auszuführen. Vor allem, wenn kleine Datenmengen verarbeitet werden, überwiegt der Overhead bei einer Verteilung zu stark. Aus diesem Grund habe ich in [Hir17] ein Konzept vorgestellt, mit dem eine sinnvolle Entscheidung über die Verteilung von Daten erreicht werden kann. Als Grundlage dient die in Abbildung 8.1 dargestellte erweiterte Architektur.

Ob Daten verteilt verarbeitet werden sollten, hängt vor allem von deren Größe, aber auch deren Komplexität sowie von der Komplexität der jeweiligen Datenverarbeitungsoperation ab. Wie im Beispiel aus Abbildung 8.1 zu sehen ist, gibt es Services, die die Daten im Hauptspeicher der jeweiligen Laufzeitumgebung verarbeiten (1) und Services, die die Daten verteilt, beispielsweise mittels Apache Spark, verarbeiten (2). Ich unterscheide hierbei zwischen *In-Service*-Datenverarbeitung und *verteilter* Datenverarbeitung. Dabei wird die In-Service-Datenverarbeitung durch Serviceimplementierungen realisiert, die auf einem einzelnen Rechner in dessen Hauptspeicher ausgeführt werden. Der Hauptspeicher ist jedoch begrenzt und kann nicht skaliert werden. Im Gegensatz hierzu wird die verteilte Datenverarbeitung durch Services realisiert, die Daten parallel auf einem Rechencluster verarbeiten.

Um zu entscheiden, welche dieser Servicetypen für bestimmte Daten ausgewählt bzw. ausgeführt werden sollten, werden die Services in einem ersten Schritt wie in Kapitel 5 beschrieben mit Policys annotiert (z.B. mittels Servicebeschreibungen wie WS-Policys), die definieren, für welche Art von Daten und für welche Datenmenge die jeweiligen Services geeignet sind. Bei geringen Datenmengen sowie einer geringen Komplexität kommen Services in Frage, die Daten ohne die Verwendung von z.B. Map-Reduce verarbeiten.

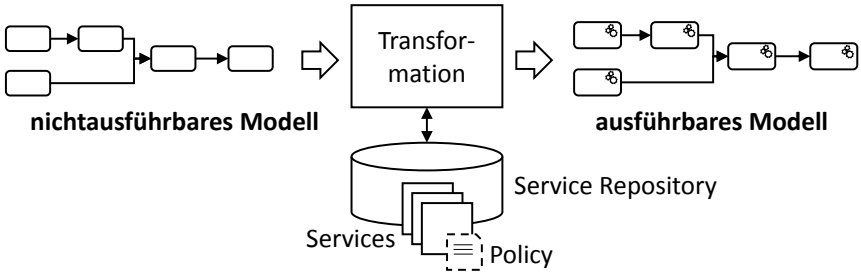

Abbildung 8.2: Policy-basierte Auswahl der Services (aus [Hir17])

Die Entscheidung, welche Services letztendlich für die Datenverarbeitung genutzt werden, wird bei der Transformation des DVM^+ in das ausführbare DPM getroffen. Dies ist in Abbildung 8.2 dargestellt. Dabei werden die Art der Daten sowie die Datenmenge mit den Policys geeigneter Services abgeglichen. Erfüllt eine Service-Policy alle Anforderungen wird ein entsprechender Serviceaufruf in die ausführbare Repräsentation eingefügt. Erfüllen mehrere Services eine Policy, sollte die Auswahl entweder zufällig geschehen, oder es müssen weitere Faktoren, wie z.B. Kosten, Laufzeitkomplexität, benötigte Rechenressourcen (z.B. Größe der Cluster) hinzu genommen werden. Die Transformation wurde in Kapitel 5 im Detail beschrieben.

8.2.1.1 Technologien zur verteilten Datenverarbeitung

Hadoop ist eine bekannte Plattform für die Ausführung von Map-Reduce, welches auf dem Hadoop-File-System (HDFS), einem verteilten Dateisystem, arbeitet. Basierend auf dem HDFS ermöglicht Hadoop eine auf mehreren Rechenclustern verteilte Verarbeitung großer Datenmengen mittels Map-Reduce. In den letzten Jahren wurde Hadoop zum de-facto Standard für die Implementierung von Map-Reduce. Für eine erste Implementierung des vorgestellten Ansatzes wurde Hadoop verwendet. Seit einiger Zeit sind jedoch moderne, effizientere Ansätze auf dem Vormarsch. Zu diesen Ansätzen gehören Apache Spark [Spa15] und Apache Flink [CKE+15]. Apache Spark ermöglicht eine deutliche Effizienzsteigerung im Vergleich zu Hadoop [GA15],

da die Daten im Hauptspeicher verarbeitet werden und nicht nach jedem Verarbeitungsschritt in das HDFS geschrieben werden müssen. Des Weiteren bietet Apache Spark eine Vielzahl weiterer Features, die insbesondere auch Grundfunktionen der Datenverarbeitung (Filterung, Aggregation) beinhalten. Außerdem ermöglicht Spark eine Unterstützung von strombasierter Datenverarbeitung, was eine höhere Abdeckung verschiedenartiger Szenarien ermöglicht. Eine weitere Plattform für effiziente Datenverarbeitung ist Apache Flink. Apache Flink ist auf strombasierte Datenverarbeitung fokussiert und bietet eine ähnliche Funktionalität wie Spark. Flink hat somit wie Spark einen Effizienzvorteil gegenüber Hadoop. Ein umfassender Vergleich von Spark und Flink ist in [MCAP16] beschrieben.

Zusammengefasst ergibt diese Diskussion, dass es sinnvoll ist, abhängig vom Anwendungsfall, verschiedene Frameworks und Datenverarbeitungstechniken, z.B. strombasiert, zu verwenden.

Die Abstraktion der eigentlichen Datenverarbeitung durch Services erlaubt es, verschiedene Serviceimplementierungen für dieselben Datenoperationen anzubieten. Diese Operationen können dann beispielsweise mit Hadoop, Spark, Flink, oder anderen Plattformen ausgeführt werden.

8.2.1.2 Vergleich von In-Service- und verteilter Datenverarbeitung

Es wurden Messungen durchgeführt, um zu untersuchen, welche Auswirkungen die Verarbeitung von Daten In-Service bzw. in verteilter Weise auf die Laufzeit haben. Des Weiteren wurde untersucht, welche Auswirkungen die Clustergröße bei der verteilten Verarbeitung von Daten hat, insbesondere ab welchem Punkt es nicht mehr lohnt, neue Rechenknoten zur Berechnung hinzu zu nehmen. Die verteilte Datenverarbeitung wurde mittels Apache Spark durchgeführt. Die ausgewählten Datenverarbeitungsoperationen waren Extraktion und Filter, da diese die (meiner Erfahrung nach) am meisten verwendeten Operationen der Datenverarbeitung sind. Dabei werden Daten aus einer CSV-Datei eingelesen und über die RMP in ein uniformes Datenmodell überführt. Die Ergebnisse dieser Messungen beziehen sich auf die von mir betreute Masterarbeit von Sarangi [Sar17]. Dabei standen sowohl bei der

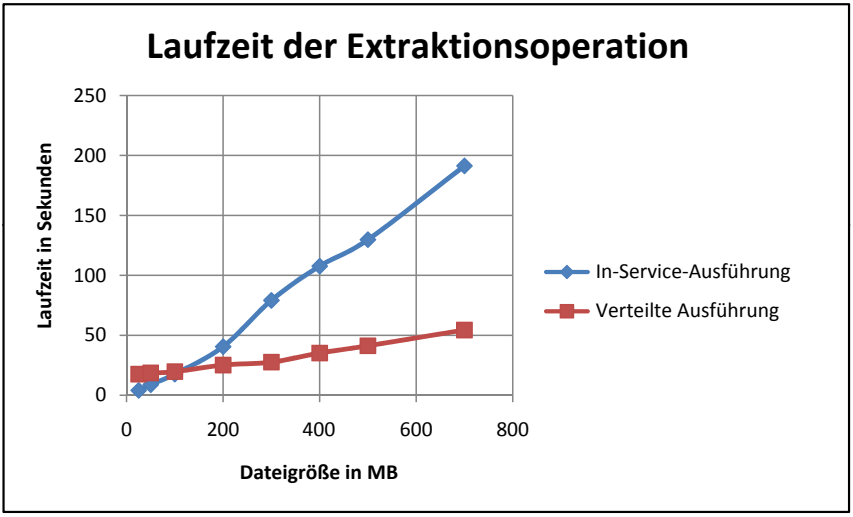

Abbildung 8.3: Laufzeitmessungen der Extraktionsoperation (aus [Sar17])

In-Service als auch bei der verteilten Datenverarbeitung genug Arbeitsspeicher zur Verfügung. In der Praxis führt die Limitierung des Arbeitsspeicher jedoch oftmals dazu, dass große Datenmengen nicht In-Service verarbeitet werden können und somit pauschal verteilt werden müssen.

Abbildung 8.3 zeigt die Laufzeit der Extraktionsoperation, einerseits verteilt mittels Map-Reduce (rot) und andererseits lokal mittels In-Service-Verarbeitung (blau) für verschiedene Datengrößen von 10 MB - 700 MB. Die verteilte Berechnung wurde in einem ersten Schritt auf einem kleinen Spark-Cluster mit einem Master und zwei Worker-Knoten durchgeführt. Eine geringe Anzahl an Rechenknoten ist vollkommen ausreichend, um zu zeigen, dass durch eine Verteilung der Datenverarbeitung prinzipiell eine hohe Effizienzsteigerung erreicht werden kann. Wie sich die Veränderung der Clustergrößen auf die Effizienz auswirkt, wird nachfolgend untersucht. Wie in Abbildung 8.3 zu sehen ist, lohnt sich die Verteilung der Daten im Falle einer Extraktionoperation erst ab 100 MB. Danach sind starke Effizienzsteigerungen auch bei einer geringen Clustergröße zu erreichen. Bei

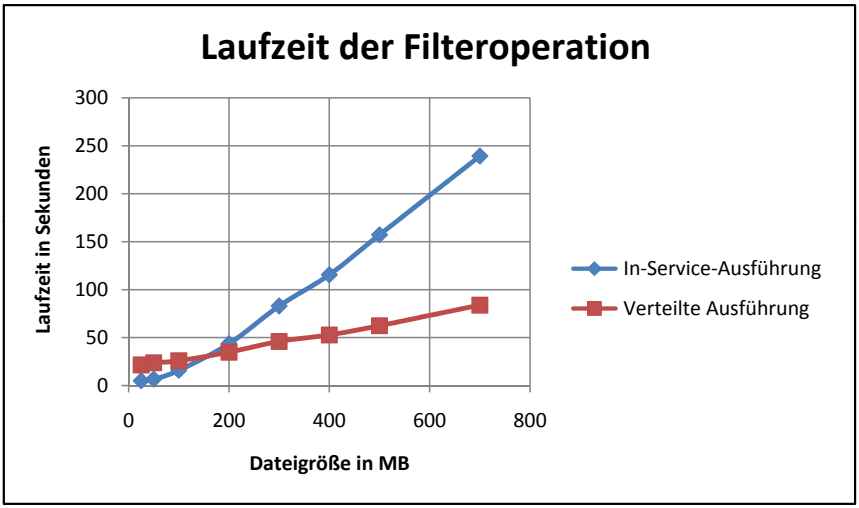

Abbildung 8.4: Laufzeitmessungen der Filteroperation (aus [Sar17])

einer Datenmenge von 700 MB unterscheidet sich die Laufzeit beider Ausführungsvarianten sogar um über zwei Minuten.

Abbildung 8.4 zeigt die Messung für die Filteroperation, einmal verteilt auf einem Cluster mit einem Master und zwei Workerknoten und einmal In-Service. Hierbei ist eine ähnliches Ergebnis wie bei der Extraktionsoperation zu sehen. Dabei lohnt sich ab einer Datengröße von 150 MB eine Verteilung der Daten. Bei Datenmengen, die darunter liegen, sollten diese In-Service verarbeitet werden, da der Kommunikationsoverhead der Verteilung höher als der Effizienzgewinn ist. Bei einer weiteren Erhöhung der Datenmenge ist auch hier eine große Effizienzsteigerung durch eine Verteilung der Datenverarbeitung zu erkennen. Die Datenmenge beträgt bei 700 MB sogar ca. drei Minuten.

Nachdem eine deutliche Verbesserung durch die vorangegangenen Messungen erkannt wurde, wird im nächsten Schritt untersucht, wie sich eine Steigerung der Clustergröße auf die Effizienz der Datenverarbeitung auswirkt. Hierfür wurde im ersten Schritt eine Eingabedatei mit 200 MB ausge-

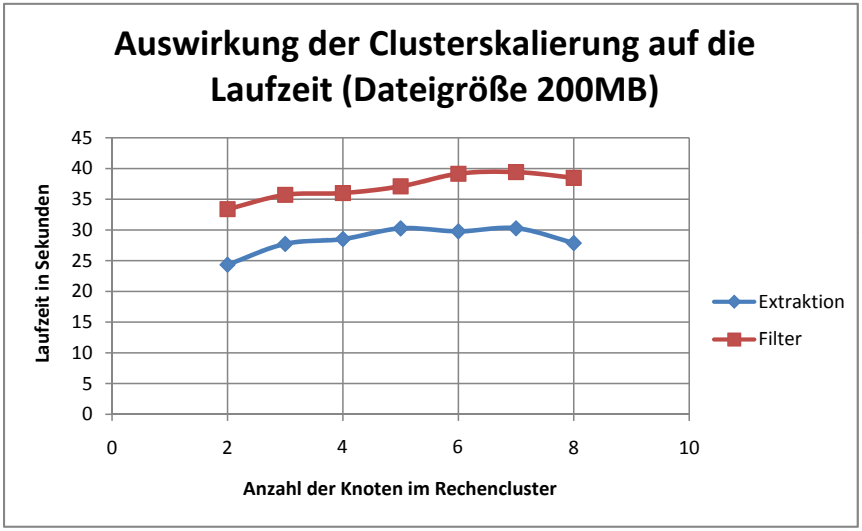

Abbildung 8.5: Auswirkungen Clusterskalierung für 200 MB Daten (aus [Sar17])

wählt und die beiden Datenoperationen Extraktion und Filter angewendet. Für Daten mit geringerer Größe ist, wie im vorigen Abschnitt gezeigt, eine Datenverarbeitung innerhalb des Services sinnvoll. Abbildung 8.5 zeigt die Ergebnisse der durchgeführten Laufzeitmessungen für verschiedene Clustergrößen mit zwei bis acht Rechenknoten. Hierbei ist zu sehen, dass die Laufzeit bei einer Steigerung der Clustergröße für beide Datenoperationen zunimmt. Die Laufzeitsteigerung ist durch den dadurch entstehenden Kommunikationsoverhead zu erklären. Die Messungen zeigen, dass für eine verhältnismäßig geringe Datenmenge eine Erhöhung der Clustergröße keine Vorteile bringt, sondern, im Gegenteil, das Ergebnis sogar verschlechtert.

Auf Basis dieser Ergebnisse wurde anschließend die Dateigröße weiter gesteigert, um zu erkennen, ab wann eine Erhöhung der Clustergröße sinnvoll ist. Abbildung 8.6 zeigt die Messergebnisse für eine Datengröße von 300 MB. Hierbei ist zu sehen, dass eine Steigerung der Clustergröße für die Extraktionsoperation keinen positiven Effekt hat und die Laufzeit sogar

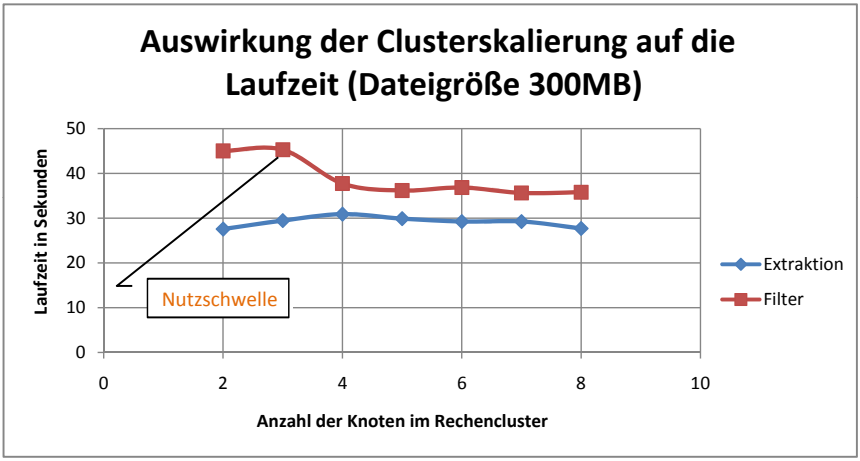

Abbildung 8.6: Auswirkungen Clusterskalierung für 300 MB Daten (aus [Sar17])

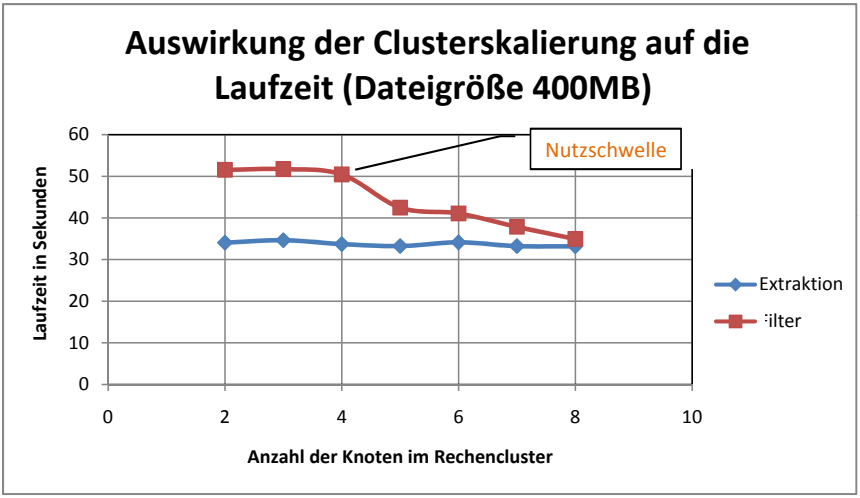

Abbildung 8.7: Auswirkungen Clusterskalierung für 400 MB Daten (aus [Sar17])

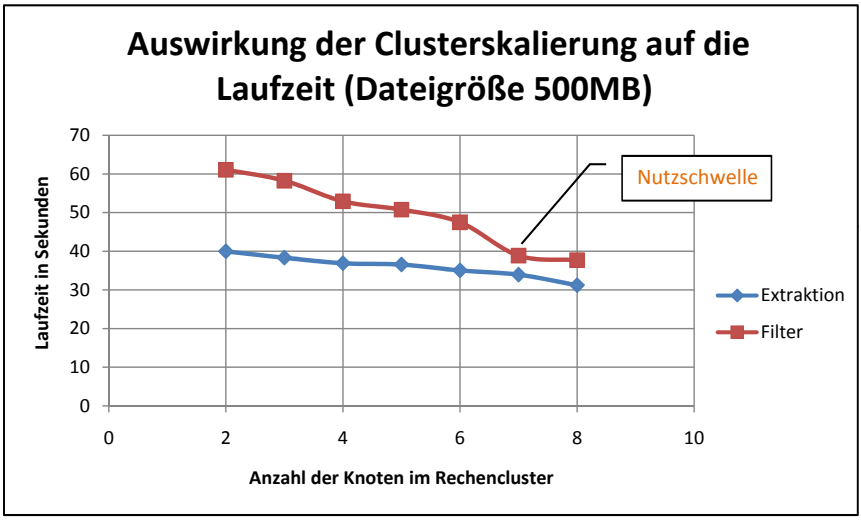

Abbildung 8.8: Auswirkungen Clusterskalierung für 500 MB Daten (aus [Sar17])

verschlechtert, analog zur Durchführung mit einer Datengröße von 200 MB. Für die Filteroperation kann hingegen eine Performanzsteigerung durch eine Erhöhung der Clustergröße erreicht werden. In Abbildung 8.6 ist zu sehen, dass eine Steigerung auf eine Clustergröße mit vier Rechenknoten bereits zu einer deutlichen Verbesserung führt. Eine weitere Steigerung der Clustergröße hat jedoch keinen weiteren positiven Effekt. Aus Kostengründen sollte die Clustergröße daher nicht weiter gesteigert werden. In diesem Fall ist eine Größe von vier optimal.

Die Auswirkungen der Clusterskalierung bei einer Erhöhung der Datengröße auf 400 MB ist in Abbildung 8.7 zu sehen. Hierbei ist festzustellen, dass eine Erhöhung der Clustergröße auf fünf Knoten zu einer deutlichen Verbesserung führt, eine weitere Erhöhung jedoch nicht. Dadurch liegt die Vermutung nahe, dass für die genannten Operationen, pro Erhöhung der Datengröße um 100 MB, ein weiterer Clusterknoten hinzugefügt werden sollte. Jedoch gilt dies nur für die Filteroperation. Bei der Extraktion kann

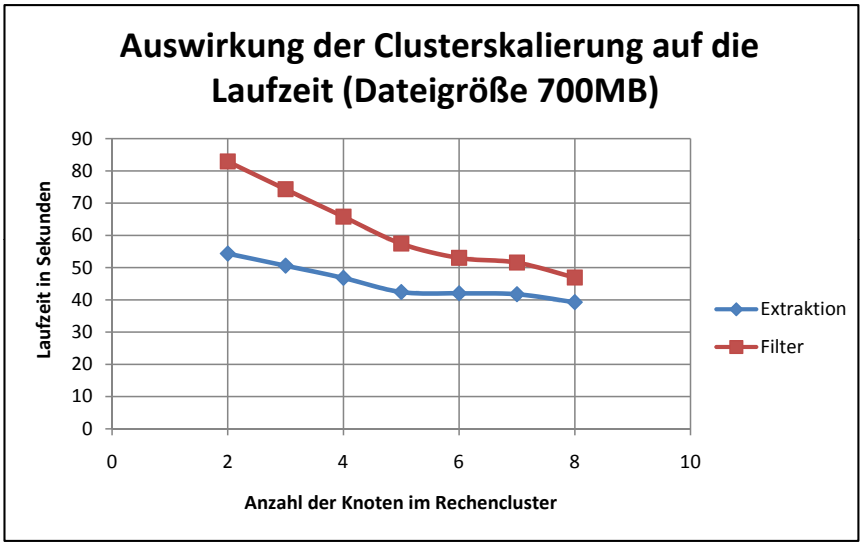

Abbildung 8.9: Auswirkungen Clusterskalierung für 700 MB Daten (aus [Sar17])

kein Effizienzgewinn festgestellt werden.

Um diese These zu überprüfen, wurde die Datengröße weiter erhöht. Abbildung 8.8 zeigt die Messungen für eine Datengröße von 500 MB. Dabei ist zu sehen, dass eine deutliche Effizienzverbesserung bei einer Clustergröße von sieben zu erwarten ist. Dies bedeutet wiederum, dass hierbei eine Erhöhung um zwei Rechenknoten notwendig ist und nicht, wie erwartet, um einen Knoten. Aufgrund dessen, dass bei einer Erhöhung auf 700 MB (siehe Abbildung 8.9) kein eindeutiger Trend erkennbar war, wurde die Datengröße weiter auf zwei GB erhöht (siehe Abbildung 8.10). Es ist zu sehen, dass eine Clustergröße von fünf optimal ist. Generell liegt die Vermutung nahe, dass eine Clustergröße über fünf Rechenknoten für Daten von 400 MB bis zu fünf GB nicht sinnvoll ist und vier bis fünf Knoten eine optimale Größe darstellen.

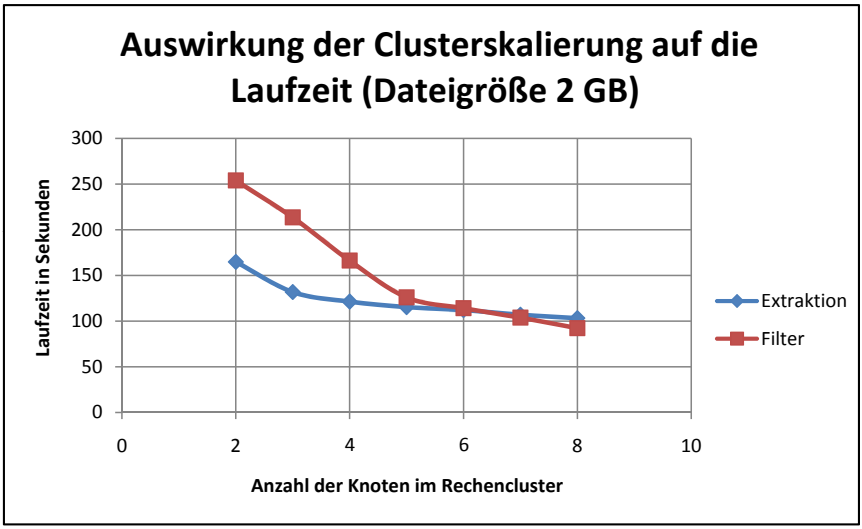

Abbildung 8.10: Auswirkungen Clusterskalierung für 2 GB Daten (aus [Sar17])

8.2.1.3 Zusammenfassung und Ausblick

In diesem Abschnitt wird beschrieben, wie sich eine Verteilung der Datenverarbeitung auf die Effizienz bei der Ausführung auswirkt. Insbesondere wurde untersucht, ob sich eine generelle Verteilung lohnt und, wenn nicht, ab welcher Datengröße eine Verteilung durchgeführt werden sollte. Hierfür wurden diverse Messungen basierend auf den am häufigsten modellierten Datenverarbeitungsoperationen Filterung und Extraktion durchgeführt. Die Messergebnisse sind wie erwartet ausgefallen. Das Ergebnis ist, dass durch eine Verteilung die Effizienz besonders bei großen Datenmengen stark erhöht werden kann. Bei kleineren Datenmengen ist der Overhead bei der Datenverarbeitung jedoch zu groß. Das bedeutet, dass es zwar sinnvoll ist, Daten verteilt zu verarbeiten, eine pauschale Verteilung jedoch auch negative Auswirkungen auf die Effizienz der Datenverarbeitung haben kann. Daher sollte dies für die Art der zu verarbeitenden Daten bzw. die verwendete Datenverarbeitungsoperation einzeln abgewägt werden.

Die in diesem Abschnitt beschriebenen Konzepte liefern einen ersten Ansatz, um in Zukunft automatisch entscheiden zu können welche Operationen im DPM verteilt ausgeführt werden sollten und welche nicht, um die bestmögliche Effizienz zu erreichen. Momentan muss bei der Modellierung des DVM$^+$ annotiert werden, ob eine enthaltene Operation verteilt ausgeführt werden kann. Das Ziel ist es, diese Entscheidung zukünftig vollautomatisch zu treffen, um den Modellierer zu entlasten. Der Einfluss der RMP, die die Daten für die Verarbeitung zur Verfügung war nicht Teil der Messungen. Eine quantitative Evaluation der RMP befindet sich in Kapitel 7.

8.2.2 Teilausführung des DVM zur Modellierungszeit

Eine weitere Verbesserung kann durch eine Teilausführung des DVM bzw. DVM$^+$ zur Modellierungszeit geschehen. Hierbei steht die Nutzbarkeit des Gesamtansatzes durch Domänennutzer im Vordergrund. Der folgende Abschnitt basiert auf [HBM17] sowie auf einer von mir betreuten studentischen Masterarbeit [Neu17], die die Konzepte implementiert hat.

Das in den vorigen Kapiteln vorgestellte Konzept zur Modellierung, Transformation und Ausführung des DVM sieht es vor, dass das Modell, nach dessen vollständiger Modellierung, als Ganzes transformiert und ausgeführt wird. Eine Ausführung als Ganzes führt einerseits dazu, dass die Ausführung, insbesondere bei großen Datenmengen, sehr lange dauern kann. Andererseits führt dies dazu, dass es bisher keine Möglichkeit gibt, Zwischenergebnisse anzuzeigen, wodurch Fehlererkennung bzw. nicht optimale Parametrisierungen frühzeitig im Modell erkannt werden können.

Abbildung 8.11 zeigt eine Gegenüberstellung der beiden Konzepte zur vollständigen und partiellen Ausführung. Auf der linken Seite ist zu sehen, dass das DVM bzw. DVM$^+$ bisher als Ganzes an die Transformationsschicht übergeben wurde, die dieses Modell in eine ausführbare Repräsentation überführt, die anschließend von einer passenden Umgebung vollständig ausgeführt wird. Auf der rechten Seite ist der neue Ansatz der partiellen Ausführung zu sehen. Dabei wird nur ein Teil des DVM transformiert und ausgeführt. Zwischenergebnisse werden in einem der Architektur neu hinzu-

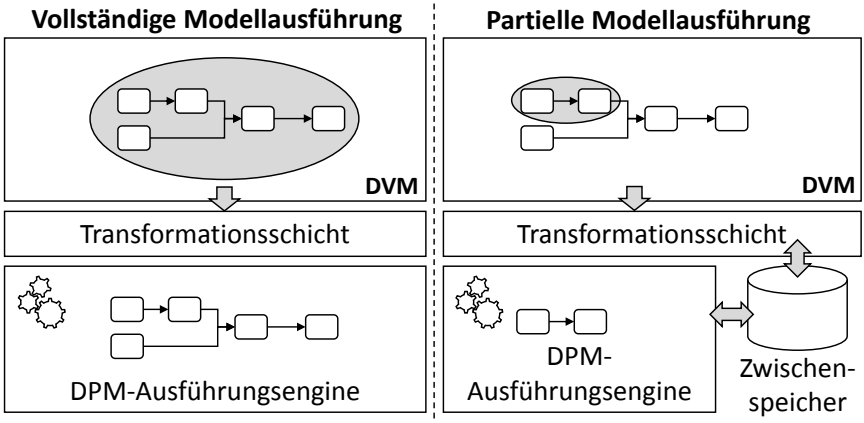

Abbildung 8.11: Bisherige, vollständige Ausführung der Datenverarbeitung (links) im Vergleich mit dem neuen optimierten Ansatz zur partiellen Ausführung (rechts) (basierend auf [HBM17])

gefügten Speicher persistiert, um als Grundlage für die Ausführung weiterer Teile des Modells zu dienen. Die partielle Ausführung geschieht während der Modellierungszeit. Das heißt, dass der Modellierer einerseits die Möglichkeit hat, Zwischenergebnisse einzusehen und andererseits, die Ausführung "gefühlt" schneller wird, da Teile des Modells während der Modellierungszeit bereits im Hintergrund ausgeführt werden. Der Modellierer muss die partielle Ausführung nicht manuell anstoßen, dies geschieht implizit bei Änderungen im Modell.

8.2.2.1 Verwandte Arbeiten

In diesem Abschnitt werden verwandte Arbeiten zur partiellen Ausführung von Modellen während der Modellierungszeit beschrieben. Die vorgestellten Konzepte stützen sich auf die Arbeiten von Altintas et al. [ABJ+04; ABJ06], sowie von Sonntag et al. [SK11; SK12; SK13].

Smart-Workflow-Re-runs
Im Jahr 2006 führten Altintas et al. [ABJ06] und Ludäscher et al. [LAB+06] sogenannte *Smart-Re-Runs* von Workflows ein, die in dem Workflow-System *Kepler* [ABJ+04] ausgeführt werden, eine spezielle Ausführungsumgebung für wissenschaftliche Workflows. Smart re-runs beschreiben eine wiederholte Ausführung von Workflows, bei denen lediglich einzelne Parameter, Operationen oder Rollen geändert wurden. Während eines solchen Smart-Re-Run wurden Teile des Workflows bereits ausgeführt und müssen nicht noch einmal ausgeführt werden. Die Ergebnisse vorheriger Ausführungen können aus Logdateien entnommen werden. Sobald Daten einer vorherigen Ausführung verfügbar sind, werden diese automatisch den geänderten Teilen des Workflows zur Verfügung gestellt, sodass sie, basierend auf den Vorergebnissen, ausgeführt werden können. Abhängigkeiten zwischen den veränderten und unveränderten Workflowteilen führen dazu, dass auch unveränderte Teile erneut ausgeführt werden müssen.

Die in diesem Abschnitt vorgestellten Konzepte basieren teilweise auf der Arbeit von Altintas et al. und Ludäscher et al., insbesondere bezüglich der wiederholten Ausführung von Workflows basierend auf vorigen Zwischenergebnissen. Im Gegensatz zu deren Arbeiten, fokussiere ich mich nicht auf wissenschaftliche Workflows, sondern stelle generische Konzepte für verschiedenartige Workflows vor mit einem Fokus auf Datenströmen bzw. große Datenmengen. Im Gegensatz dazu zielt das Konzept der Smart-Workflow-Re-runs auf die effiziente Variation von Parametern und Operationen eines wissenschaftlichen Workflows (engl. scientific workflow).

Des Weiteren werden durch die Modellierungsanforderungen und automatische Transformation verschiedenartige Workflowalternativen unterstützt. Dies ist der größte Unterschied zu den Konzepten von Altintas et al. Des Weiteren müssen in den Konzepten von Altintas et al. und Ludäscher et al. die Smart-Re-Runs manuell angestoßen werden. In unserem Ansatz wird die partielle Ausführung implizit während der Modellierung angestoßen, um eine explizite Entscheidung hierüber dem Modellierer abzunehmen.

Model-as-you-go Workflows

Sonntag und Karastoyanova [SK11; SK12; SK13] stellen das Konzept *Model-as-you-go* für wissenschaftliche Workflows vor, um eine wiederholte Ausführung, u.a. während der Modellierungszeit, zu ermöglichen. Die Konzepte wurden auf die Workflowsprache BPEL angewandt. Der Model-as-you-go-Ansatz ermöglicht das partielle Zurücksetzen eines Workflows, damit dieser ab einer bestimmten Workflowaktivität erneut ausgeführt werden kann. Dabei steht vor allem die Unterstützung des Modellierers im Vordergrund.

Ein weiterer Ansatz, den Sonntag und Karastoyanova vorstellen, ist die wiederholte Ausführung einer Workflowinstanz, zum Beispiel im Fehlerfall. Diese Instanz wird hierbei auf den Stand einer vorherigen Ausführung gesetzt, wobei sogenannte *Audit-Trails* zum Einsatz kommen. Dieser Stand wird anschließend für eine fortgesetzte Ausführung verwendet.

Neben den bereits erwähnten Arbeiten von Altintas et al., bauen die Konzepte dieses Abschnittes auch auf den Arbeiten von Sonntag und Karastoyanova auf. Im Gegensatz zu deren Arbeit, fokussiert sich das in diesem Abschnitt beschriebene Konzept nicht auf eine spezifische Workflowsprache, sondern auf das abstrakte DVM. Dieses Modell kann auf verschiedene ausführbare Repräsentationen bzw. DPMs abgebildet werden, darunter unter anderem BPEL, BPMN und die leichtgewichtige Workflowumgebung Node-RED. Das bedeutet, dass das DVM generisch ist und für verschiedene Workflowtechnologien eingesetzt werden kann. In Folge dessen, können auch die Konzepte zur partiellen Ausführung für verschiedenartige Ausführungsumgebungen angewandt werden. Die Konzepte von Model-as-you-go-Workflows sind zwar prinzipiell auch generisch, jedoch beschreiben Sonntag und Karastoyanova in ihrer Arbeit lediglich, wie deren Konzepte auf die Workflowsprache BPEL angewendet werden können. Darüber hinaus fokussiere ich mich, im Gegensatz zu den Model-as-you-go-Kontrollflussgraphen, auf das DVM, das einen Datenfluss beschreibt. Des Weiteren erlaubt mein Ansatz das implizite Anstoßen der partiellen Ausführung. Sonntag und Karastoyanova fordern hingegen ein explizites Anstoßen.

Weitere Verwandte Arbeiten
Aghaee und Pautasso [AP13] stellen Herausforderungen und Möglichkeiten der *Live-Programmierung* im Bereich Mashups vor. Die erarbeiteten Herausforderungen decken sich teilweise mit den Herausforderungen der Konzepte dieses Beitrags. Dabei definieren Aghaee und Pautasso den Begriff der Live-Programmierung, bei der Code während dessen Erstellung ausgeführt wird, und nennen einige Voraussetzungen, mit denen umgegangen werden muss, um derartige Ansätze zu ermöglichen. Diese Herausforderungen gelten auch für meine Konzepte zur partiellen Ausführung. Die Herausforderungen umfassen: (i) Minimierung der Antwortzeiten, (ii) Umgang mit Limitierungen bei Serviceaufrufen, und (iii) die optimale Ausnutzung der Bildschirmgröße. Vor allem die ersten beiden Voraussetzungen sind relevant für die Konzepte der partiellen Ausführung.

Pautasso und Alonso [PA05] stellen *JOpera* vor, eine visuelle Kompositionssprache. In deren Arbeit versuchen sie mit den Herausforderungen bzgl. Abhängigkeiten, die bei der Kombination von Daten- und Kontrollfluss entstehen, umzugehen. Dabei können, ähnlich zu den Konzepten dieser Dissertation, Datenflussgraphen modelliert werden, die Serviceaufrufe zur Datenextraktion und Verarbeitung repräsentieren. Dieses Datenflussmodell wird dann mit dem zugehörigen Kontrollflussgraphen synchronisiert. Das JOpera Flussmodell ist sehr ähnlich zu dem Datenverarbeitungsmodell dieser Dissertation. JOpera unterstützt das Anzeigen von Zwischenergebnissen [PA03], jedoch keine partielle Ausführung des Modells.

8.2.2.2 Herausforderungen und Anforderungen an das Konzept zur partiellen Ausführung des DPM

Um die in den vorigen Abschnitten beschriebenen Ziele der partiellen Ausführung zu erreichen wurden verschiedene Herausforderungen identifiziert, mit denen umgegangen werden muss. Diese umfassen:

- **Aktuelle Daten:** Das Ziel ist es, möglichst immer auf aktuellen Daten zu arbeiten. Werden jedoch Zwischenergebnisse gespeichert, wie es vorgesehen ist, können diese Daten veralten, was wiederum zu irre-

führenden oder falschen Ergebnissen führen kann, die somit wertlos werden. Wie schnell Daten veralten, hängt sehr stark von der Art der Daten ab, insbesondere mit welcher Häufigkeit sich die zugrundeliegende Datenmenge ändert. Beispielsweise ändern sich Sensordaten in der Regel häufig, wohingegen historische Wetterdaten gleich bleiben. Der Umgang mit veralteten Daten stellt eine der Hauptherausforderungen für die partielle Ausführung dar.

- **Darstellung von Zwischenergebnissen:** Um eine Darstellung von Zwischenergebnissen zu ermöglichen, muss eine partielle Ausführung des DVM ermöglicht werden. Hierbei tritt die Herausforderung auf, dass alle Abhängigkeiten in diesem Modell analysiert werden müssen, um festzustellen, welche Teile partiell ausgeführt werden können und welche Teile aufgrund von fehlenden Zwischenergebnissen (noch) nicht ausführbar sind.
- **Erhöhung der Nutzbarkeit für den Modellierer:** Werden Teile des DVM während der Modellierungszeit ausgeführt, muss der Modellierer einerseits darüber informiert werden und andererseits ständig Feedback über den Stand der Ausführung bekommen. Insbesondere durch das Anzeigen von Zwischenergebnissen kann so die Nutzbarkeit erhöht werden. Eine Herausforderung ist es, die Zwischenstände in einem passenden Format anzuzeigen und zu gewährleisten, dass dies nicht aufdringlich oder störend ist.

Auf Basis dieser Herausforderungen wurden Anforderungen an die Konzepte abgeleitet, die ebenfalls als Grundlage für deren Validierung dienen. Die Anforderungen stammen, wie die Herausforderungen auch, aus einer intensiven Literaturrecherche sowie aus Experteninterviews in diesem Forschungsbereich. Die Anforderungen umfassen:

- **(R1) Definition einer Gültigkeitsdauer für Daten:** Die erste Anforderung ist es, eine Möglichkeit zu schaffen, die Gültigkeitsdauer von Zwischenergebnissen zu definieren. Eine Definition der Gültigkeitsdauer ist notwendig, da es stark von der Art der zugrundeliegenden

Daten abhängt, wie schnell diese veralten. Zum Beispiel ändert sich ein Datensatz über historische Wetterdaten überhaupt nicht, wohingegen Daten, die von einem Sensor produziert werden, (zum Beispiel Messungen von Temperaturwerten jede Sekunde), zu sich ständig ändernden Datenwerten führen. Die Daten veralten also schnell.

- **(R2) Verhindern veralteter Daten:** Sobald die Anforderung **(R1)** erfüllt ist, muss verhindert werden, dass veraltete Daten weiterhin verwendet werden. Hierfür müssen Ergebnisse dynamisch nachladen bzw. aktualisiert werden. Zusätzlich muss eine Überwachung eingeführt werden, die ständig überprüft, ob ein Datensatz veraltet ist und gegebenenfalls einen Auffrischmechanismus anstößt, sobald die Gültigkeitsdauer der Daten erreicht ist.
- **(R3) Überprüfen von Abhängigkeiten:** Eine partielle Ausführung des DVM erfordert eine Möglichkeit die Abhängigkeiten zwischen dessen Knoten zu beschreiben. Das bedeutet zum Beispiel, dass ein Knoten nur dann ausgeführt werden kann, wenn ein Vorgängerknoten zuerst ausgeführt wird. Beispielsweise müssen Daten aus einer Datenbank erst extrahiert werden, bevor diese gefiltert werden können. Derartige Abhängigkeiten müssen dynamisch während der Modellierungszeit erkannt werden.
- **(R4) Aufteilung des Modells in Teilgraphen:** Um eine partielle Ausführung des DVM zu ermöglichen, muss dieses in Teilgraphen aufteilbar sein. Jeder Teilgraph kann dabei Abhängigkeiten zu anderen Teilgraphen besitzen, wie in **(R3)** beschrieben.
- **(R5) Persistente Speicherung von Zwischenergebnissen:** Um eine partielle Ausführung zu ermöglichen und um Zwischenergebnisse dem Modellierer darzustellen, müssen Daten, die durch eine partielle Ausführung entstanden sind, sowohl persistiert als auch effizient auffindbar sein.
- **(R6) Nutzerfeedback:** Nutzerfeedback ist bei einer partiellen Ausführung im Hintergrund von großer Wichtigkeit, da der Modellierer

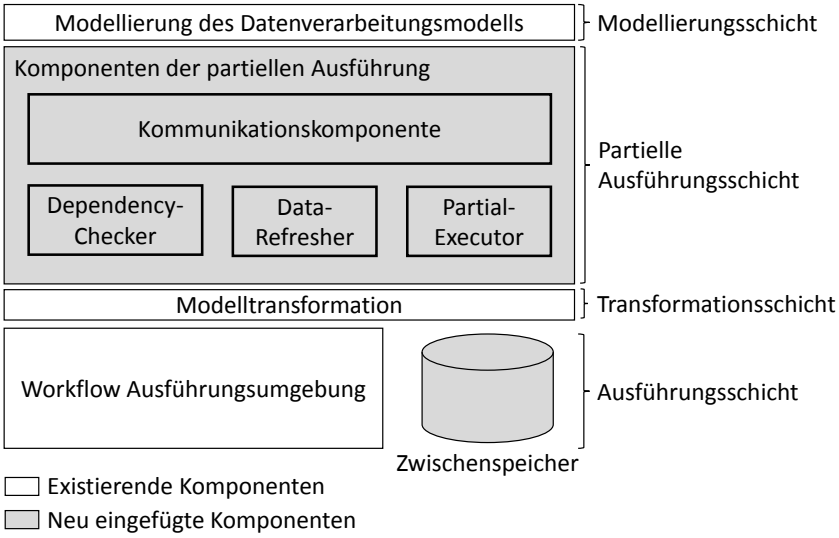

Abbildung 8.12: Erweiterte Architektur (vgl. Abbildung 3.1) zur partiellen Ausführung des Datenverarbeitungsmodells (basierend auf [HBM17])

jederzeit über den Stand der Ausführung informiert sein sollte. Des Weiteren sollte der Modellierer nicht davon überrascht werden, dass Daten bereits verarbeitet werden, ohne dass dies explizit angestoßen wurde. Daher muss eine Möglichkeit geschaffen werden, um Nutzerfeedback über den bisherigen Stand der Ausführung zur Verfügung zu stellen.

8.2.2.3 Architektur und konzeptionelle Lösung

Die bisher in dieser Dissertation beschriebene Architektur wurde für die partielle Ausführung, wie in Abbildung 8.12 dargestellt, erweitert. Hier ist zu sehen, dass die bisherige Architektur aus drei Schichten besteht: 1) die Modellierungsschicht, in der das DVM erstellt wird, 2) die Transformationsschicht, in der das DVM bzw. DVM$^+$ in das DPM überführt wird und

3) die Ausführungsschicht, in der das DPM in einer passenden Umgebung ausgeführt wird. Die in Abschnitt 8.2.1 eingeführte Schicht zur Verteilung der Datenverarbeitung wird aus Gründen der Übersicht an dieser Stelle nicht gezeigt.

Um eine partielle Ausführung zu ermöglichen, wurde eine weitere Schicht zwischen der Modellierung und Transformationsschicht eingeführt, die *Partielle Ausführungsschicht*. Dabei wurden in dieser Schicht folgende Komponenten eingeführt: 1) der *Dependency-Checker*, eine Komponente zur Überprüfung von Abhängigkeiten von Knoten im DVM, 2) der *Partial-Executor*, die Kernkomponente, die das Modell in mehrere Teilgraphen aufteilt und deren partielle Ausführung anstößt, 3) die *Data-Refresher*-Komponente, die durch dynamisches Nachladen veraltete Daten verhindert, und 4) die *Kommunikationskomponente*, die als Schnittstelle zwischen der Modellierung und der Partiellen Ausführungsschicht dient.

Neben der neu hinzugefügten Schicht musste die bisherige Modellierungsschicht angepasst werden, um den in Abschnitt 8.2.2.2 beschriebenen Anforderungen gerecht zu werden. In den folgenden Abschnitten werden diese Änderungen sowie die neu hinzugefügten Komponenten beschrieben.

Anpassung der Modellierung

Die Modellierungsschicht musste angepasst werden, um die Anforderungen R_i zu erfüllen. Hierfür wurden die Parametrisierung der Knoten des Modells erweitert, die neu geschaffene Kommunikationsschnittstelle zur partiellen Ausführung integriert, eine Möglichkeit geschaffen Zwischenresultate in der Modellierungsschicht anzuzeigen sowie ermöglicht, den aktuellen Stand der Ausführung zu verfolgen.

Bezüglich der erweiterten Parametrisierung wurden zwei Parameter hinzugefügt, um die partielle Ausführung zu ermöglichen. Hierbei wurde ein Parameter für die Gültigkeitsdauer von Daten hinzugefügt. Über den Parameter kann bestimmt werden, wie lange ein Resultat eines Knotens im Modell valide ist bzw. wann dieser erneut ausgeführt werden muss. Dadurch kann das Veralten von Daten verhindert werden (vgl. Anforderung **(R2)**). Der zweite Parameter definiert den Stand der Ausführung eines Knotens.

(a) neutral / neu platzierter Knoten

(b) nicht richtig konfiguriert, veraltete Ergebnisse oder Fehler bei der Ausführung

(c) bereit zur Verarbeitung oder in Verarbeitung

(d) 30 Prozent abgeschlossen

(e) abgeschlossen

Abbildung 8.13: Mögliche Zustände von Knoten während der Modellierung und partiellen Ausführung des DVM basierend auf einem Beispielknoten (basierend auf [HBM17])

Dieser Parameter kann die Werte *ausführbar*, *wird ausgeführt*, *unkonfiguriert*, *ausgeführt*, oder *veraltet* annehmen. Dieser Zustand wird ständig durch die partielle Ausführungsschicht synchronisiert.

Neben diesen Änderungen wurde die Modellierungsschicht dahingehend angepasst, dass das Modell nach jeder Änderung an die Komponente zur partiellen Ausführung übergeben wird, beispielsweise bei einer Änderung der Parametrisierung von Knoten oder dem Hinzufügen von neuen Anforderungen. Die Kommunikationskomponente der partiellen Ausführung speichert die Modelle ab und kann somit Unterschiede im Modell feststellen. Sobald ein Unterschied erkannt wird, kann entsprechend eine partielle Ausführung von Knoten des Modells implizit angestoßen werden.

Um dem Modellierer den bisherigen Stand der Ausführung anzuzeigen, wurde die grafische Darstellung der Knoten im Modell angepasst. Dadurch kann der Modellierer den momentanen Stand der Ausführung einsehen und gegebenenfalls auf diesen reagieren, zum Beispiel durch eine Rekonfiguration der Knoten. Abbildung 8.13 zeigt eine Auswahl von möglichen Zuständen, die Knoten während der Modellierung und Ausführung annehmen können. Der in dieser Abbildung dargestellte Knoten repräsentiert eine beispielhafte

Datenquelle eines Krankenhauses, die Patientendaten zur Verfügung stellt. Sobald ein Knoten dem Modell hinzugefügt wurde, wird dieser initial als "neutral" angezeigt (Abbildung 8.13a). Dieser Zustand ändert sich während der Konfiguration des Knotens wie in Abbildung 8.13b dargestellt, bis eine korrekte, vollständige Parametrisierung bereitgestellt wurde, d.h., sobald alle Pflichtfelder erfüllt sind. Hierdurch kann schnell erkannt werden, ob weitere Parametrisierungen von Knoten notwendig sind. Sobald die Parametrisierung abgeschlossen ist, wird der Knoten automatisch ausgeführt und dessen Zustand ändert sich wie in Abbildung 8.13c dargestellt.

Sollte der Service, der die entsprechende Datenoperation ausführt, eine Möglichkeit bieten, den aktuellen Zustand der Verarbeitung abzufragen, kann der Zustand direkt dem Modellierer angezeigt werden (siehe Abbildung 8.13d). Nach der Ausführung ändert sich der Zustand des Knotens entweder zu Abbildung 8.13b, im Falle eines Fehlers, oder zu Abbildung 8.13e im Falle einer erfolgreichen Ausführung. Sobald Ergebnisdaten veraltet sind, also ihre Gültigkeitsdauer erreichen, wird der Zustand des entsprechenden Knotens ebenfalls zu Abbildung 8.13b geändert.

Um Zwischenergebnisse anzuzeigen, wurde eine Möglichkeit geschaffen, die Ergebnisse der partiellen Ausführung aus dem Zwischenspeicher auszulesen. Ob Zwischenergebnisse verfügbar sind, ist aus den oben beschriebenen Zuständen ersichtlich. Dadurch kann früh auf Ergebnisse reagieren werden und ggf. die Parametrisierung angepasst werden. Momentan werden an dieser Stelle jedoch nur Rohdaten angezeigt, eine zu den Daten passende Visualisierung ist Teil zukünftiger Arbeiten.

Dependency-Checker

Der Dependency-Checker findet Abhängigkeiten im DVM. Dabei wird überprüft, ob ein Knoten im Modell basierend auf bisherigen Zwischenergebnissen ausgeführt werden kann oder sogar unabhängig von anderen Knoten ist. Es wird außerdem analysiert, welche Knoten ausgeführt werden müssen, um einen spezifischen Knoten verarbeiten zu können. Diese Analyse wird durch sogenannte *Abhängigkeitsgraphen* ermöglicht, die auf dem DVM basieren und bei dessen Änderung synchronisiert werden. Ein Abhängigkeitsgraph

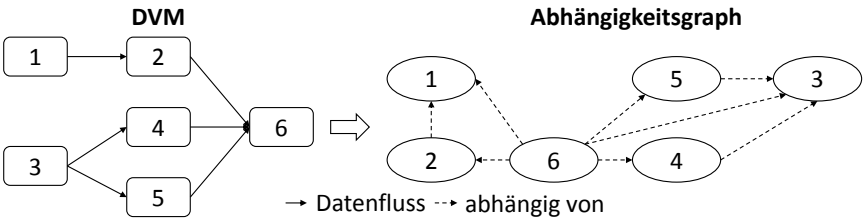

Abbildung 8.14: Beispiel eines Abhängigkeitsgraphen (basierend auf [HBM17])

enthält dieselben Knoten wie das DVM, jedoch statt den Datenfluss mittels den Kanten zu beschreiben, werden Abhängigkeiten beschrieben. Eine gerichtete Kante drückt aus, dass ein Knoten abhängig von einem anderen ist, also dass dieser davor ausgeführt werden muss.

Abbildung 8.14 zeigt ein Beispiel eines initialen Modells ohne bisher ausgeführte Knoten (links). Auf der rechten Seite ist dessen Abhängigkeitsgraph zu sehen. Alle Knoten im Abhängigkeitsgraph können genau dann ausgeführt werden, wenn sie keine ausgehenden Kanten besitzen sowie vollständig und korrekt parametrisiert sind. Sobald ein Knoten erfolgreich ausgeführt wurde, werden alle *eingehenden* Kanten im Abhängigkeitsgraph entfernt. Sobald dessen Ergebnis, aufgrund einer erreichten Gültigkeitsdauer, veraltet ist, werden die Kanten erneut hinzugefügt.

Der Abhängigkeitsgraph dient als Grundlage für die partielle Ausführung, die im Folgenden beschrieben wird.

Partial-Executor

Die Partial-Executor-Komponente stellt die Hauptfunktionalität zur partiellen Ausführung zur Verfügung. Die notwendigen Schritte, um eine partielle Ausführung zu ermöglichen sind in Abbildung 8.15 dargestellt. Nachdem in Schritt 1 das DVM bzw. DVM$^+$ modelliert wurde, wird dieses nach jeder Änderung im Modell an den Dependency-Checker übergeben. Dieser überprüft in Schritt 2, ob Knoten oder Teilgraphen im Modell enthalten sind, die bereits ausgeführt werden können, also die keine Abhängigkei-

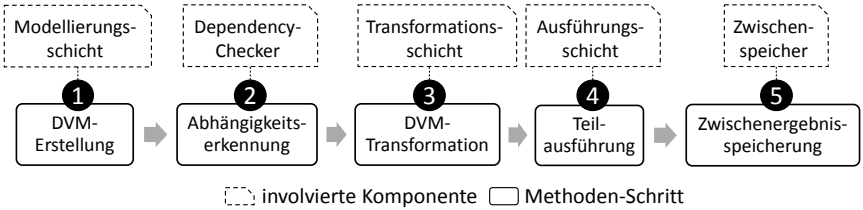

Abbildung 8.15: Schritte der partiellen Ausführung

ten besitzen. Die ausführbaren Teile des DVM werden anschließend an die Partial-Executor-Komponente übergeben. Diese übergibt die Teilgraphen an die Transformationsschicht (vgl. Kapitel 5), die die DVM-Teile in ausführbare Teil-DPMs (bspw. BPEL-Fragmente) überführt (Schritt 3). Hierfür musste die bisher verfügbare Transformation nicht angepasst werden. Die überführten Teil-DPMs werden, ggf. basierend auf vorigen Ergebnissen, einzeln in der entsprechenden Umgebung ausgeführt (Schritt 4) und die Ergebnisse in dem Zwischenspeicher (Schritt 5) abgelegt. Die Zwischenergebnisse können anschließend in der Modellierungsschicht visualisiert werden. Die Partial-Executor-Komponente übernimmt die Koordination der involvierten Komponenten.

Eine Herausforderung bei der partiellen Ausführung ist das Extrahieren vorheriger Ergebnisse als Grundlage der Ausführung einzelner Fragmente. Aufgrund dessen, dass der Stand der Ausführung überwacht wird, kann entschieden werden, ob bereits Zwischenergebnisse verfügbar sind oder ob vor der Ausführung eines Knotens zuerst weitere Knoten im Graph ausgeführt werden müssen. Nimmt ein abhängiger Knoten den Zustand *ausgeführt* an, liegen Zwischenergebnisse vor, die für eine weitere Ausführung verwendet werden können (siehe Abbildung 8.13e). Besitzt dieser Knoten jedoch den Zustand *veraltet* (vgl. Abbildung 8.13b), muss dieser erneut ausgeführt werden, da die Ergebnisse nicht mehr aktuell sind. In diesem Fall muss zuerst gewartet werden, bis die abhängigen Knoten erfolgreich ausgeführt wurden. In manchen Fällen werden Knoten ausgeführt und während der Ausführung ändert sich deren Konfiguration. In diesem Fall muss der Knoten erneut

ausgeführt werden, da sonst mit veralteten Zwischenergebnissen weitergerechnet wird. Daher stellt ein sich häufig änderndes Modell auch eine große Herausforderung für eine partielle Ausführung dar, da eine wiederholte Ausführung mit Kosten verbunden ist.

Die Konzepte zur partiellen Ausführung funktionieren am besten, wenn das DVM schrittweise von den Datenquellen zu den Datensenken modelliert wird. Wird umgekehrt vorgegangen, führt dies dazu, dass die meisten Knoten nicht partiell ausgeführt werden können, da sie Abhängigkeiten zu den noch nicht modellierten Datenquellen oder darauf aufbauende Knoten besitzen. Daher ist eine Modellierungsmethodik von den Datenquellen zu den Datensenken Grundvoraussetzung für eine gut funktionierende Lösung.

Data-Refresher

Die Data-Refresher-Komponente verhindert das Veralten von Zwischenresultaten. Es wird laufend überprüft, ob Daten ihre Gültigkeitsdauer erreicht haben und, falls notwendig, werden diese umgehend erneuert. Wie bereits in Abschnitt 8.2.2.3 beschrieben, wurde im Modell ein zusätzlicher Parameter für die Gültigkeitsdauer von Resultaten eingeführt. Die Data-Refresher-Komponente misst die Zeit nach der Resultaterstellung und setzt den Zustand des entsprechenden Knotens im Modell auf *veraltet*, sobald dessen Gültigkeitsdauer abgelaufen ist. Daraufhin wird der Abhängigkeitsgraph angepasst und die Wiederausführung des betroffen Knotens wird angestoßen (siehe Abschnitt 8.2.2.3).

Kommunikationskomponente

Die Kommunikationskomponente übernimmt die Kommunikation mit der Modellierungsschicht. Alle Anfragen werden von der Modellierungsschicht an die entsprechenden Komponenten weitergegeben. Sobald beispielsweise ein Knoten vollständig parametrisiert oder geändert wurde, wird die partielle Ausführung dieses Knotens durch die Partial-Executor-Komponente angestoßen. Des Weiteren übernimmt die Kommunikationskomponente das Benachrichtigen des Dependency-Checkers, sobald die Ausführung erfolgreich war. Analog wird die Modellierungsschicht benachrichtigt, sodass der

geänderte Zustand dem Modellierer zur Verfügung gestellt werden kann. Die Kommunikationskomponente abstrahiert von den restlichen Komponenten und ermöglicht eine einfachere Kommunikation mit der Modellierungsschicht, da nur noch eine einzelne Schnittstelle bedient werden muss anstatt der vielen Komponentenschnittstellen.

8.2.2.4 Evaluation und Implementierung der Konzepte

In diesem Abschnitt werden die Konzepte zur partiellen Ausführung basierend auf den beschriebenen Anforderungen aus Abschnitt 8.2.2.2 evaluiert.

Evaluation

Die erste Anforderung **(R1)** fordert eine Definition der Gültigkeitsdauer von Daten, die festlegt, wie lange Zwischenresultate gültig sind. Dies ist notwendig, um veraltete Ergebnisse zu erkennen. Diese Anforderung konnte erfüllt werden, indem ein neuer Parameter zu den Knoten im DVM hinzugefügt wurde: die maximale Gültigkeitsdauer. Um eine Parametrisierung durch Domänennutzer zu ermöglichen, kann dieser lediglich die Werte *keine*, *unbegrenzt*, oder *begrenzt* annehmen. Durch diesen Parameter kann festgestellt werden, ob Resultate der partiellen Ausführung veraltet sind.

Die zweite Anforderung **(R2)** fordert, dass das Veralten von Daten verhindert werden muss. Dies wurde durch die vorgestellte Komponente "Data-Refresher" ermöglicht (vgl. Abschnitt 8.2.2.3). Diese Komponente überwacht die Zwischenergebnisse und überprüft ob deren maximale Gültigkeitsdauer erreicht wurde. Ist dies der Fall, wird die erneute Resultatgenerierung durch partielle Ausführung des Modells angestoßen.

Die dritte Anforderung **(R3)** fordert, dass Abhängigkeiten im DVM analysiert werden müssen, damit eine Entscheidung getroffen werden kann, welche Teile ausführbar sind. Dies wird durch den vorgestellten Dependency-Checker ermöglicht, der über die Abhängigkeitsgraphen darstellt, welche Komponenten ausführbar sind. Die Überwachung der Abhängigkeiten wird durch diese Graphen umgesetzt.

Die vierte Anforderung **(R4)** fordert, dass das Modell in Teilgraphen auf-

geteilt werden muss, damit die Teilgraphen einzeln ausführbar sind, statt das Modell als Ganzes zu transformieren und auszuführen. Die Anforderung wird durch die Partial-Executor-Komponente erfüllt, die das Modell in Teilgraphen aufteilt und diese einzeln ausführen kann.

Die fünfte Anforderung **(R5)** fordert das Speichern von Zwischenergebnissen. Dies wird durch einen leichtgewichtigen Zwischenspeicher ermöglicht. Die Zwischenergebnisse werden als Ganzes gespeichert, ohne das Ziel diese durchsuchen oder traversieren zu können. Hierfür ist eine Speicherung als Schlüssel-Wert-Paar ausreichend. Der Zwischenspeicher sollte nah am Ausführungsort liegen, um einen Kommunikationsoverhead zu vermeiden.

Die letzte Anforderung **(R6)** fordert, dass Nutzerfeedback bereitgestellt werden muss, damit dem Modellierer bewusst wird, dass das Modell partiell im Hintergrund ausgeführt wird. Nutzerfeedback wird durch die Anpassung der Modellierungsschicht ermöglicht. Es wurde eine Möglichkeit geschaffen, den momentanen Stand der Ausführung sowie Zwischenergebnisse anzuzeigen (vgl. Abbildung 8.12).

Die Konzepte dieses Beitrags ermöglichen darüber hinaus die folgenden Vorteile: Durch die partielle Ausführung kann die für die Modellierung benötigte Zeit bereits genutzt werden, um Teile des Modells im Hintergrund auszuführen. Dadurch kann eine schnellere Gesamtausführung erreicht werden. Dies wirkt sich positiv auf die Nutzererfahrung aus. Des Weiteren wird dadurch auch das Anzeigen von Zwischenergebnissen ermöglicht. Zwischenergebnisse sind wichtig, um die Ausführung transparenter zu machen und die Nutzererfahrung zu steigern.

Implementierung

Die Konzepte zur partiellen Ausführung wurden durch die von mir betreute studentische Masterarbeit von Neugebauer [Neu17] implementiert und in die bestehende Implementierung des Gesamtkonzeptes integriert. Die Implementierung kann wie folgt aufgeteilt werden: 1) Änderungen in der Oberfläche des Modellierungswerkzeugs und 2) Neuimplementierung der Komponenten in der Partiellen Ausführungsschicht.

Um die Konzepte vollständig integrieren zu können, musste das darunter-

liegende Framework Alloy ausgetauscht werden, da dieses nicht die notwendigen Funktionalitäten, beispielsweise das Einfärben der Knoten im Modell, unterstützt. Dieses Framework wurde im Rahmen dieser Implementierung durch jsPlumb[1] ersetzt. Dadurch ist es möglich, den Stand der Ausführung durch Einfärben der Knoten darzustellen (vgl. Abbildung 8.13). Des Weiteren wurde eine Visualisierung für Zwischenresultate geschaffen.

Die Partielle Ausführungsschicht wurde neu implementiert und kann durch eine HTTP-Schnittstelle erreicht werden. Diese Schnittstelle wird von der Modellierungsschicht aufgerufen, um Informationen über den bisherigen Stand sowie Zwischenergebnisse abzurufen. Die Schicht wurde vollständig in Java implementiert und auf einem Apache-Tomcat-Applikationsserver betrieben. Um Zwischenergebnisse zu speichern wird eine MongoDB-NoSQL-Datenbank verwendet. Die Schnittstellen nach außen wurden mittels Java-Servlets geschaffen. Jede der genannten Komponenten wurde modular in separaten Java-Paketen implementiert. Diese werden mittels des Kompositionswerkzeugs Apache-Ant zusammengefügt.

8.2.2.5 Partielle Ausführung – Zusammenfassung und zukünftige Arbeiten

In diesem Abschnitt wurde ein Konzept beschrieben, mit dem das vorgestellte DVM partiell während der Modellierungszeit ausgeführt werden kann. Um eine partielle Ausführung zu ermöglichen, wurde eine neue Schicht in die bisherige Architektur zwischen der Modellierung und Transformation hinzugefügt. Diese Schicht enthält Komponenten, um Abhängigkeiten im Modell zu erkennen, dieses in Teilgraphen aufzuteilen und die Teilgraphen einzeln auszuführen. Hierbei werden Zwischenergebnisse der Ausführung in einem Zwischenspeicher persistiert, um als Grundlage für die Ausführung weiterer Teilgraphen zu dienen. Es wurde eine Möglichkeit geschaffen, die Gültigkeitsdauer von Zwischenresultaten zu definieren, um festzustellen, wann die Resultate veraltet sind und neu berechnet werden müssen.

Darüber hinaus wurde die Modellierungsschicht angepasst, um die neuen Konzepte zu unterstützen. Hierfür wurde es ermöglicht, dem Modellierer

[1] https://jsplumbtoolkit.com/

Feedback über den Stand der Ausführung zu bekommen. Des Weiteren ist es nun auch möglich, während der Modellierung Zwischenergebnisse einzusehen. In der Zukunft sollten außerdem Nutzerstudien durchgeführt werden, um die Konzepte zu evaluieren.

Momentan existierten einige Limitierungen. Bisher muss die Gültigkeitsdauer der Ergebnisse manuell bei der Parametrisierung der Knoten angegeben werden. Die Angabe der Gültigkeitsdauer ist eine schwierige Aufgabe, da hierfür explizites Wissen darüber notwendig ist, ab welchem Zeitpunkt Daten nicht mehr weiter verwendet werden sollten. Aus diesem Grund wird in zukünftigen Arbeiten an einem Konzept gearbeitet, um dies automatisch zu bestimmen. Neben diesen Arbeiten ist eine vollständige Integration in den von Behringer et al. [BHM17] vorgestellten Ansatz für nutzerbezogene Visuelle Analyse geplant.

8.3 Zusammenfassung

In diesem Kapitel werden die DVM-Ausführung sowie zwei Verbesserungen beschrieben, mit denen die Ausführung effizienter bzw. nutzerfreundlicher gestaltet werden kann.

Zusätzlich zur Beschreibung der Ausführung wurden zwei Verbesserungen vorgestellt. Bei der ersten Verbesserung wird beschrieben, wie die Datenverarbeitung durch eine Verteilung auf ein Rechencluster effizienter durchgeführt werden kann. Hierbei wurde insbesondere untersucht, wann eine Verteilung der Datenverarbeitung überhaupt sinnvoll ist und wie sich die Steigerung der Clustergröße auf die Effizienz auswirkt.

Die zweite Verbesserung beschreibt die partielle Ausführung des DVM während der Modellierungszeit. Die partielle Ausführung führt zu einer gefühlt schnelleren Ausführung, da viele Teile des Modells bis zur vollständigen Modellierung bereits ausgeführt wurden. Des Weiteren wird dadurch das Anzeigen von Zwischenergebnissen ermöglicht. Damit kann eine verbesserte Nutzerfahrung ermöglicht werden. Zur Realisierung wurde eine neue Schicht in die bisherige Architektur eingefügt, die die Koordination der partiellen

Ausführung übernimmt und die Modellierungsschicht entsprechend anpasst.

KAPITEL 9

GESAMTEVALUATION

In diesem Kapitel werden die Gesamtbeiträge dieser Dissertation evaluiert. Hierfür wurde eine qualitative sowie eine quantitative Evaluation durch die Integration der Konzepte in Forschungsprojekte des Instituts für Parallele und Verteilte Systeme der Universität Stuttgart durchgeführt.

9.1 Qualitative Evaluation

Zur Evaluation der Beiträge dieser Dissertation wurde im Jahr 2015 in Rotterdam sowie im Jahr 2016 in Lugano an der *Rapid-Mashup-Challenge (RMC)* im Rahmen der International Conference on Web Engineering (ICWE) teilgenommen[1]. Bei der RMC geht es darum, basierend auf einer vorgegebenen Aufgabenstellung, Mashups zu realisieren. Das Ziel war dabei eine Komposition verschiedenartiger Datenquellen, um den größtmöglichen Nutzen aus den enthaltenen Daten zu gewinnen. Dabei wurde die Aufgabenstellung (u.a. welche Datenquellen verwendet werden müssen) jeweils einen Monat vor der Challenge veröffentlicht. Ich nahm dabei mit den Konzepten dieser

[1] Webseite des Jahres 2015: `http://mashup.inf.usi.ch/challenge/2015/`; Webseite des Jahres 2016: `http://challenge.webengineering.org/`

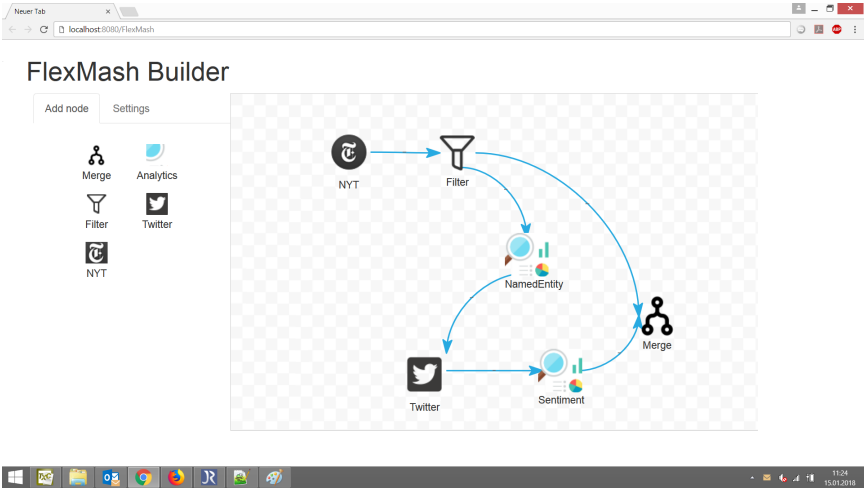

Abbildung 9.1: Präsentiertes DVM$^+$ bei der Rapid Mashup Challenge 2015 (aus [HM16a])

Dissertation teil und fokussierte mich somit auf Data-Mashups, also auf die Komposition von Daten heterogener, verteilter Quellen. Dabei wurde das vorgestellte DVM sowie dessen Anforderungsannotationen (DVM$^+$) und die Transformation in das DPM genutzt, um die gestellten Aufgaben zu lösen. Die Ergebnisse wurden in [HM16a] für das Jahr 2015 und in [HB17] für das Jahr 2016 veröffentlicht. Bei der Challenge wurden neben den Beiträgen dieser Dissertation auch deren prototypische Implementierung – genannt FlexMash – vorgestellt. Mit dem entwickelten Gesamtansatz dieser Dissertation wurde die Aufgabenstellung der Challenge bearbeitet und das Ergebnis einer Fachjury sowie einem Expertenpublikum aus dem Bereich Webentwicklung und Datenverarbeitung vorgestellt. Zur Vorstellung der Konzepte und des Werkzeugs standen hierfür jeweils 10 Minuten zur Verfügung.

Dabei waren wichtige Bewertungskriterien die Originalität der Konzepte sowie die Mächtigkeit deren Umsetzung. Die genauen Kriterien stammen aus [PD16] und sind im Folgenden aufgelistet:

- **Konzeptidee** Das Kriterium Konzeptidee umfasst vor allem die Nütz-

lichkeit des Ansatzes sowie die enthaltene Funktionalität.

- **Komplexität der Lösung** Mit diesem Kriterium wird bewertet, wie komplex das zu lösende Problem ist sowie wie schwierig es ist, die Lösung hierfür zu entwickeln.
- **Eleganz des Lösungsansatzes** Dieses Kriterium beschäftigt sich mit der Einfachheit und Verständlichkeit der entstandenen Lösung.
- **Mächtigkeit des Werkzeugs** Dieses Kriterium bewertet den Umfang und die Mächtigkeit der Funktionalität des Werkzeugs.

Bewertet wurden die Kriterien jeweils auf einer Skala von 1 bis 5 Punkten. Im Rahmen der Rapid-Mashup-Challenge 2015 wurde ein Analyseszenario gezeigt, bei dem unstrukturierte Artikel aus der New York Times-Datenbank analysiert wurden bezüglich der Einschätzung (engl. sentiment) des Lesers über das dort behandelte Thema (positiv, negativ, neutral). Dabei wurde eine positive Einschätzung von einer neutralen und negativen unterschieden. Hierfür wurden im ersten Schritt die Artikel aus der Datenquelle extrahiert, und es wurden wichtige Stichworte mittels Text Mining (Named Entity Recognition) extrahiert. Basierend auf diesen Stichworten wurden zum Thema passende Tweets aus der Twitter API gesucht und hinsichtlich Lesereinschätzung bewertet. Die Ausgabe ist eine Bewertung für jeden Artikel einer bestimmten Kategorie (Sport, Wirtschaft, Politik, etc.). Das DVM für dieses Szenario, das bei der Challenge vorgestellt wurde, ist in Abbildung 9.1 dargestellt. In Tabelle 9.1 und Tabelle 9.2 sind die Challenge-Ergebnisse für das Jahr 2015 im Vergleich zu sehen. Man erkennt, dass FlexMash in den wichtigen Kategorien “Konzeptidee” und “Komplexität der Lösung” am besten abschneidet. Lediglich bei der Mächtigkeit des Werkzeugs gab es Abzüge, da die Werkzeugimplementierung zum Zeitpunkt der Challenge ein früher Prototyp war.

Dabei belegte ich den zweiten Platz bei der Challenge, d.h., dass das entstandene Konzept und die zugehörige Implementierung FlexMash von der Fachjury positiv bewertet wurde.

Im Jahr 2016 trat ich erneut bei der Challenge an, um eine Weiterentwick-

Tool	Mashup Idea	Mashup Complexity	Mashup Solution Elegance	Tool Power	Number of Votes
FlexMash	3.35	3.54	3.15	2.92	13
UI-Oriented Computing	3.07	2.57	3.18	2.36	14
SmartComposition	2.79	2.79	2.61	2.79	14
EFESTO	3.29	3.36	3.29	3.82	14
WebMakeup	2.61	2.21	2.64	2.96	14
WLS	3.07	2.90	2.70	3.10	15

Tabelle 9.1: Bewertung des Prototypen FlexMash bei der Rapid Mashup Challenge 2015 (aus [PD16])

Position	Tool	Total score
1	EFESTO	13.75
2	FlexMash	12.96
3	WLS	11.77
4	UI-Oriented Computing	11.18
5	SmartComposition	10.96
6	WebMakeup	10.48

Tabelle 9.2: Gesamtergebnis der Rapid Mashup Challenge 2015 (aus [PD16])

lung des Ansatzes und des Prototypen, FlexMash 2.0, vorzustellen. Dabei wurde ein Analyseszenario zur Untersuchung von Verkehrsunfällen in New York City thematisiert. Hierfür wurden Unfalldaten aus der New York City Open-Data-Plattform mit Wetterdaten aggregiert, und es wurden Zusammenhänge zwischen diesen Daten untersucht. Das DVM für diese Analyse ist in Abbildung 9.2 dargestellt. Zusätzlich wurden Konzepte für eine interaktive Analyse von Daten präsentiert, die auf [Beh16; BHM17] basieren.

Bei der Challenge 2016 haben sich die Bewertungskriterien im Vergleich zu 2015 geändert [GD17]:

- **Ausdrucksstärke** Dieses Kriterium bewertet den Funktionsumfang der Tools – je mehr Funktionen unterstützt werden, desto höher die Bewertung.

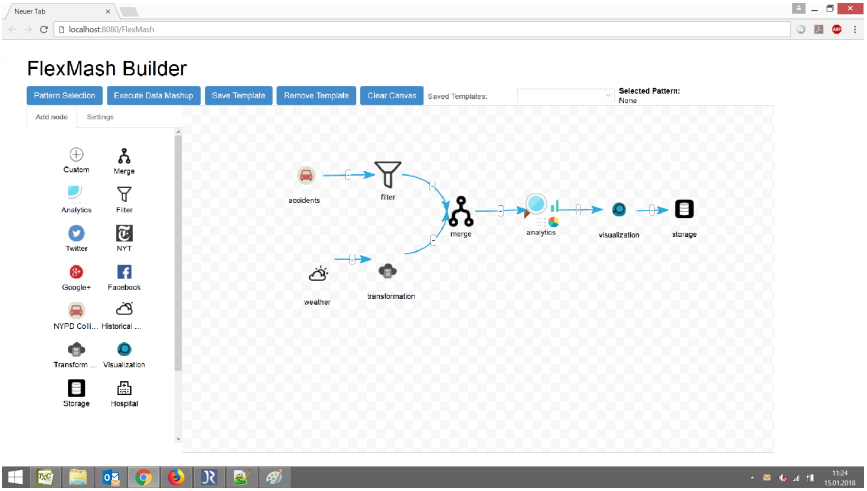

Abbildung 9.2: Präsentiertes DVM$^+$ bei der Rapid Mashup Challenge 2016 (aus [HB17])

Rank	Tool	Expressive power	Flexibility	Maturity	Intuitiveness	Demo effectiveness	Average
1	SmartComposition	3.73	3.77	3.55	3.45	3.68	3.64
2	FlexMash 2.0	3.42	3.15	3.77	3.42	3.35	3.42
3	EFESTO	2.50	2.44	3.00	3.38	2.50	2.76
4	Search-based mashups	2.67	2.56	3.72	2.33	2.44	2.74
5	Uduvudu Editor	2.88	2.71	2.33	2.42	2.75	2.62
6	Linked widgets	2.32	2.46	2.82	1.93	2.18	2.34
7	CAMUS	1.85	1.60	1.95	1.60	1.90	1.78
8	Toolet	1.50	1.61	1.17	1.50	1.33	1.42

Tabelle 9.3: Bewertungsergebnisse der Rapid Mashup Challenge 2016 [GD17]

- **Flexibilität** Dieses Kriterium bewertet vor allem die Erweiterbarkeit bzw. Anpassbarkeit des Ansatzes an verschiedene, neu auftretende Anforderungen. Je geringer der Anpassungsaufwand bei neuen Anforderungen ist, desto besser die Bewertung.
- **Ausgereiftheit** Das Kriterium Ausgereiftheit bewertet, wie weit die

Entwicklung fortgeschritten ist. Je näher der entwickelte Prototyp an einer "produktionsreifen" Version ist, desto höher die Bewertung dieses Kriteriums.

- **Intuitivität** Dieses Kriterium beschreibt, wie intuitiv der Ansatz ist, d.h., wie schnell unerfahrene Nutzer damit umgehen können bzw. welche Vorkenntnisse notwendig sind. Je geringer das notwendige Vorwissen, desto höher wird dieses Kriterium bewertet.
- **Effektivität der Demo** Dieses Kriterium bewertet, ob die Demo die Zuschauer von dem Mehrwert und der Effektivität des Ansatzes überzeugt. Dabei wird auch das Auftreten des Vortragenden bewertet.

Die Ergebnisse der Rapid-Mashup-Challenge 2016 sind in Tabelle 9.3 dargestellt. Man erkennt, dass die FlexMash-Implementierung in allen Bereichen gut abschneiden konnte, insbesondere bezüglich "Ausgereiftheit" und "Ausdrucksstärke".

Zusammenfassend ist festzustellen, dass sowohl die Konzepte dieser Dissertation als auch die prototypischen Implementierungen FlexMash bzw. FlexMash 2.0 durch eine unabhängige Fachjury und im Wettbewerb mit konkurrierenden Lösungsansätzen und -prototypen überaus positiv abgeschnitten haben. Vergleicht man die Bewertungskriterien der beiden RMC-Teilnahmen mit den für die vorliegende Dissertation genannten Zielen und Anforderungen (vgl. Abschnitt 1.2), so stellt man eine recht große Übereinstimmung fest. Daher ist die positive Gutachtereinschätzung der Challenge auch als eine Bewertung zu den in der Dissertation aufgestellten Ziele und Herausforderungen zu sehen.

9.2 Quantitative Evaluation

In diesem Abschnitt wird die quantitative Evaluation beschrieben, die durch die Integration von in dieser Dissertation entwickelten Konzepten in Forschungsprojekte des Instituts durchgeführt wurde.

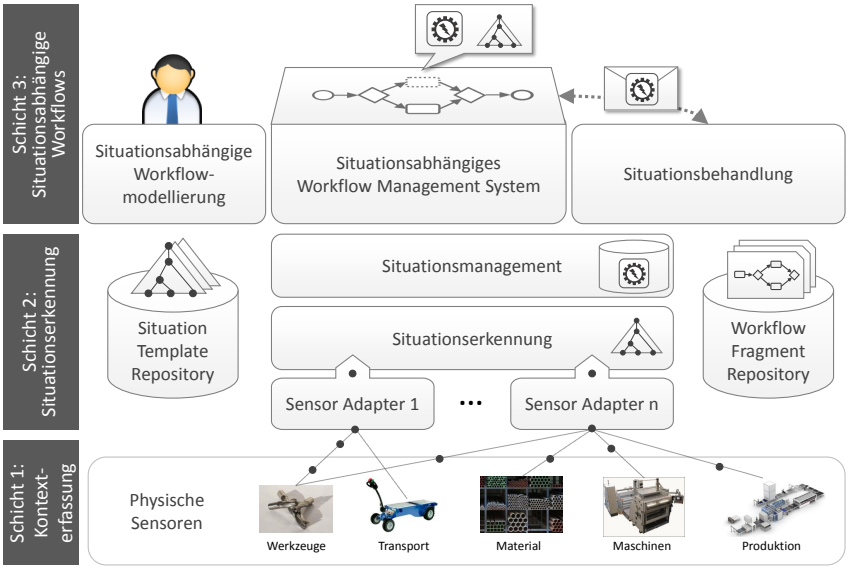

Abbildung 9.3: Übersicht über das Projekt SitOPT (aus Wieland et al. [WSBL15])

9.2.1 Integration in das Projekt SitOPT

SitOPT [WSBL15] ist ein Projekt, finanziert durch die Deutsche Forschungsgemeinschaft, das sich mit der situationsbedingten Adaption von Workflows befasst. Genauer geht es dabei darum, eine situationsabhängige Modellierung von Workflows zu ermöglichen, wobei abhängig vom Kontext verschiedenartige Workflowfragmente eingefügt werden. Beispielsweise kann ein Workflow der die Produktion einer Fabrik steuert beim Ausfall einer Maschine so angepasst werden, dass die Produktion auf eine baugleiche Maschine umgeleitet wird. Des Weiteren könnte bei auftretendem Materialmangel automatisch ein Workflow angestoßen werden, der eine Materialbestellung veranlasst. Die Konzepte dieser Dissertation wurden im Rahmen dieses Projektes integriert, verifiziert, und evaluiert.

Die Integration der Konzepte betrifft verschiedene Teile des Projektes, des-

sen Gesamtarchitektur in Abbildung 9.3 dargestellt ist. In dieser Abbildung ist zu sehen, dass das Gesamtkonzept von SitOPT in zwei Teile gegliedert ist: 1) die Erkennung von Situationen basierend auf Kontextdaten (Ebene 1 und 2), wobei Effizienz und Zuverlässigkeit von großer Wichtigkeit sind, und 2) die Adaption der Workflows basierend auf den erkannten Situationen (Ebene 3). Die Integration der Konzepte dieser Dissertation findet vor allem in den Ebenen Situationserkennung und Kontexterfassung statt. Im Folgenden wird beschrieben, wie die Konzepte dieser Dissertation angewandt wurden, um die Ziele dieses Projektes zu erreichen.

9.2.1.1 Situationserkennung basierend auf Kontextdaten

Der folgende Abschnitt basiert größtenteils auf den folgenden Veröffentlichungen [HWS+15; HWS+16]. Eine der Hauptaufgaben zur Ermöglichung situationsbasierter Workflows ist das Erkennen von Situationen basierend auf Kontextdaten. Eine Situation ist eine Zustandsänderung in der Umgebung, z.B. der Ausfall einer Produktionsmaschine. Als Kontextdaten werden insbesondere Sensordaten bezeichnet, die die entsprechende Umgebung überwachen. Jedoch können hier auch weitere Datenquellen in Frage kommen, beispielsweise Prozessdaten oder statische Datenquellen wie Datenbanken. Durch eine Filterung und Aggregation von Kontextdaten verschiedener Quellen können Situationen erkannt werden.

Das Ziel ist es einerseits die Datenverarbeitung zur Erkennung von Situationen zu modellieren und andererseits effizient zur Situationserkennung auszuführen. Bei der Modellierung sollen sowohl die zur Erkennung der Situation verwendeten Datenquellen (z.B. Sensoren) als auch die notwendigen Datenoperationen (Filterung und Aggregation) definierbar sein. Offensichtlich werden damit die in dieser Dissertation entwickelten Modellierungskonzepte des DVM direkt angesprochen. Im Folgenden wird die von Dey et al. [Dey01] eingeführte Definition von Kontext verwendet: *"any information that can be used to characterize the situation of an entity, where an entity can be a person, place, or object"*. Das bedeutet, dass Kontext als jegliche Information bezeichnet werden kann, die eine Situation einer bestimmten

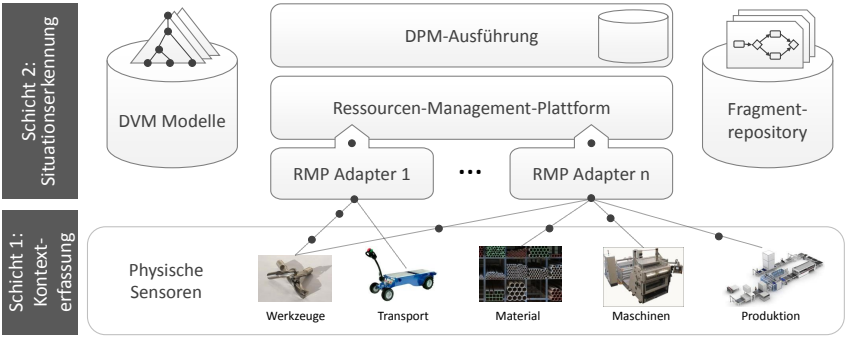

Abbildung 9.4: Situationserkennung in SitOPT durch die Beiträge dieser Dissertation

Entität beschreibt, wobei eine Entität eine Person, ein Ort, oder ein Objekt sein kann.

Um die Modellierung sowie die Erkennung von Situationen zu ermöglichen, wurde das in den Konzepten dieser Dissertation eingeführte DVM-Modellierungskonzept zugrunde gelegt. In Anbetracht dessen, dass Kontextdaten im Rahmen dieses Projekts vorrangig von Sensoren erfasst wurden, ist ein vereinfachtes Modell zur Datenextraktion und -verarbeitung ausreichend. Vor allem die einfache Struktur und die geringe Datenmenge, die von Sensoren produziert werden, erfordert kein komplexes Modell. Die meisten erfassten Sensordaten enthalten jeweils einen Wert in einem primitiven Datentyp wie Ganzzahl, Gleitkommazahl, oder Wahrheitswert sowie einen Zeitstempel, der beschreibt wann dieser Wert erfasst wurde. Sensordaten werden meist über *Middleware*-Komponenten zur Verfügung gestellt, die an die Sensoren direkt angeschlossen werden und deren Werte über eine Schnittstelle bereitstellen. Hierfür wird die in Kapitel 7 beschriebene Ressourcen-Management-Plattform verwendet. Die RMP bietet bereits Funktionalitäten, um Sensordatenquellen automatisch mittels Adaptern dynamisch anzuschließen und über eine uniforme Schnittstelle auf deren Daten zuzugreifen. Die Architektur zur Umsetzung der Situationserkennung in SitOPT durch die in dieser Dissertation vorgestellten Beiträge ist in Abbildung 9.4 zu sehen.

Um Situationen basierend auf Sensorwerten zu erkennen sind einfache Vergleichsoperationen ausreichend, wie zum Beispiel *"falls Temperaturwert größer als 50 Grad, melde Situation Überhitzung"*. Aus diesem Grund wurden die modellierbaren Datenoperationen auf vier Knotentypen begrenzt: 1) ein Extraktionsknoten zur Extraktion der Sensordaten, 2) ein Filterknoten, der Sensorwerte mit festgelegten Werten vergleicht (z.B. Wert > 50), 3) ein Aggregationsknoten, der verschiedene Vergleichsknoten miteinander verbindet (entspricht den logischen Operatoren AND, OR, XOR), und 4) ein Knoten der die erkannte Situation an die Workflowadaption weitergibt. Die Auswahl dieser Knoten zur Erkennung von Situationen basiert auf den von Zweigle et al. [ZHKL09; Zwe11] und Häusermann [HHL+10] eingeführten Situation Templates, ebenfalls ein graphbasiertes Modell mit folgenden Knotentypen:

- **Kontextknoten**
  Kontextknoten repräsentieren eine Kontextdatenquelle, beispielsweise einen Sensor. Somit steht jeder Kontextknoten für einen Sensorwert zu einem spezifischen Zeitpunkt. Kontextknoten werden ausschließlich mit Bedingungsknoten verbunden.
- **Bedingungsknoten**
  Bedingungsknoten werden zum Vergleich der Kontextdaten mit festen Werten genutzt. Hierdurch werden die durch die Kontextknoten bereitgestellten Daten gefiltert. Hierfür stehen die Operationen *kleiner als*, *größer als*, *gleich* und *ungleich* zur Verfügung.
- **Operationsknoten**
  Operationsknoten dienen dazu mehrere Bedingungsknoten zu aggregieren, wobei die Vergleichsoperatoren AND, OR, oder XOR verwendet werden. Operationsknoten werden mit Situationsknoten verbunden.
- **Situationsknoten**
  Der Situationsknoten repräsentiert die Situation selbst. Ist die Ausgabe des Operationsknotens *wahr*, so ist die Situation eingetreten.

Abbildung 9.5 zeigt auf der linken Seite ein Beispiel eines Situation Tem-

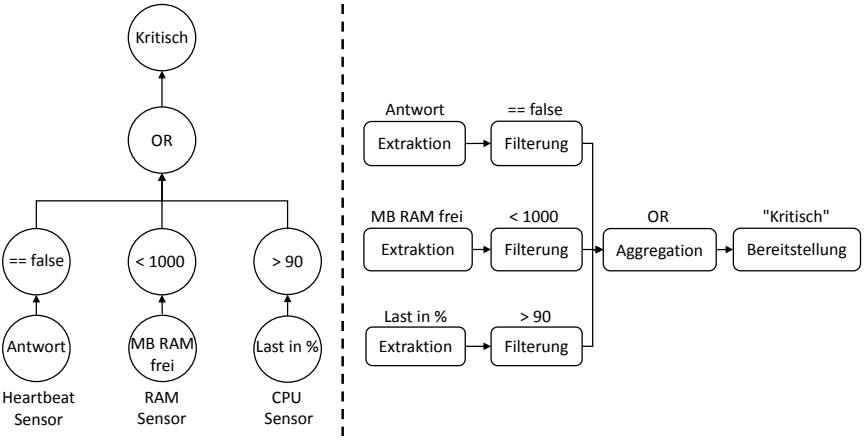

Abbildung 9.5: Beispiel eines Situation Templates (links), Abbildung mittels DVM (rechts)

plates. Zur Veranschaulichung der Funktionsweise wird eine sehr einfache Situation modelliert. Das Ziel ist die Fernüberwachung eines Rechners, um zu bestimmen, ob der Rechner eine *kritische* Auslastung annimmt. Hierfür werden verschiedene Sensoren bereitstellt: 1) einen Heartbeat-Sensor, der mittels periodischen Nachrichten feststellt, ob der Rechner erreichbar ist oder nicht, 2) einen RAM Sensor, der die Auslastung des Hauptspeichers überwacht, und 3) einen CPU Sensor, der die CPU-Auslastung überwacht. Diese Sensoren wurden im Beispiel mittels Kontextknoten modelliert. Damit die Situation eintritt, muss für jeden dieser Sensoren eine bestimmte Bedingung gelten, die über die Bedingungsknoten modelliert wurden. Für den Heartbeat-Sensor muss gelten, dass dieser Sensor keine Antwort zurückgibt und somit den Wert “false” liefert, der RAM-Sensor muss weniger als 1000 MB zur Verfügung haben und der CPU-Sensor mehr also 90% Auslastung messen. Trifft eine dieser Bedingungen zu, was in diesem Beispiel durch den Operationsknoten *OR* modelliert wurde, trifft auch die Situation *kritisch* zu.

Die ursprünglich von Zweigle et al. eingeführten Templates wurden bisher lediglich nur zur Modellierung von Situationen, nicht zu dessen Erkennung

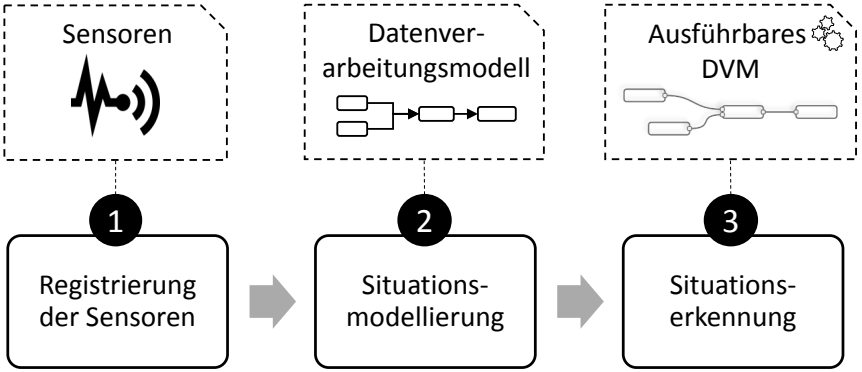

Abbildung 9.6: Methode zur Modellierung und Erkennung von Situationen (basierend auf [HWS+16])

genutzt. Um eine Situationserkennung zu ermöglichen, muss das abstrakte Situation Template Modell ausführbar gemacht werden. Hierfür wird dieses Modell als DVM nachmodelliert. Dies ist in Abbildung 9.5 rechts dargestellt. Es ist zu sehen, dass Situation Templates vollständig abgebildet werden können, um ausführbar zu werden. Es Bedarf lediglich der Modellierung der einfachen Datenoperationen *Extraktion*, *Filter*, und *Aggregation*, um Kontextknoten, Bedingungsknoten, Operationsknoten und Situationsknoten abzubilden (vgl. Abbildung 9.5, rechts).

Der vollständige Ablauf von der Situationsmodellierung bis zur Erkennung ist in Abbildung 9.6 dargestellt. Es sind drei übergeordnete Schritte notwendig: 1) die Registrierung von Sensoren um deren Kontextdaten für eine Verarbeitung zugreifbar zu machen, 2) Modellierung der Situation mittels dem DVM, wie im vorigen Abschnitt beschrieben und 3) Erkennung der Situationen durch Transformation und Ausführung des Modells.

Die Registrierung der Sensoren wird durch die in Kapitel 7 beschriebene Ressourcen-Management-Plattform ermöglicht. Sensoren können allgemein als Datenquelle betrachtet werden, wobei Daten in regelmäßigen Abständen aktualisiert werden wodurch ein Stream entsteht. Die entwickelten Konzepte der RMP können daher vollständig für das Anschließen von Sensoren

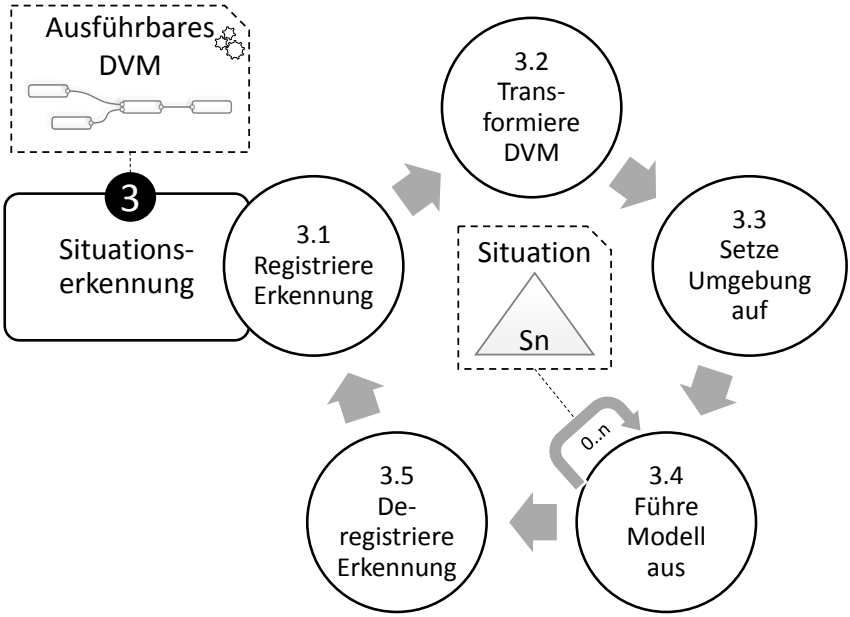

Abbildung 9.7: Methode zur Ausführung der Situationserkennung (basierend auf [HWS+16])

verwendet werden. Dabei werden wie in Kapitel 7 beschrieben spezielle Adapter für verschiedenartige Sensoren automatisch mit dem TOSCA-Standard bereitgestellt, die an die Schnittstellen der Sensoren anbinden, die Daten extrahieren, und sie über die RMP zur Verfügung stellen [HBS+16].

Dadurch können sowohl Sensoren als auch alle anderen zur Erkennung von Situationen benötigten Datenquellen mittels der RMP angeschlossen werden. Nachdem die Sensoren angeschlossen sind sowie das DVM zur Situationserkennung erstellt wurde, kann es ausgeführt werden. Dies wird durch die in Abbildung 9.7 dargestellte Methode umgesetzt. Im ersten Schritt der Methode wird die Erkennung der Situation angestoßen (3.1), was als *Registrierung* der Situation bezeichnet werden kann. Danach wird die Trans-

formation des Modells in eine ausführbare Repräsentation durchgeführt (3.2), die Ausführungsumgebung wird automatisch aufgesetzt (3.3) und das transformierte Modell wiederholt in einem definierten Zeitabstand, zum Beispiel jede Sekunde, ausgeführt (3.4), um Situationen zu erkennen. Bei jeder Ausführung werden Sensordaten aus der RMP extrahiert, mit den modellierten Bedingungen verglichen und die Ergebnisse aggregiert. Tritt die Situation auf, wird ein vorher definierter Service benachrichtigt. Dies wird so lange wiederholt, bis die Situation wieder deregistriert wird (3.5), woraufhin die Ausführung gestoppt wird.

Die mittels dem DVM modellierte Situation kann dabei auf verschiedene Repräsentationen (BPEL, BPMN, Node-RED) abgebildet werden sowie in heterogenen Ausführungsumgebungen ausgeführt werden.

9.2.1.2 Quantitative Evaluation – Laufzeitmessungen

Insbesondere die Erkennung von Situationen ist sehr laufzeitkritisch. Aus diesem Grund wurden Messungen der Ausführung des Modells zur Erkennung vorgenommen. Die Ergebnisse der Messungen für die oben beschriebenen Methodenschritte sind in Tabelle 9.4 zu sehen.

Um die Laufzeitmessungen durchzuführen wurde ein Ubuntu Image basierend auf der Virtualisierungsumgebung Openstack[1] verwendet. Dieses Image hat 8 GB RAM sowie 8 Intel(R) Xeon(R) CPU E5-2630 0 @ 2.30GHz CPUs, um die Messungen durchzuführen. Für die Ausführung wurde die sehr leichtgewichtige Fluss-Engine Node-RED verwendet. Bei den Messungen wurden die Laufzeit für die Transformation des Modells, dessen Platzierung in einer passenden Ausführungsumgebung sowie dessen Ausführung separat betrachtet. Die Ausführungsumgebung selbst wurde bereits aufgesetzt.

Für die Messungen wurde das in Abbildung 9.5 dargestellte Modell verwendet, mit dem ein Rechner fernüberwacht wird. Dieses Modell enthält 8 Knoten, die transformiert und ausgeführt werden müssen.

Tabelle 9.4 zeigt die Messungen für ein einzelnes dieser Modelle. Das Ziel der Messungen ist es zu zeigen, dass die Erkennung von Situationen mit den

[1] http://www.openstack.org/

Tabelle 9.4: Laufzeitmessungen des Prototyps (basierend auf [HWS+15])

Messung	Transformation	Platzierung	Ausführung
1	219 ms	141 ms	6 ms
2	219 ms	126 ms	6 ms
3	234 ms	125 ms	5 ms
4	203 ms	141 ms	5 ms
5	204 ms	140 ms	6 ms
⊘	215,8 ms	134,6 ms	5,6 ms

vorgestellten Konzepten effizient, also im Millisekundenbereich, durchgeführt werden kann. Bisherige Ansätze zur Situationserkennung, wie zum Beispiel der von Dargie et al. [DEM+13] vorgestellte Ansatz, der basierend auf OWL 2 Reasoning arbeitet, erreichen lediglich eine Laufzeit von minimal 800ms für die Erkennung. Dabei ist die von Dargie et al. erkannte Situation sehr ähnlich zur Situation aus Abbildung 9.5, auf der unsere Messungen basieren. Wie in Tabelle 9.4 zu sehen ist, beträgt die Laufzeit zur Situationserkennung lediglich 6 ms, was die Effizienzanforderungen des Projektes SitOPT mehr als erfüllt.

Zusätzlich wurde ein Lasttest durchgeführt, um zu überprüfen wie viele Situationen gleichzeitig transformiert und parallel in einer einzelnen Umgebung ausgeführt werden können. Die Ergebnisse dieses Lasttests sind in Tabelle 9.5 zu sehen.

Tabelle 9.5: Lasttest des Prototyps (basierend auf [HWS+15])

# Modelle	Transformation ⊘	Deployment ⊘	Parallele Laufzeit ⊘
1	215,8 ms	134,6 ms	5,6 ms
2	424,4 ms	209,2 ms	37,6 ms
5	1093 ms	350 ms	176,4 ms
10	2475 ms	659,2 ms	404,4 ms

Hier ist zu sehen, dass sich die Laufzeit bei der Ausführung mittels Node-RED bei der Ausführung von mehreren Modellen parallel stark erhöht. Bei der parallelen Ausführung von zwei Modellen steigt die Laufzeit bereits auf 38 ms an statt wie erwartet bei ca. 6 ms zu bleiben. Bei 10 parallel

ausgeführten Modellen beträgt die Gesamtlaufzeit sogar 404 ms. Daher ist zu vermuten, dass die Ausführungsumgebung einen signifikanten Overhead bei der parallelen Ausführung mehrerer Modelle erzeugt. Bei genauerer Untersuchung der Ausführungsumgebung wurde festgestellt, dass kein echter Parallelismus stattfindet sondern alle Modelle in einem Thread ausgeführt werden. Dabei wird eine Pseudoparallelität durch die sequentielle, abwechselnde Ausführung von Knoten aus den verschiedenen Modellen umgesetzt. Das interne Scheduling von Node-RED sorgt zusätzlich für längere Wartezeiten zwischen der Ausführung von verschiedenen Knoten. Hierdurch können Rückschlüsse auf die Anforderungsdefinition gezogen werden. Ist Parallelismus von hoher Wichtigkeit, sollte die Ausführung des Modells nicht wie bei Node-RED quasi-parallel durchgeführt werden, sondern auf echtparallele Verarbeitung geachtet werden.

Ein Ansatz zu einer echtparallelen Verarbeitung von DVMs und damit paralleler Situationserkennung, ist Complex Event Processing. Im Rahmen der studentischen Masterarbeit von Franco da Silva [Fra15] wurden im ersten Schritt mehrere CEP-Ausführungsumgebungen verglichen. Diese umfassen CEP Esper [Esp], Odysseus [Old], und den Complex Event Processor von WSO2 [WSO]. Insbesondere die Skalierbarkeit stand im Vordergrund, da die bisher verwendete Ausführungsumgebung Node-RED damit, wie die Messungen zeigen, Probleme hat. Die Wahl fiel auf CEP Esper, da diese Ausführungsengine den anderen bezüglich Mächtigkeit und Effizienz überlegen ist. Details der Auswahl sind der Masterarbeit [Fra15] zu entnehmen.

Nach der Auswahl der am besten geeigneten Ausführungsumgebung wurde eine automatische Transformation des DVM bzw. DVM$^+$ auf CEP-Ausdrücke realisiert, welche für eine effektive Situationserkennung verwendet werden. Aufgrund des in dieser Dissertation entwickelten Transformationskonzeptes konnte die Abbildung auf ausführbare DVMs leicht auf die neue Ausführungsumgebung CEP Esper umgestellt werden.

Die Resultate wurden durch Messungen der Laufzeit evaluiert und Verbesserungen im Vergleich zum bisherigen Ansatz konnten gezeigt werden. Die detaillierten Ergebnisse sind in der Veröffentlichung [FHWM16] von Franco da Silva festgehalten.

9.2.1.3 Situationserkennung – Verwandte Arbeiten

Kontextmodelle [GBH+05] sind ein verbreitetes Konzept als Grundlage für das Erkennen von Situationen. Diese Modelle bilden bestimmte Aspekte einer Umgebung so detailliert wie möglich ab und dienen als Basis für verschiedenartige kontextbezogene Applikationen und Systeme. Während Kontextmodelle die Umgebung beschreiben, ist für die Erkennung von Situationen eine Verarbeitung bzw. Aggregation der Kontextdaten notwendig. Aus diesem Grund basiert die in diesem Abschnitt vorgestellte Situationserkennung auf Kontextmodellen und bietet darüber hinaus eine Möglichkeit Situationen einfach zu modellieren und zu erkennen. Die Situationserkennung geschieht mittels des in dieser Dissertation vorgestellten DVM. Dabei können die Situation Templates von Zweigle et al. [ZHKL09; Zwe11] und Häusermann [HHL+10] auf das DVM abgebildet werden. Anschließend kann dieses transformiert und ausgeführt werden. Die Ausführung des DVM entspricht somit der eigentlichen Situationserkennung.

Andere Systeme zur Erkennung von Situationen nutzen ontologiebasierte oder regelbasierte Ansätze [WZGP04]. Im Vergleich zu dem in diesem Abschnitt vorgestellten Ansatz sind viele kontextbezogene Systeme zur Situationserkennung hingegen entweder eingeschränkt auf bestimmte geographische Bereiche oder unterstützen nur spezifische Anwendungsfälle [BMK+00]. In dem in diesem Abschnitt vorgestellten Ansatz werden hingegen keine Einschränkungen geographischer Bereiche unterstützt, da ein globales Kontextmodell zugrundeliegt. Zusätzlich ist dieser Ansatz nicht auf bestimmte Anwendungsfälle begrenzt. Es können jegliche Situationen modelliert und basierend auf verfügbaren Kontextinformationen erkannt werden. Des Weiteren zeigen Messungen der Situationserkennung, dass insbesondere ontologiebasierte Ansätze nicht den oftmals hohen Effizienzanforderungen zur Erkennung von kritischen Situationen entsprechen.

Neben diesen ontologiebasierten Ansätzen existieren Ansätze, die basierend auf Mustererkennung, beispielsweise mittels Machine Learning [ASRH13] oder, wie bereits erwähnt basierend auf Reasoning von Ontologien [DEM+13] Situationen erkennen. Im Gegensatz zu diesen Ansätzen

basiert der von mir entwickelte Ansatz auf der Ausführung von Datenflüssen. Um diesen weiter zu verbessern können Complex Event Processing (CEP) [BK09]-Systeme verwendet werden.

9.2.1.4 Zusammenfassung und Fazit der Integration in SitOPT

SitOPT stellt einen Spezialfall der Datenverarbeitung dar. Hierbei stehen die Geschwindigkeit sowie die Heterogenität der Daten im Vordergrund. Zur Erkennung von Situationen ist es notwendig, Daten zu extrahieren, die Daten nach Prädikaten zu filtern, und letztendlich zu aggregieren. Die Situationserkennung kann durch das in Kapitel 4 vorgestellte DVM und den entwickelten Modelltransformationen realisiert werden. Aufgrund dessen, dass bei der Situationserkennung, insbesondere im Projekt SitOPT, oftmals die Effizienz eine große Rolle spielt und Faktoren wie Robusheit weniger wichtig sind, muss die Anforderungsdefinition für derartige Projekte entsprechend angepasst werden. Die erfolgreiche Durchführung dieses Projekts zeigt die Anwendbarkeit der Beiträge, um eine Situationserkennung zu ermöglichen.

9.2.2 Integration im Bereich Industrie 4.0

Zusätzlich wurden die Konzepte dieser Dissertation in Teilen in weitere Projekte im Bereich Industrie 4.0 integriert. Dies wird in den folgenden Abschnitten beschrieben.

9.2.2.1 Kontextabhängige Datenbereitstellung in der Produktion

In [HHM17a; HHM17b] beschreiben Hoos et al. wie Kontext dabei helfen kann, Arbeitern in Fabriken die richtige Information zum richtigen Zeitpunkt am richtigen Ort zur Verfügung zu stellen. Beispielsweise benötigt ein Arbeiter bei einem Fehlerfall an seiner Station Informationen passend zu seiner Position, dem momentan gefertigten Teil, dessen Fähigkeiten, etc. Diese Informationen sollen helfen, den Fehler zu beheben. Oftmals werden dazu Daten aus verschiedenartigen Datenquellen benötigt. In der Regel umfassen diese CAD-Modelle oder Maschinendokumentationen. Insbesondere

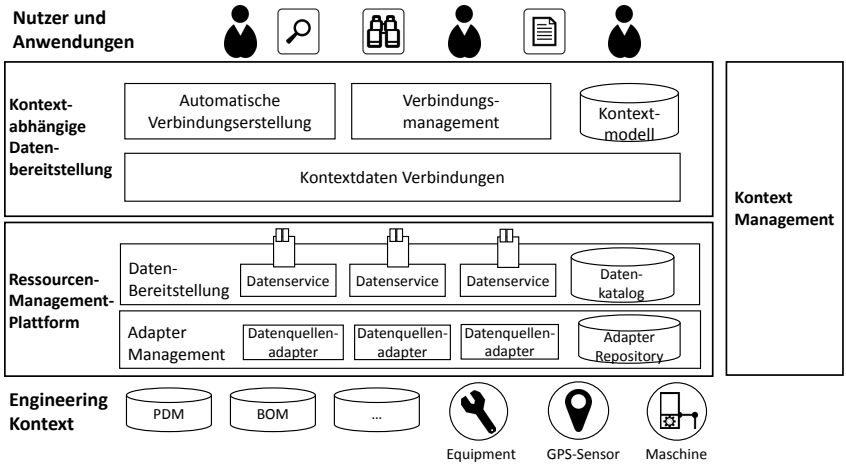

Abbildung 9.8: Integration der RMP in Architektur zur kontextabhängigen Datenbereitstellung (basierend auf [HHM17a])

in Produktionsumgebungen sind die Datenquellen jedoch sehr dynamisch. Durch Neuwentwicklungen und Simulationen entstehen stets neue Daten, Datenquellen können durch das Anschließen neuer Geräte oder Produktionsmaschinen hinzukommen oder bei deren Ausfall wegfallen.

Um die von Hoos et al. angestrebten Ziele zu erreichen, wurde die RMP integriert. Diese Integration ist in Abbildung 9.8 dargestellt. Dabei ist zu sehen, dass auf oberster Ebene dieser Lösungsarchitektur Anwendungen liegen, über die Daten kontextbezogen bereitgestellt werden. Diese Anwendungen können zum Beispiel Apps auf mobilen Endgeräten darstellen. Welche Datenquellen zu einem bestimmten Kontext angezeigt werden sollten, wird in den Komponenten der Kontextabhängigen Datenbereitstellung bestimmt.

Die Bereitstellung der Kontextdaten wird von der darunterliegenden Schicht übernommen, die durch die RMP realisiert ist. Die RMP dient der Bereitstellung der Daten und zur Abstraktion von den entsprechenden Quellen. Insbesondere die Dynamik der Datenquellen stellt in Produktionsumgebungen eine große Herausforderung dar. Stetig hinzukommende oder wegfallende Datenquellen müssen beachtet werden. Sobald neue Daten

entstehen, sollten sie zur Verknüpfung mit Kontextinformationen automatisch zur Verfügung gestellt werden. Die Datenbereitstellung wird durch die RMP ermöglicht. Die Umgebung wird überwacht und es werden beim Erscheinen neuer Datenquellen Adapter provisioniert, die deren Daten extrahieren und über die Schnittstellen der RMP den darüber liegenden Schichten zur Verfügung stellen. Umgekehrt wird beim Wegfall von Datenquellen der entsprechende Adapter deprovisioniert und die Kontextabhängige Datenbereitstellung wird benachrichtigt, woraufhin entsprechende Verbindungen zum Kontextmodell gelöscht werden.

Durch die Integration der RMP in die Konzepte von Hoos et al. zeigt sich, dass die RMP vielen Anwendungsbereichen, beispielsweise im Bereich Industrie 4.0, dienen kann. Zusätzlich wurden Teile der Konzepte in einem weiteren Projekt, der sogenannten Sozialen Fabrik, integriert. Dies wird im Folgenden beschrieben.

9.2.2.2 Die Soziale Fabrik

In [KHW+17] stellen Kassner et al. die *Soziale Fabrik* vor. Dabei handelt es sich um ein soziales Netzwerk, das Menschen und Maschinen in der Produktion verknüpft. Die Kommunikation findet ausschließlich über natürliche Sprache in Form von Text statt. Um Texteingaben durch Maschinen interpretierbar zu machen, werden automatische Textanalysemechanismen der natürlichen Sprachverarbeitung [FS06], wie Tokenisierung oder Part-of-Speech-Tagging eingesetzt.

Die Konzepte dieser Dissertation finden auch im Rahmen der Sozialen Fabrik Anwendung. Hierfür wurde die RMP integriert, um die Maschinen der Produktion als Datenquellen anzubinden sowie die beschriebene Erkennung von Situationen, mittels DVM-Verarbeitung.

Wie die Integration durchgeführt wurde, ist in Abbildung 9.9 dargestellt. Das soziale Netzwerk der Sozialen Fabrik besteht aus einem Frontend, das die Menschen und Maschinen der physischen Welt mittels Nutzerprofilen repräsentiert sowie aus einer Middleware, die für die Textanalyse wie auch die Benachrichtigung von Menschen bei Fehlersituationen verantwortlich ist.

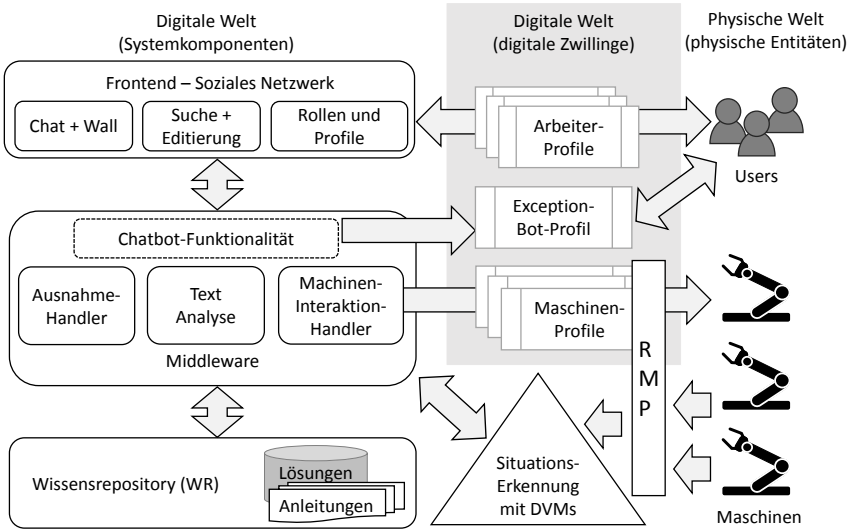

Abbildung 9.9: Architektur der Sozialen Fabrik (basierend auf [KHW+17])

Wissen über bisher aufgetretene Situationen sowie zugehörige Lösungsvorschläge sind in einem Wissensrepository (vgl. Abbildung 9.9) abgespeichert. Die Integration der Konzepte dieser Dissertation betrifft hierbei zwei Aspekte: 1) das automatisierte Anschließen von Maschinen, um deren Daten auszulesen und um die Kommunikationen mit den Maschinen zu ermöglichen, und 2) die Erkennung von Fehlersituationen in der Produktion, die zur Benachrichtigung von Menschen im sozialen Netzwerk führt.

Für das Anschließen der Maschinen werden die Konzepte der RMP analog zu dem in Hoos et al. [HHM17a; HHM17b] beschriebenen Ansatz integriert. Dabei werden die Maschinen jedoch manuell registriert, und es existieren keine Crawler, die die Umgebung kontinuierlich auf neu hinzugekommene Maschinen überprüfen. Aufgrund dessen, dass Maschinen in der Produktion selten ersetzt, ausgetauscht oder neue Maschinen angeschafft werden, ist eine manuelle Registrierung an der RMP ausreichend. Ein Ausfall von Maschinen wird bereits in den Produktionsprozessen detektiert, und es werden

daraufhin Maßnahmen der Instandhaltung angestoßen, um das Problem zu beheben. Beim Anschließen der Maschinen passieren zwei Dinge: erstens werden deren Daten der Situationserkennung zur Detektion von Fehlerfällen über die Schnittstelle der RMP zur Verfügung gestellt. Zweitens wird es außerdem ermöglicht, auf die Maschine zuzugreifen, um diese steuern zu können. Die Steuerung der Maschinen stellt eine neue Funktionalität der RMP dar, da die RMP lediglich dazu genutzt wurde, Daten aus Datenquellen zu extrahieren. Hierfür werden spezielle "Steuerungsadapter" provisioniert, die zu den Maschinen verbinden und grundlegende Operationen wie anschalten und ausschalten durchführen. Der Zugriff auf die Maschinen geschieht ebenfalls über die Schnittstelle der RMP.

Des Weiteren stellt die RMP die Daten der Maschinen zur Erkennung von Fehlersituationen zur Verfügung. Diese Erkennung wird wie in Abschnitt 9.2.1 beschrieben durchgeführt. Nach der Modellierung und Ausführung des DVM zur Situationserkennung wird, im Falle eines Fehlers an einer Maschine, die für diese verantwortliche Person automatisch benachrichtigt.

9.3 Zusammenfassung

Die Evaluation der Konzepte fand qualitativ durch eine Bewertung einer Fachjury sowie quantitativ durch die Integration dieser in verschiedenen Projekten und Ansätzen statt. Es war das Ziel zu zeigen, dass diese einen Beitrag in realen Anwendungen leisten können.

Das erste Projekt, in der die Ansätze integriert wurden, ist das Forschungsprojekt SitOPT, das darauf abzielt, situationsabhängige Workflows zu schaffen. Es kamen die Konzepte insbesondere bei der Erkennung von Situationen zum Einsatz, wobei hohe Effizienzanforderungen gefordert waren. Die Datenquellen umfassen vor allem Kontextdaten bereitgestellt von Sensoren. Die Umgebungen sind sehr dynamisch, was bedeutet, dass Datenquellen stetig hinzukommen und wegfallen können. Um mit dieser Dynamik umgehen zu können, wurden Kontextdatenquellen an die RMP angeschlossen. Die RMP bietet nicht nur deren automatische Anbindung sondern darüber hinaus eine

Überwachung der Datenquellen, um deren Wegfall zu bemerken.

Basierend auf den angebundenen Kontextdatenquellen kann anschließend ein DVM erstellt werden, das definiert, welche Daten extrahiert werden müssen sowie wie die Daten gefiltert und aggregiert werden, um Situationen zu erkennen. Dabei werden die Anforderungen so definiert, dass Effizienz am Wichtigsten ist, Robustheit hingegen ist von geringerer Priorität. Durch Ausführung dieses Modells können Situationen zeitnah erkannt werden, was durch durchgeführte Messungen belegt werden kann.

Neben der Integration im Projekt SitOPT wurde die RMP in ein Projekt zur kontextabhängigen Datenbereitstellung in Produktionsumgebungen sowie in die Soziale Fabrik integriert, um Datenquellen dynamisch anzubinden, sowie Daten kontextabhängig zur Verfügung zu stellen.

Allgemein gilt, dass die Eigenschaften der RMP immer dann vonnöten sind, wenn Datenquellen dynamisch wegfallen oder hinzukommen. Dies ist vor allem in den Domänen Internet der Dinge oder Industrie 4.0 der Fall. Durch die Integration der RMP in Projekte dieser Domänen könnte deren Anwendbarkeit belegt werden.

Des Weiteren wurde gezeigt, dass DVMs bzw DVM^+-Modelle allgemein zur Situationserkennung verwendet werden können. Notwendige Effizienzanforderungen werden erfüllt. Der in dieser Dissertation vorgestellten Modellierungs-, Transformations- und Provisionierungsansatz ermöglicht vielfältige flexible und dynamische Einsatzformen. Das Gesamtkonzept ermöglicht darüber hinaus der gestiegenen Komplexität derartiger Anwendungen besser zu begegnen. Das zeigen die Anwendungsbeispiele und die dort bewerteten Prototypen FlexMash, RMP, etc.

Kapitel 10

Zusammenfassung und zukünftige Arbeiten

Trotz des Fortschritts in den Bereichen Datenverarbeitung, Datenanalyse, Cloud Computing und paralleler Datenverarbeitung, stellt die Verarbeitung großer Datenmengen insbesondere für Domänennutzer eine große Herausforderung dar. Bisherige Ansätze sind oftmals sehr statisch, das heißt, dass die Anforderungen der Nutzer bezogen auf spezielle Anwendungsfälle oftmals nicht erfüllt werden können. Ein System, das alle Anforderungen erfüllen kann, ist ebenfalls nicht realistisch. Insbesondere nichtfunktionale Anforderungen, wie z.B. Kosten, Sicherheit, Robustheit, finden in bisherigen Ansätzen kaum Beachtung.

Aus diesem Grund wird in dieser Dissertation ein holistisches Konzept vorgestellt, mit dem Daten flexibel und anforderungsgetrieben verarbeitet werden können. Die vorgestellte Gesamtlösung basiert dabei vollständig auf etablierten Standards oder de-facto Standards (z.B. BPEL), um die Anwendbarkeit sowie die Zukunftssicherheit der Lösung zu gewährleisten. Bei den Konzepten steht insbesondere die Anwendbarkeit durch Domänennutzer

außerhalb des IT-Bereichs im Vordergrund. Um Domänennutzer bestmöglich zu unterstützen wird die Ausführungsumgebung dynamisch vollautomatisiert aufgebaut. Dabei werden die spezifischen Anforderungen der Nutzer berücksichtigt. Hierfür wurden in dieser Dissertation Konzepte geschaffen, die sich in fünf Beiträge aufteilen lassen:

1. Abstrakte Modellierung der Datenverarbeitung durch Domänennutzer, wobei die Anwender durch Konzepte wie einer Vorschlagsgenerierung zusätzlich unterstützt werden.
2. Dynamische Auswahl der Ausführungsumgebung basierend auf der Definition nichtfunktionaler Anforderungen.
3. Automatische Provisionierung der Ausführungsumgebung mittels des TOSCA-Standards.
4. Anschließen der Datenquellen durch die Ressourcen-Management-Plattform.
5. Workflowbasierte Ausführung der servicebasierten Datenverarbeitung sowie deren Optimierung.

Im ersten Beitrag wurden Anforderungen aufgestellt, die ein Modell zur Definition der Datenverarbeitung bzw. des Datenflusses erfüllen muss. Auf Basis dieser Anforderungen wurde ein Modell, genannt DVM, basierend auf dem Pipes-and-Filters-Muster definiert, das ein etabliertes Mittel zur Modellierung von Kontroll- und Datenfluss darstellt. Das grundlegende Pipes-and-Filters-Modell wurde dahingehend erweitert, um eine Abstraktion von Details durch mehrere Hierarchieebenen zu ermöglichen. Darüber hinaus wurde die DVM-Modellierung durch eine Vorschlagsgenerierung für Domänennutzer vereinfacht. Um eine Abstraktion von technischen Details der zu verwendenden Datenverarbeitungsoperationen zu ermöglichen, wurden Modellierungsmuster vorgestellt. Modellierungsmuster sind oft wiederkehrende Folgen von Operationen, wie Filter, Datenaggregation, oder Analysealgorithmen. Diese Modellierungsmuster lassen sich auf verschiedene Implementierungen abbilden, die jeweils unterschiedliche nichtfunktionale Anforderungen erfüllen.

Im zweiten Beitrag wird es ermöglicht, Anforderungen an die Datenverarbeitung zu definieren, wodurch das erweiterte Modell DVM$^+$ entsteht. Dabei können Domänennutzer Anforderungen abstrakt mittels eines Punktevergabesystems verteilen, um die Anforderungen zu priorisieren. Anwender mit Expertenwissen können die Anforderungen darüber hinaus exakt mittels eines policy-basierten Formats definieren. Anschließend wird das DVM$^+$ in ein ausführbares Modell, genannt DPM, überführt. Dieses ausführbare Modell ist workflowbasiert und ermöglicht die Datenverarbeitung durch eine Orchestrierung den Operatoren bzw. Modellierungsmustern zugrundeliegenden Services.

Nachdem die passenden Implementierung zur Ausführung des DPM aufgefunden wurden, wird im nächsten Schritt, im dritten Beitrag, die Ausführungsumgebung automatisch bereitgestellt. Eine dynamische Bereitstellung spart Kosten, da Ressourcen nur dann in Anspruch genommen werden, wenn sie auch benötigt werden. Um die Provsionierung der Ausführungsumgebung standardbasiert durchzuführen kommt OASIS TOSCA zum Einsatz. TOSCA ist zu diesem Zeitpunkt der einzige Standard zur Provisionierung und zum Management von Anwendung und wird daher gegenüber anderen Technologien wie Docker oder Vagrant bevorzugt.

Bei der automatischen Provisionierung wird zuerst eine TOSCA Topologie der im vorigen Schritt aufgefundenen Implementierungen erstellt. Diese Topologie enthält jegliche anwendungsspezifischen Komponenten, jedoch fehlen Plattform- und Infrastrukturkomponenten (Web Server, Virtuelle Maschinen, etc.), die für deren Ausführung benötigt werden. Diese Komponenten werden mittels einer automatisierten Topologievervollständigung hinzugefügt. Anschließend kann die vollständige Topologie in einer TOSCA-Ausführungsumgebung zur automatischen Provisionierung der Implementierungen verwendet werden.

Nach dem automatischen Aufsetzen der Ausführungsumgebung werden die Datenquellen angeschlossen, die die zu verarbeitenden Daten bereitstellen. Dies geschieht in Beitrag vier durch die Ressourcen-Management-Plattform (RMP), eine Middlewarekomponente zum automatischen Anschließen von Datenquellen sowie zum uniformen Datenzugriff. Die RMP abstra-

hiert von den Datenquellen und vereinfacht den Datenzugriff. Darüber hinaus ermöglicht die RMP Datentransformation und -bereinigung zur Steigerung der Datenqualität. Die RMP nutzt ebenfalls den TOSCA-Standard, um Datenquellen automatisch anzuschließen. Hierfür werden Adapter auf einer Plattform provisioniert, die in der Lage sind, sich zu den Datenquellen zu verbinden, Daten zu extrahieren sowie an die RMP weiterzureichen. Die Daten können anschließend über eine REST-Schnittstelle abgefragt werden.

In Beitrag fünf findet die Datenverarbeitung statt. Dazu muss lediglich das transformierte DPM in einer passenden Umgebung ausgeführt werden. Dabei handelt es sich um einen Workflow, der Services aufruft, die die Daten verarbeiten. Die Reihenfolge der aufzurufenden Services ist im DPM definiert. Die Datenverarbeitung kann somit automatisiert durchgeführt werden.

Darüber hinaus wurde untersucht, wie sich eine Verteilung der Datenverarbeitung auf deren Effizienz auswirkt. Hierfür wurden die beiden Datenverarbeitungsoperationen Filterung und Aggregation einerseits als In-Service-Implementierung implementiert und ausgeführt und andererseits als verteilte Implementierung mittels Apache Spark. Hierbei fiel auf, dass eine Verteilung der Datenverarbeitung große Vorteile bringen kann. Jedoch muss für jede Datenoperation einzeln abgewägt werden, ob diese Operation verteilt ausgeführt werden sollte. Dies hängt maßgeblich von der Datengröße, aber auch von der Komplexität der Datenoperation ab bzw. davon, wie gut sich diese verteilt ausführen lässt. Auch die optimale Größe des Rechenclusters variiert stark mit den auszuführenden Datenoperationen bzw. der Menge der zu verarbeiteten Daten. Die Untersuchung im Rahmen diesen Beitrags geben einen Hinweis, wie Datenmenge und optimale Clustergröße zusammenhängen.

Zusätzlich zu diesen Untersuchungen wurde eine Optimierung des vorgestellten Gesamtansatzes geschaffen – eine Teilausführung des DVM während der Modellierungszeit. Hierbei wurde ein Konzept geschaffen, mit dem das DVM bereits während der Modellierung teilweise ausgeführt werden kann. Sobald eine Datenverarbeitungsoperation vollständig modelliert wurde, wird im Hintergrund eine passende Implementierung für deren Modellierungsmuster, basierend auf den nichtfunktionalen Anforderungen

aufgefunden, provisioniert und ausgeführt. Basiert die Datenverarbeitungsoperation auf einem vorigen Ergebnis, wird dieses Zwischenergebnis aus einem Zwischenspeicher extrahiert. Dieser Zwischenspeicher enthält die Ergebnisse aller partiell ausgeführten Datenoperationen. Dies ermöglicht einerseits eine schnellere Ausführung, da große Teile des Modells bereits im Hintergrund verarbeitet wurden. Des Weiteren ermöglicht dies die Einsichtnahme des Benutzers in die Zwischenergebnisse.

Die Beiträge wurden qualitativ durch eine Bewertung einer Fachjury und quantitativ durch die Integration in aktuelle Forschungsprojekte evaluiert, wodurch deren Anwendbarkeit weitestgehend realitätsnah bestätigt werden konnte. Diese Projekte umfassen das Forschungsprojekt SitOPT sowie Industrieprojekte des Bereichs Industrie 4.0.

10.1 Bewertung der Dissertation

Die Evaluation der Beiträge dieser Dissertation hat gezeigt, dass diese einen hohen Mehrwert für die Datenverarbeitung durch Domänennutzer darstellen können. Insbesondere die Berücksichtigung verschiedener funktionaler und nichtfunktionaler Anforderungen sowie der dynamische Aufbau der Ausführungsumgebung wurden in bisherigen Ansätzen nicht beachtet und stellen somit eine Forschungslücke dar, die von dieser Dissertation erfüllt wird.

In bisherigen, etablierten Werkzeugen und Ansätzen zur Datenverarbeitung müssen nichtfunktionale Anforderungen durch die manuelle Auswahl passender Implementierungen erfüllt werden. Ändern sich die Anforderungen, beispielsweise da sich der Anwendungsfall ändert, müssen die Implementierungen neu ausgewählt, konfiguriert und aufgesetzt werden. Diese Vorgehensweise führt zu hohen Kosten, die durch Auswahl und Provisionierung der zur Ausführung benötigten Softwarekomponenten entstehen. Durch die Beiträge dieser Dissertation kann dies automatisiert werden. Des Weiteren muss der Domänennutzer, der die Datenverarbeitung ausführen möchte, kein tiefes technisches Wissen über die Implementierungen besitzen bzw.

wie diese die nichtfunktionalen Anforderung spezieller Anwendungsfälle erfüllen können. Nur die wenigsten der Domänennutzer kennen beispielsweise Implementierungen bestimmter Verschlüsselungstechniken, die für die gewünschte Sicherheit benötigt werden, oder wissen, wie eine Verteilung der Datenverarbeitung zur Effizienzsteigerung umgesetzt werden kann.

Durch ein dynamisches, automatisches Aufsetzen der Ausführungsumgebung können darüber hinaus Kosten gespart werden. Geht man davon aus, dass Implementierungen für jegliche Kombinationen an funktionalen und nichtfunktionalen Anforderungen, unter Umständen sogar in verschiedenen Versionen, bereits aufgesetzt und lauffähig sind, führt dies zu hohen Kosten. Zwar bieten viele Software-as-a-Service-Anbieter bereits Implementierungen einiger Dienste an, jedoch kann eine volle Abdeckung der möglichen Funktionalität bei weitem nicht gewährleistet werden. Somit ist ein eigenständiges Aufsetzen, beispielsweise unter Hinzunahme der Ressourcen eines Platform-as-a-Service-Anbieters sinnvoll und kann zu einer hohen Kosteneinsparung führen, was durch die vorgestellten Beiträge zur Software-Provisionierung erreicht werden konnte.

Die qualitative Evaluation der Ergebnisse durch die Teilnahme an der Rapid Mashup Challenge sowie die quantitative Evaluation durch eine Integration in aktuelle Forschungsprojekte zeigt die Anwendbarkeit der Konzepte. Durchgeführte Messungen zeigen die Effizienz dieser integrierten Prototypen. Darüber hinaus dienen die entstandenen Beiträge als Grundlage zweier Forschungsarbeiten, die im folgenden Abschnitt beschrieben werden.

10.2 Zukünftige Arbeiten

10.2.1 Interaktive Datenverarbeitung und visuelle Analyse

Das erste zukünftige Forschungsgebiet beschäftigt sich mit der interaktiven Datenverarbeitung und einer visuellen Analyse von Daten. Dabei ist es das Ziel, die Datenverarbeitung interaktiv zu gestalten. Im Rahmen dieser Dissertation wurden Konzepte geschaffen, bei denen nach einer ursprünglichen Modellierung der Datenverarbeitung diese vollautomatisiert ausgeführt

wird. In Zukunft soll es darüber hinaus möglich sein, interaktiv in die Datenverarbeitung einzugreifen und diese durch gezielte Entscheidungen zu beeinflussen. Dabei stellt der Domänennutzer die zentrale Instanz dar. Insbesondere ist es von großer Wichtigkeit, diesem Daten in einem verständlichen, übersichtlichen Format zu visualisieren. Auf Basis dieser Daten können anschließend manuell oder semi-automatisch Daten selektiert, gefiltert, oder aggregiert werden.

Um diese Ziele zu erreichen, wurden bereits erste Vorarbeiten geschaffen. Diese umfassen einerseits die partielle Ausführung des DVM wie in Abschnitt 8.2.2 beschrieben. Dadurch kann ein erster Schritt dahingehend ermöglicht werden, Zwischenergebnisse der Datenverarbeitung anzuzeigen. Dies ist notwendig, um den Domänenexperten in die Datenverarbeitung zu integrieren. Des Weiteren wurde von Behringer et al. [BHM17] ein Prozess basierend auf dem KDD-Prozess vorgestellt, mit dem Daten interaktiv verarbeitet werden können. Diese beiden Vorarbeiten stellen erste Schritte für eine vollständige Integration vom Domänennutzer in den Datenverarbeitungs- und Analyseprozess dar.

10.2.2 Dynamische Platzierung von Datenverarbeitungsoperationen

Das zweite zukünftige Forschungsgebiet basierend auf den Konzepten dieser Dissertation beschäftigt sich mit einer dynamischen Platzierung von Datenverarbeitungsoperationen auf heterogener Infrastruktur. Im Rahmen dieser Arbeit wurden Services, die bestimmte Datenoperationen implementieren, auf beliebiger Infrastruktur provisioniert. In einer zukünftigen Forschungsarbeit soll untersucht werden, wie die Ressourcen der bestehenden Recheninfrastruktur bestmöglich ausgenutzt werden können. Hierfür wird im ersten Schritt ein digitales Abbild der Recheninfrastruktur erstellt, das unter anderem verfügbare Ressourcen beschreibt. Durch eine Überwachung dieser Infrastruktur wird dies stets aktualisiert, um das digitale Abbild synchron zu halten. Anschließend soll ein Algorithmus entwickelt werden, der Datenverarbeitungsoperationen auf die bestmögliche Infrastruktur provisioniert unter Berücksichtigung der verfügbaren Ressourcen. Dabei muss unter anderem

untersucht werden, wie rechenintensiv diese Operationen sind und wie mit Ausfällen in der Infrastruktur umgegangen werden kann. Um diese Aufgaben zu ermöglichen, soll die in Kapitel 7 vorgestellte RMP weiterentwickelt werden.

Publikationen des Autors

Im Folgenden werden die eigenen Publikationen des Autors dieser Dissertation chronologisch sortiert aufgelistet. Dabei wird zwischen Publikationen als Erstautor und als Co-Autor unterschieden:

Publikationen als Erstautor:

2014

- Hirmer, Pascal; Breitenbücher, Uwe; Binz, Tobias; Leymann, Frank: Automatic Topology Completion of TOSCA-based Cloud Applications. In: Proceedings des CloudCycle14 Workshops auf der 44. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 2014

2015

- Hirmer, Pascal; Mitschang, Bernhard: FlexMash - A Flexible Data Mashup Tool. Demo@International Conference on Web Engineering (ICWE), Rotterdam, Netherlands, 2015
- Hirmer, Pascal; Reimann, Peter; Wieland, Matthias; Mitschang, Bernhard: Extended Techniques for Flexible Modeling and Execution of Data Mashups. In: Proceedings of the 4th International Conference

on Data Management Technologies and Applications (DATA), Colmar, France, 2015

- Hirmer, Pascal: Flexible Modeling and Execution of Data Integration Flows Poster@9th Symposium and Summer School On Service-Oriented Computing (SummerSOC), Crete, Greece, 2015
- Hirmer, Pascal; Wieland, Matthias; Schwarz, Holger; Mitschang, Bernhard; Breitenbücher, Uwe; Leymann, Frank: SitRS - A Situation Recognition Service based on Modeling and Executing Situation Templates. In: Proceedings of the 9th Symposium and Summer School On Service-Oriented Computing (SummerSOC), Crete, Greece, 2015

2016

- Hirmer, Pascal: Flexible Execution and Modeling of Data Processing and Integration Flows. In: Proceedings of the 10th Symposium and Summer School On Service-Oriented Computing, Crete, Greece, 2016
- Hirmer, Pascal; Wieland, Matthias; Breitenbücher, Uwe; Mitschang, Bernhard: Dynamic Ontology-based Sensor Binding. In: Proceedings of the 20th East-European Conference on Advances in Databases and Information Systems, Prague, Czech republic, 2016
- Hirmer, Pascal; Wieland, Matthias; Breitenbücher, Uwe; Mitschang, Bernhard: Automated Sensor Registration, Binding and Sensor Data Provisioning. In: Proceedings of the CAiSE 2016 Forum at the 28th International Conference on Advanced Information Systems Engineering, Ljubljana, Slowenia, 2016
- Hirmer, Pascal; FlexMash 2.0. Demo @16th International Conference on Web Engineering (ICWE2016), Lugano, Switzerland, 2016
- Hirmer, Pascal; Mitschang, Bernhard: FlexMash - Flexible Data Mashups Based on Pattern-based Model Transformation. In: Rapid Mashup Development Tools (Book), 2016
- Hirmer, Pascal; Mitschang, Bernhard: TOSCA4Mashups - Enhanced

Method for On-Demand Data Mashup Provisioning. In: Research and Development Journal, 2016

- Hirmer, Pascal; Wieland, Matthias; Schwarz, Holger; Mitschang, Bernhard; Breitenbücher, Uwe; Gómez Sáez, Santiago; Leymann, Frank: Situation Recognition and Handling based on Executing Situation Templates and Situation-aware Workflows, In: Computing Journal, 2016
- Hirmer, Pascal; Franco da Silva, Ana Cristina; Wieland, Matthias; Breitenbücher, Uwe; Képes, Kálmán; Mitschang, Bernhard: Automating the Provisioning and Configuration of Devices in the Internet of Things, In: Complex Systems Informatics and Modeling Quarterly, 2016

2017

- Hirmer, Pascal: Effizienz-Optimierung daten-intensiver Data Mashups am Beispiel von Map-Reduce, In: Proceedings der Datenbanksysteme für Business, Technologie und Web (BTW), 17. Fachtagung des GI-Fachbereichs, Workshopband, 2017
- Hirmer, Pascal; Behringer, Michael; Mitschang, Bernhard: Partial Execution of Mashup Plans During Modeling Time. In Proceedings of the 11th Symposium and Summer School On Service-Oriented Computing (SummerSOC), 2017
- Hirmer, Pascal; Behringer, Michael: FlexMash 2.0 – Flexible Modeling and Execution of Data Mashups. In: Rapid Mashup Development Tools - Second International Rapid Mashup Challenge, RMC 2016, Lugano, Switzerland, June 6, 2016, Revised Selected Papers. Communications in Computer and Information Science, 2017

Publikationen als Co-Autor:

2015

- Breitenbücher, Uwe; Hirmer, Pascal; Képes, Kálmán; Kopp, Oliver; Leymann, Frank; Wieland, Matthias: A Situation-Aware Workflow Modelling Extension. In: Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services (iiWAS 2015). S. 478-484. ACM, 2015

2016

- Mormul, Mathias; Hirmer, Pascal; Wieland, Matthias; Mitschang, Bernhard: Situation Model as Interface between Situation Recognition and Situation-Aware Applications. In: Research and Development, 2016
- Wieland, Matthias; Hirmer, Pascal; Steimle, Frank; Gröger, Christoph; Mitschang, Bernhard; Rehder, Eike; Lucke, Dominik;Abdul Rahman, Omar; Bauernhansl, Thomas: Towards a Rule-Based Manufacturing Integration Assistant. In: Proceedings of the 49th CIRP Conference on Manufacturing Systems (CIRP-CMS 2016), Stuttgart, Germany, 2016

2017

- Kassner, Laura; Hirmer, Pascal; Wieland, Matthias; Steimle, Frank; Königsberger, Jan: Connecting People, Machines & Data in Manufacturing for Context-Aware Exception Escalation. In: Proceedings of the Hawaii International Conference on System Sciences 50 (HICSS), Hawaii, 2017
- Behringer, Michael; Hirmer, Pascal: Towards Interactive Data Processing and Analytics - Involving the Human in the Loop, In Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS), 2017
- Franco da Silva, Ana Cristina; Hirmer, Pascal; Képes, Kálmán; Breitenbücher, Uwe; Kopp, Oliver; Leymann, Frank; Mitschang, Bernhard:

Internet of Things Out of the Box: Using TOSCA for Automating the Deployment of IoT Environments. In Proceedings of the 7th International Conference on Cloud Computing and Services Science (CLOSER), 2017

- Hoos, Eva; Hirmer, Pascal; Mitschang, Bernhard: Improving Problem Resolving on the Shop Floor by Context-Aware Decision Information Packages. In Proceedings of the CAiSE 2017 Forum at the 29th International Conference on Advanced Information Systems Engineering (CAiSE), 2017
- Franco da Silva, Ana; Hirmer, Pascal; Wieland, Matthias; Mitschang, Bernhard: SitRS XT – Towards Near Real Time Situation Recognition. In: Journal of Information and Data Management, 2016
- Hoos, Eva; Hirmer, Pascal; Mitschang, Bernhard: Context-Aware Decision Information Packages: An Approach to Human-Centric Smart Factories. In Proceedings of the European Conference on Advances in Databases and Information Systems (ADBIS), 2017
- Franco da Silva, Ana Cristina; Hirmer, Pascal; Breitenbücher, Uwe; Mitschang, Bernhard: Customization and provisioning of complex event processing using TOSCA. In Proceedings of the 11th Symposium and Summer School On Service-Oriented Computing (SummerSOC), 2017
- Hüffmeyer, Marc; Hirmer, Pascal; Wieland, Matthias; Schreyer, Ulf; Mitschang, Bernhard: SitAC – A System for Situation-aware Access Control - Controlling Access to Sensor Data. In: Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP), 2017

2018

- Franco da Silva, Ana Cristina; Hirmer, Pascal; Mitschang, Bernhard: An Approach for CEP Query Shipping to Support Distributed IoT Environments. In: Proceedings of the 14th Workshop on Context and Activity Modeling and Recognition (COMOREA) at IEEE Percom, 2018

- Ghabri, Rachaa; Hirmer, Pascal; Mitschang, Bernhard: A hybrid approach to implement data driven optimization into production environments. In: Proceedings of the 21st International Conference on Business Information Systems (BIS), 2018
- Hoos, Eva; Hirmer, Pascal; Mitschang, Bernhard: Automated Creation and Provisioning of Decision Information Packages for the Smart Factory, In: Complex Systems Informatics and Modeling Quarterly, 2018
- Franco da Silva, Ana Cristina; Hirmer, Pascal; Breitenbücher, Uwe; Kopp, Oliver; Mitschang, Bernhard: TDLIoT: A Topic Description Language for the Internet of Things, In: Proceedings of the 18th International Conference on Web Engineering (ICWE), 2018
- Hüffmeyer, Marc; Hirmer, Pascal; Wieland, Matthias; Schreyer, Ulf; Mitschang, Bernhard: Situation-aware Access Control for Industrie 4.0. In: Revised and Selected Papers from ICISSP17, Communications in Computer and Information Science, Springer, 2018

Des Weiteren sind folgende Softwarepublikationen entstanden:

- FlexMash Data Mashup Tool
  Github: `https://github.com/hirmerpl/FlexMash`
- SitOPT-Implementierungen zur Situationserkennung
  Github: `https://github.com/SitOPT`
- Erweiterung der RMP (mittlerweile MBP – Multi-purpose Binding and Provisioning Platform)
  Github: `https://github.com/ana-silva/MBP`

Betreute studentische Arbeiten

Im Folgenden werden studentische Arbeiten aufgelistet, die an der Implementierung der Konzepte dieser Dissertation mitgewirkt haben und von mir betreut werden.

- Del Gaudio, Daniel; Bohn, Johannes; Paparoditis, Nikolas: Entwurf und Prototypische Implementierung einer Data Mashup-Plattform, Projekt-INF, 2015
- Josip, Ledic; Schöck, Janis: Vergleich von Event-Processing- und Streaming-Engines, Fachstudie, 2015
- Jansa, Paul Eine OSLC-Plattform zur Unterstützung der Situationserkennung in Workflows, Diplomarbeit, 2015
- Schöck Janis: Testwerkzeug zur Simulation dynamischer Datenströme, Bachelorarbeit, 2016
- Kalyoncu, Baris: Codefragment-Repositorium für Data Mashup Transformationsmuster, Diplomarbeit, 2016
- Grumbein, Sven: Automatisierte, Ontologie-basierte Sensorintegration für das Internet der Dinge, Diplomarbeit, 2016

- Del Gaudio, Daniel: TOSCA4Mashups – Provisionierung und Ausführung von Data Mashups in der Cloud, Bachelorarbeit, 2016
- Franco da Silva, Ana Cristina: Situationserkennung basierend auf Complex Event Processing, Masterarbeit, 2016
- Beisel, Martin; Rakel, Simon; Rodi, Michael; Voigt, Marc: TodoMVC für Cloud-Deployment und -Management, Fachstudie, 2016
- Blehm, Alexander; Kabierschke, Oliver; Lehmann, Simon: Analyse und Vergleich von IoT-Plattformen, Prozessanalyse, 2016
- Hüneburg, Armin: Automatische, TOSCA-basierte Provisionierung des Situationserkennungssystems SitOPT, Bachelorarbeit, 2016
- Ivanova, Victoria: Eine Schnittstelle zur einfachen Erweiterbarkeit von Datenverarbeitungs-Diensten, Bachelorarbeit, 2017
- Neugebauer, Nikolai: Dynamische Teilausführung von Workflows zur Modellierungszeit, Masterarbeit, 2017
- Sarangi, Sunayana: Optimizing the efficiency of data-intensive data mashups using Map-Reduce, Masterarbeit, 2017
- Mahrous, Khaled: A framework for service based data processing, Masterarbeit, 2017
- Ziegler, Julian: Ein Sicherheitskonzept für verteilte Datenverarbeitungssysteme, Masterarbeit, 2017
- Fouskas, Alexandros: Automatisches Auffinden und Anbinden von IoT-Geräten, Bachelorarbeit, 2017

Zusätzlich zu den aufgeführten Arbeiten wurden diverse Fachpraktika im Themenbereich dieser Dissertation durchgeführt. Die Ergebnisse fließen ebenfalls in die Implementierungen im Zusammenhang mit dieser Dissertation ein.

Literaturverzeichnis

[ABJ+04] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, S. Mock. „Kepler: an extensible system for design and execution of scientific workflows“. In: *Proceedings of the 16th International Conference on Scientific and Statistical Database Management, 2004.* Juni 2004, S. 423–424 (Zitiert auf S. 178, 179).

[ABJ06] I. Altintas, O. Barney, E. Jaeger-Frank. „Provenance Collection Support in the Kepler Scientific Workflow System“. In: *Proceedings of the 2006 International Conference on Provenance and Annotation of Data*. IPAW'06. Chicago, IL: Springer-Verlag, 2006, S. 118–132 (Zitiert auf S. 178, 179).

[ACD+03] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte et al. *Business process execution language for web services*. online. 2003 (Zitiert auf S. 47, 107).

[AIS+77] C. Alexander, S. Ishikawa, M. Silverstein, J. R. i Ramió, M. Jacobson, I. Fiksdahl-King. *A pattern language*. Gustavo Gili, 1977 (Zitiert auf S. 76, 84).

[AIS93a] R. Agrawal, T. Imieliński, A. Swami. „Mining Association Rules Between Sets of Items in Large Databases“. In: *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*. SIGMOD '93. Washington, D.C., USA: ACM, 1993, S. 207–216. URL: http://doi.acm.org/10.1145/170035.170072 (Zitiert auf S. 45, 80).

[AIS93b] R. Agrawal, T. Imieliński, A. Swami. „Mining Association Rules Between Sets of Items in Large Databases“. In: *SIGMOD Rec.* 22.2 (Juni 1993), S. 207–216. URL: http://doi.acm.org/10.1145/170036.170072 (Zitiert auf S. 45, 80).

[AP13] S. Aghaee, C. Pautasso. „Live mashup tools: challenges and opportunities“. In: *Proceedings of the 1st International Workshop on Live Programming, LIVE 2013, San Francisco, California, USA, May 19, 2013*. 2013, S. 1–4. URL: http://dx.doi.org/10.1109/LIVE.2013.6617338 (Zitiert auf S. 181).

[Arn+07] Arnold et al. „Pattern Based SOA Deployment“. In: *ICSOC*. Springer, 2007, S. 1–12 (Zitiert auf S. 138).

[ASRH13] J. Attard, S. Scerri, I. Rivera, S. Handschuh. „Ontology-based Situation Recognition for Context-aware Systems“. In: *Proceedings of the 9th International Conference on Semantic Systems*. Graz, Austria, 2013 (Zitiert auf S. 213).

[BBH+13] T. Binz, U. Breitenbücher, F. Haupt, O. Kopp, F. Leymann, A. Nowak, S. Wagner. „OpenTOSCA – A Runtime for TOSCA-based Cloud Applications“. Englisch. In: *Proceedings of 11th International Conference on Service-Oriented Computing (ICSOC 13)*. Bd. 8274. LNCS. Springer Berlin Heidelberg, Dez. 2013, S. 692–695 (Zitiert auf S. 58, 134).

[BBK+12] U. Breitenbücher, T. Binz, O. Kopp, F. Leymann, D. Schumm. „Vino4TOSCA: A Visual Notation for Application Topologies based on TOSCA“. Englisch. In: *Proceedings of the 20th International Conference on Cooperative Information Systems (CoopIS 2012)*. Lecture Notes in Computer Science. Springer-Verlag, Sep. 2012 (Zitiert auf S. 30, 58, 138, 139).

[BBK+14a] U. Breitenbücher, T. Binz, K. Képes, O. Kopp, F. Leymann, J. Wettinger. „Combining Declarative and Imperative Cloud Application Provisioning based on TOSCA“. Englisch. In: *Proceedings of the IEEE International Conference on Cloud Engineering (IC2E)*. IEEE Computer Society, März 2014, S. 87–96 (Zitiert auf S. 56, 116, 134).

[BBK+14b] U. Breitenbücher, T. Binz, O. Kopp, F. Leymann, M. Wieland. „Context-aware Cloud Application Management“. Englisch. In: *Proceedings of the 4th International Conference on Cloud Computing and Services Science (CLOSER 2014)*. SciTePress, Apr. 2014, S. 499–509 (Zitiert auf S. 57).

[BBKL14a] U. Breitenbücher, T. Binz, O. Kopp, F. Leymann. „Automating Cloud Application Management Using Management Idioms“. Englisch. In: *Proceedings of the Sixth International Conferences on Pervasive Patterns and Applications (PATTERNS 2014)*. Xpert Publishing Services, Mai 2014, S. 60–69 (Zitiert auf S. 57).

[BBKL14b] U. Breitenbücher, T. Binz, O. Kopp, F. Leymann. „Vinothek - A Self-Service Portal for TOSCA“. Englisch. In: *Proceedings of the 6th Central-European Workshop on Services and their Composition (ZEUS 2014)*. Hrsg. von N. Herzberg, M. Kunze. Bd. 1140. CEUR Workshop Proceedings. CEUR-WS.org, März 2014, S. 69–72 (Zitiert auf S. 58).

[BBLS12] T. Binz, G. Breiter, F. Leyman, T. Spatzier. „Portable Cloud Services Using TOSCA“. In: *IEEE Internet Computing* 16.3 (Mai 2012), S. 80–85 (Zitiert auf S. 154).

[Beh16] M. Behringer. „Visual Analytics im Kontext der Daten- und Analysequalität am Beispiel von Data-Mashups“. Diplomarbeit. Magisterarb. Universität Stuttgart, 2016 (Zitiert auf S. 200).

[BHM17] M. Behringer, P. Hirmer, B. Mitschang. „Towards Interactive Data Processing and Analytics - Putting the Human in the Center of the Loop“. In: *Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS)*. 2017 (Zitiert auf S. 194, 200, 227).

[Bin+12] T. Binz et al. „Formalizing the Cloud through Enterprise Topology Graphs“. Englisch. In: *CLOUD*. IEEE, Juni 2012, S. 742–749 (Zitiert auf S. 138).

[BK09] A. Buchmann, B. Koldehofe. „Complex event processing“. In: *it-Information Technology Methoden und innovative Anwendungen der Informatik und Informationstechnik* (2009) (Zitiert auf S. 214).

[BKL16] A. Blehm, O. Kabierschke, S. Lehmann. *Analyse und Vergleich von IoT-Plattformen*. Prozessanalyse. 2016 (Zitiert auf S. 161).

[BMK+00] B. Brumitt, B. Meyers, J. Krumm, A. Kern, S. Shafer. „EasyLiving: Technologies for Intelligent Environments“. English. In: *Handheld and Ubiquitous Computing*. Springer Berlin Heidelberg, 2000 (Zitiert auf S. 213).

[Bor04] C. Borgelt. „Recursion Pruning for the Apriori Algorithm.“ In: *FIMI*. Bd. 126. 2004 (Zitiert auf S. 80).

[BPM02] A. C. P. Barbosa, F. A. Porto, R. N. Melo. „Configurable data integration middleware system“. en. In: *Journal of the Brazilian Computer Society* (2002) (Zitiert auf S. 160).

[Bre16] U. Breitenbücher. „Eine musterbasierte Methode zur Automatisierung des Anwendungsmanagements“. Diss. Universität Stuttgart, 2016 (Zitiert auf S. 57).

[BRRV16] M. Beisel, S. Rakel, M. Rodi, M. Voigt. *TodoMVC für Cloud-Deployment und -Management*. Fachstudie. 2016 (Zitiert auf S. 57).

[Cha04] D. Chappell. *Enterprise service bus*. O'Reilly Media, Inc., 2004 (Zitiert auf S. 47).

[Che76] P. P.-S. Chen. „The Entity-Relationship Model – Toward a Unified View of Data“. In: *ACM Trans. Database Syst.* 1.1 (März 1976), S. 9–36. URL: http://doi.acm.org/10.1145/320434.320440 (Zitiert auf S. 81).

[CKE+15] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, K. Tzoumas. „Apache Flink™: Stream and Batch Processing in a Single Engine“. In: *IEEE Data Eng. Bull.* 38.4 (2015), S. 28–38. URL: http://sites.computer.org/debull/A15dec/p28.pdf (Zitiert auf S. 42, 168).

[Coh+09] D. Cohn et al. „Business artifacts: A Data-centric Approach to Modeling Business Operations and Processes“. In: *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* (2009) (Zitiert auf S. 35, 76, 88).

[Das17] S. Das. *Modeling Recommendation for Data Flow Models*. Masterarbeit, noch nicht veröffentlicht. 2017 (Zitiert auf S. 83, 91).

[Del16] D. Del Gaudio. *TOSCA4Mashups – Provisionierung und Ausführung von Data Mashups in der Cloud*. Bachelorarbeit. 2016 (Zitiert auf S. 140).

[DEM+13] W. Dargie, Eldora, J. Mendez, C. Mobius, K. Rybina, V. Thost, A.-Y. Turhan. „Situation Recognition for Service Management Systems Using OWL 2 Reasoners“. In: *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*. 2013 (Zitiert auf S. 211, 213).

[DeV16] A. DeVan. *The 7 V's of Big Data*. online. Apr. 2016. URL: https://impact.com/marketing-intelligence/7-vs-big-data/ (Zitiert auf S. 39, 43).

[Dev96] B. Devlin. *Data Warehouse: From Architecture to Implementation*. Hrsg. von L. D. Cote. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1996 (Zitiert auf S. 30).

[Dey01] A. K. Dey. „Understanding and Using Context“. In: *Personal Ubiquitous Comput.* 5.1 (Jan. 2001), S. 4–7. URL: http://dx.doi.org/10.1007/s007790170019 (Zitiert auf S. 204).

[DM14] F. Daniel, M. Matera. *Mashups - Concepts, Models and Architectures*. Data-Centric Systems and Applications. Springer, 2014. URL: http://dx.doi.org/10.1007/978-3-642-55049-2 (Zitiert auf S. 21, 85).

[DN09] X. L. Dong, F. Naumann. „Data Fusion: Resolving Data Conflicts for Integration“. In: *Proc. VLDB Endow.* 2.2 (Aug. 2009), S. 1654–1655. URL: https://doi.org/10.14778/1687553.1687620 (Zitiert auf S. 32).

[Doc] Docker. *Docker Software Container Platform*. online. https://www.docker.com/ (Zitiert auf S. 30, 57, 139).

[EFGK03] P. T. Eugster, P. A. Felber, R. Guerraoui, A.-M. Kermarrec. „The Many Faces of Publish/Subscribe“. In: *ACM Comput. Surv.* 35.2 (Juni 2003), S. 114–131. URL: http://doi.acm.org/10.1145/857076.857078 (Zitiert auf S. 147, 153).

[Eil+06] T. Eilam et al. „Managing the configuration complexity of distributed applications in Internet data centers“. In: *Communications Magazine, IEEE* 44.3 (März 2006), S. 166–177 (Zitiert auf S. 137).

[Eil+11] T. Eilam et al. „Pattern-based Composite Application Deployment“. In: *IM*. IEEE, Mai 2011, S. 217–224 (Zitiert auf S. 138).

[Esp] EsperTech. *Esper Complex Event Processing Engine*. online. URL: `http://www.espertech.com/` (Zitiert auf S. 212).

[Fan15] H. Fang. „Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem". In: *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*. Juni 2015, S. 820–824 (Zitiert auf S. 59).

[FBFL15] C. Fehling, J. Barzen, M. Falkenthal, F. Leymann. „PatternPedia - Collaborative Pattern Identification and Authoring". Deutsch. In: *Proceedings of PURPLSOC (Pursuit of Pattern Languages for Societal Change). The Workshop 2014*. n.n., Juli 2015, S. 252–284 (Zitiert auf S. 78, 84).

[Feh15] C. Fehling. „Cloud computing patterns : identification, design, and application". Diss. Universität Stuttgart, 2015 (Zitiert auf S. 78, 84).

[FHWM16] A. C. Franco da Silva, P. Hirmer, M. Wieland, B. Mitschang. „SitRS XT – Towards Near Real Time Situation Recognition". Englisch. In: *Journal of Information and Data Management* 7.1 (Apr. 2016), S. 4–17 (Zitiert auf S. 212).

[FLR+14] C. Fehling, F. Leymann, R. Retter, W. Schupeck, P. Arbitter. *Cloud Computing Patterns*. Deutsch. Springer Wien, Jan. 2014, S. 394 (Zitiert auf S. 40).

[FPS96] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth. „The KDD Process for Extracting Useful Knowledge from Volumes of Data". In: *Communications of the ACM* 39.11 (Nov. 1996), S. 27–34 (Zitiert auf S. 20, 45).

[Fra15] A. C. Franco da Silva. *Situation Recognition based on Complex Event Processing*. Masterarbeit. 2015 (Zitiert auf S. 212).

[FS06] R. Feldman, J. Sanger. *Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. New York, NY, USA: Cambridge University Press, 2006 (Zitiert auf S. 41, 156, 216).

[GA15] S. Gopalani, R. Arora. „Comparing Apache Spark and Map Reduce with Performance Analysis using K-Means“. In: *International Journal of Computer Applications* 113.1 (März 2015), S. 8–11 (Zitiert auf S. 168).

[GAH+08] A. Grant, M. Antonioletti, A. C. Hume et al. „OGSA-DAI: Middleware for Data Integration: Selected Applications“. In: *eScience, 2008. eScience '08. IEEE Fourth International Conference on*. 2008 (Zitiert auf S. 160).

[GBH+05] M. Großmann, M. Bauer, N. Hönle, U.-P. Käppeler, D. Nicklas, T. Schwarz. „Efficiently Managing Context Information for Large-Scale Scenarios“. In: *Proc. of the Third IEEE Intl. Conf. on Pervasive Computing and Communications*. 2005 (Zitiert auf S. 213).

[GBMP13] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami. „Internet of Things (IoT): A vision, architectural elements, and future directions“. In: *Future Generation Computer Systems* 29.7 (2013). Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services & Cloud Computing and Scientific Applications - Big Data, Scalable Analytics, and Beyond, S. 1645–1660. URL: `http://www.sciencedirect.com/science/article/pii/S0167739X13000241` (Zitiert auf S. 19).

[GD17] M. Gaedke, F. Daniel. „Challenge Outcome and Conclusion“. In: *Rapid Mashup Development Tools: Second International Rapid Mashup Challenge, RMC 2016, Lugano, Switzerland, June 6, 2016, Revised Selected Papers*. Hrsg. von F. Daniel, M. Gaedke. Cham: Springer International Publishing, 2017, S. 129–134 (Zitiert auf S. 200, 201).

[GR07] J. Gama, P. P. Rodrigues. „Data stream processing“. In: *Learning from Data Streams-Processing Techniques in Sensor Networks* (2007), S. 25–39 (Zitiert auf S. 20).

[Gro06] O. M. Group. „Business Process Modeling Notation“. In: *Needham, MA, USA* 2.2 (2006) (Zitiert auf S. 47, 107).

[Gru16] S. Grumbein. *Automatisierte, Ontologie-basierte Sensorintegration für das Internet der Dinge*. Diplomarbeit. 2016 (Zitiert auf S. 157).

[GZK10] M. M. Gaber, A. Zaslavsky, S. Krishnaswamy. „Data Stream Mining“. In: *Data Mining and Knowledge Discovery Handbook*. Hrsg. von O. Maimon, L. Rokach. Boston, MA: Springer US, 2010, S. 759–787 (Zitiert auf S. 20, 31, 42).

[Has] HashiCorp. *Vagrant – Development Environments Made Easy*. online. `https://www.vagrantup.com/` (Zitiert auf S. 30, 57, 139).

[Hat] R. Hat. *Ansible*. online. URL: `http://www.ansible.com` (Zitiert auf S. 139).

[HB17] P. Hirmer, M. Behringer. „FlexMash 2.0 – Flexible Modeling and Execution of Data Mashups“. Englisch. In: Bd. 696. Rapid Mashup Development Tools - Second International Rapid Mashup Challenge, RMC 2016, Lugano, Switzerland, June 6, 2016, Revised Selected Papers. Springer International Publishing, Jan. 2017, S. 10–29 (Zitiert auf S. 65, 91, 93, 116, 198, 201).

[HBBL14] P. Hirmer, U. Breitenbücher, T. Binz, F. Leymann. „Automatic Topology Completion of TOSCA-based Cloud Applications“. Englisch. In: *Proceedings des CloudCycle14 Workshops auf der 44. Jahrestagung der Gesellschaft für Informatik e.V. (GI)*. Bd. 232. LNI. Bonn: Gesellschaft für Informatik e.V. (GI), Sep. 2014, S. 247–258 (Zitiert auf S. 38, 82, 125, 129–131, 133, 139, 143).

[HBM17] P. Hirmer, M. Behringer, B. Mitschang. „Partial Execution of Mashup Plans During Modeling Time“. In: *Research and Development* (2017) (Zitiert auf S. 163, 177, 178, 184, 186, 188).

[HBS+16] P. Hirmer, U. Breitenbücher, A. C. F. da Silva, K. Képes, B. Mitschang, M. Wieland. „Automating the Provisioning and Configuration of Devices in the Internet of Things“. Englisch. In: *Complex Systems Informatics and Modeling Quarterly* 9 (Dez. 2016), S. 28–43 (Zitiert auf S. 37, 148, 158, 209).

[Her12] M. Herschel. „Datenintegration“. In: *it – Information Technology* 54.3 (2012), S. 103–104. URL: `http://dx.doi.org/10.1524/itit.2012.9075` (Zitiert auf S. 41).

[HHL+10] K. Häussermann, C. Hubig, P. Levi, F. Leymann, O. Siemoneit, M. Wieland, O. Zweigle. „Understanding and Designing Situation-Aware Mobile and Ubiquitous Computing Systems“. In: *Proceedings of the International Conference on Computer, Electrical, and Systems Science, and Engineering 2010 (ICCESSE 2010)*. 2010 (Zitiert auf S. 206, 213).

[HHM17a] E. Hoos, P. Hirmer, B. Mitschang. „Context-Aware Decision Information Packages: An Approach to Human-Centric Smart Factories“. In: *Proceedings of the European Conference on Advances in Databases and Information Systems (ADBIS)*. 2017 (Zitiert auf S. 148, 214, 215, 217).

[HHM17b] E. Hoos, P. Hirmer, B. Mitschang. „Improving Problem Resolving on the Shop Floor by Context-Aware Decision Information Packages“. In: *Proceedings of the CAiSE 2017 Forum at the 29th International Conference on Advanced Information Systems Engineering (CAiSE)*. 2017 (Zitiert auf S. 214, 217).

[Hir15] P. Hirmer. „Flexible Modeling and Execution of Data Integration Flows“. Englisch. In: *Proceedings of the 9th Advanced Summer School on Service Oriented Computing*. Hrsg. von J. Barzen, R. Khalaf, F. Leymann, B. Mitschang. IBM Research Report, Dez. 2015, S. 153–154 (Zitiert auf S. 35, 93).

[Hir17] P. Hirmer. „Effizienz-Optimierung daten-intensiver Data Mashups am Beispiel von Map-Reduce“. Deutsch. In: *Proceedings der Datenbanksysteme für Business, Technologie und Web (BTW), 17. Fachtagung des GI-Fachbereichs, Workshopband*. Hrsg. von B. Mitschang, N. Ritter, H. Schwarz, M. Klettke, A. Thor, O. Kopp, M. Wieland. Bd. P-266. LNI. Stuttgart: Gesellschaft für Informatik (GI), März 2017, S. 111–116 (Zitiert auf S. 93, 100, 163, 166–168).

[HM16a] P. Hirmer, B. Mitschang. „FlexMash - Flexible Data Mashups Based on Pattern-Based Model Transformation“. Englisch. In: Bd. 591. Rapid Mashup Development Tools. Springer International Publishing, Feb. 2016, S. 12–30 (Zitiert auf S. 36, 70, 91, 93, 95, 116, 198).

[HM16b] P. Hirmer, B. Mitschang. „TOSCA4Mashups: enhanced method for on-demand data mashup provisioning“. Englisch. In: *Computer Science -*

Research and Development (Okt. 2016), S. 1–10 (Zitiert auf S. 125, 127, 129, 131).

[HPY00] J. Han, J. Pei, Y. Yin. „Mining Frequent Patterns Without Candidate Generation“. In: *SIGMOD Rec.* 29.2 (Mai 2000), S. 1–12. URL: http://doi.acm.org/10.1145/335191.335372 (Zitiert auf S. 75, 81).

[HRWM15] P. Hirmer, P. Reimann, M. Wieland, B. Mitschang. „Extended Techniques for Flexible Modeling and Execution of Data Mashups“. Englisch. In: *Proceedings of the 4th International Conference on Data Management Technologies and Applications (DATA)*. Hrsg. von M. Helfert, A. Holzinger, O. Belo, C. Francalanci. Colmar: SciTePress, Juli 2015, S. 111–122 (Zitiert auf S. 35, 70, 76, 107).

[HW12] E. R. Hancock, R. C. Wilson. „Pattern analysis with graphs: Parallel work at Bern and York“. In: *Pattern Recognition Letters* 33.7 (2012). Special Issue on Awards from {ICPR} 2010, S. 833–841. URL: http://www.sciencedirect.com/science/article/pii/S0167865511002637 (Zitiert auf S. 75).

[HW79] J. A. Hartigan, M. A. Wong. „Algorithm AS 136: A K-Means Clustering Algorithm“. In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28.1 (1979), S. 100–108. URL: http://www.jstor.org/stable/2346830 (Zitiert auf S. 45).

[HWBM16a] P. Hirmer, M. Wieland, U. Breitenbücher, B. Mitschang. „Automated Sensor Registration, Binding and Sensor Data Provisioning“. Englisch. In: *Proceedings of the CAiSE'16 Forum, at the 28th International Conference on Advanced Information Systems Engineering (CAiSE 2016)*. Bd. 1612. CEUR Workshop Proceedings. Ljubljana, Slovenia: CEUR-WS.org, Juni 2016, S. 81–88 (Zitiert auf S. 37, 148).

[HWBM16b] P. Hirmer, M. Wieland, U. Breitenbücher, B. Mitschang. „Dynamic Ontology-based Sensor Binding“. Englisch. In: *Advances in Databases and Information Systems. 20th East European Conference, ADBIS 2016, Prague, Czech Republic, August 28-31, 2016, Proceedings*. Bd. 9809. Information Systems and Applications, incl. Internet/Web, and HCI. Prague, Czech Republic: Springer International Publishing, Aug. 2016, S. 323–337 (Zitiert auf S. 37, 146, 148, 149, 151).

[HWF+17] P. Hirmer, T. Waizenegger, G. Falazi, M. Abdo, Y. Volga, A. Askinadze, M. Liebeck, S. Conrad, T. Hildebrandt, C. Indiono, S. Rinderle-Ma, M. Grimmer, M. Kricke, E. Peukert. „The First Data Science Challenge at BTW 2017“. In: *Datenbank-Spektrum* (Juli 2017). URL: https://doi.org/10.1007/s13222-017-0263-8 (Zitiert auf S. 31).

[HWS+15] P. Hirmer, M. Wieland, H. Schwarz, B. Mitschang, U. Breitenbücher, F. Leymann. „SitRS - A Situation Recognition Service based on Modeling and Executing Situation Templates“. Englisch. In: *Proceedings of the 9th Symposium and Summer School On Service-Oriented Computing*. Hrsg. von J. Barzen, R. Khalaf, F. Leymann, B. Mitschang. Bd. RC25564. Technical Paper. IBM Research Report, Dez. 2015, S. 113–127 (Zitiert auf S. 204, 211).

[HWS+16] P. Hirmer, M. Wieland, H. Schwarz, B. Mitschang, U. Breitenbücher, S. G. Sáez, F. Leymann. „Situation recognition and handling based on executing situation templates and situation-aware workflows“. Englisch. In: *Computing* (Okt. 2016), S. 1–19 (Zitiert auf S. 204, 208, 209).

[IBMa] IBM. *IBM SPSS*. online. https://www.ibm.com/analytics/de/de/technology/spss/ (Zitiert auf S. 27).

[IBMb] IBM. *Node-RED – Flow-based programming for the Internet of Things*. online. URL: https://nodered.org/ (Zitiert auf S. 107).

[IHR+12] I. Ishaq, J. Hoebeke, J. Rossey, E. De Poorter, I. Moerman, P. Demeester. „Facilitating Sensor Deployment, Discovery and Resource Access Using Embedded Web Services“. In: *Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*. Juli 2012, S. 717–724 (Zitiert auf S. 161).

[Iva17] V. Ivanova. *Eine Schnittstelle zur einfachen Erweiterbarkeit von Datenverarbeitungsdiensten*. Bachelorarbeit. 2017 (Zitiert auf S. 117, 119, 120).

[IWM03] A. Inokuchi, T. Washio, H. Motoda. „Complete Mining of Frequent Patterns from Graphs: Mining Graph Data“. In: *Machine Learning* 50.3 (2003), S. 321–354. URL: http://dx.doi.org/10.1023/A:1021726221443 (Zitiert auf S. 75).

[Jai16] A. Jain. *The 5 Vs of Big Data*. online. Sep. 2016. URL: https://www.ibm.com/blogs/watson-health/the-5-vs-of-big-data/ (Zitiert auf S. 39, 42).

[KAF+08] D. A. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, G. Melançon. „Visual Analytics: Definition, Process, and Challenges“. In: *Information Visualization*. Hrsg. von A. Kerren, J. T. Stasko, J.-D. Fekete, C. North. Berlin, Heidelberg: Springer, 2008, S. 154–175 (Zitiert auf S. 43).

[Kal16] B. Kalyoncu. *Codefragment-Repositorium für Data Mashup Transformationsmuster*. Diplomarbeit. 2016 (Zitiert auf S. 115, 117, 118).

[Kas17] L. B. Kassner. *Product Life Cycle Analytics - Analytics auf unstrukturierten Daten für eine intelligentere Fertigung*. Brandsberg 6, 53797 Lohmar: JOSEF EUL VERLAG GmbH, 2017 (Zitiert auf S. 156).

[KBBL13] O. Kopp, T. Binz, U. Breitenbücher, F. Leymann. „Winery - A Modeling Tool for TOSCA-based Cloud Applications“. Englisch. In: *Proceedings of 11th International Conference on Service-Oriented Computing (ICSOC'13)*. Bd. 8274. LNCS. Springer Berlin Heidelberg, Dez. 2013, S. 700–704 (Zitiert auf S. 58, 128, 139, 140).

[Kep13] K. Kepes. *Konzept und Implementierung eine Java-Komponente zur Generierung von WS-BPEL 2.0 BuildPlans für OpenTOSCA*. Deutsch. Bachelor Thesis: University of Stuttgart, Institute of Architecture of Application Systems. Juli 2013 (Zitiert auf S. 116, 142).

[KHW+17] L. Kassner, P. Hirmer, M. Wieland, F. Steimle, J. Königsberger, B. Mitschang. „The Social Factory: Connecting People, Machines and Data in Manufacturing for Context-Aware Exception Escalation“. Englisch. In: *Proceedings of the 50th Hawaii International Conference on System Sciences*. Online, Jan. 2017, S. 1–10 (Zitiert auf S. 216, 217).

[Kie16] C. Kiefer. „Assessing the Quality of Unstructured Data: An Initial Overview“. In: *LWDA*. 2016 (Zitiert auf S. 156).

[KK05] M. Kuramochi, G. Karypis. „Finding Frequent Patterns in a Large Sparse Graph*“. In: *Data Min. Knowl. Discov.* 11.3 (Nov. 2005), S. 243–271. URL: http://dx.doi.org/10.1007/s10618-005-0003-9 (Zitiert auf S. 76).

[KM17] J. Königsberger, B. Mitschang. „Business Objects plus (BO+): An Approach to Enhance Service Reuse and Integration in Cross-Domain SOA Compounds“. In: *2017 IEEE International Conference on Information Reuse and Integration (IRI)*. Aug. 2017, S. 49–58 (Zitiert auf S. 76).

[Kop16] O. Kopp. „Partnerübergreifende Geschäftsprozesse und ihre Realisierung in BPEL“. Diss. Universität Stuttgart, 2016 (Zitiert auf S. 106, 116).

[KT11] K. T. Kearney, F. Torelli. „The SLA Model“. In: *Service Level Agreements for Cloud Computing*. Hrsg. von P. Wieder, J. M. Butler, W. Theilmann, R. Yahyapour. New York, NY: Springer New York, 2011, S. 43–67 (Zitiert auf S. 100).

[Kün+11] V. Künzle et al. „PHILharmonicFlows: Towards a Framework for Object-aware Process Management“. In: *Journal of Software Maintenance and Evolution: Research and Practice* (2011) (Zitiert auf S. 35, 88).

[LAB+06] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, Y. Zhao. „Scientific Workflow Management and the Kepler System: Research Articles“. In: *Concurr. Comput. : Pract. Exper.* 18.10 (Aug. 2006), S. 1039–1065. URL: http://dx.doi.org/10.1002/cpe.v18:10 (Zitiert auf S. 179).

[Lev66] V. I. Levenshtein. „Binary codes capable of correcting deletions, insertions, and reversals“. In: *Soviet physics doklady*. Bd. 10. 8. 1966, S. 707–710 (Zitiert auf S. 31).

[Lew06] D. Lewis. „What is Web 2.0?“ In: *Crossroads* 13.1 (Sep. 2006), S. 3–3. URL: http://doi.acm.org/10.1145/1217666.1217669 (Zitiert auf S. 21).

[LN07] U. Leser, F. Naumann. *Informationsintegration - Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen*. dpunkt.verlag, 2007 (Zitiert auf S. 31, 41, 161).

[LS07] R. Lu, S. Sadiq. „A Survey of Comparative Business Process Modeling Approaches“. In: *Business Information Systems: 10th International Conference, BIS 2007, Poznan, Poland, April 25-27, 2007. Proceedings*. Hrsg. von W. Abramowicz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, S. 82–94 (Zitiert auf S. 107).

[Luc01] D. C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001 (Zitiert auf S. 20, 31).

[LVCD13] F. Li, M. Vögler, M. Claeßens, S. Dustdar. „Towards automated IoT application deployment by a cloud-based approach“. In: *Proceedings of the IEEE 6th International Conference on Service-Oriented Computing and Applications (SOCA)*. IEEE. 2013, S. 61–68 (Zitiert auf S. 160).

[LWB08] A. Langegger, W. Wöß, M. Blöchl. „A Semantic Web Middleware for Virtual Data Integration on the Web“. In: *The Semantic Web: Research and Applications: 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008 Proceedings*. Springer Berlin Heidelberg, 2008, S. 493–507 (Zitiert auf S. 160).

[Mah17] K. Mahrous. *A framework for service-based data processing*. Masterarbeit. 2017 (Zitiert auf S. 107, 120).

[Man08] L. Mandel. *Describe REST Web services with WSDL 2.0 - A how-to guide*. online. Mai 2008. URL: `https://www.ibm.com/developerworks/library/ws-restwsdl/` (Zitiert auf S. 106, 120).

[MBD+12] A. McAfee, E. Brynjolfsson, T. H. Davenport, D. Patil, D. Barton. „Big data“. In: *The management revolution. Harvard Bus Rev* 90.10 (2012), S. 61–67 (Zitiert auf S. 19).

[MCAP16] O. C. Marcu, A. Costan, G. Antoniu, M. S. Pérez-Hernández. „Spark Versus Flink: Understanding Performance in Big Data Analytics Frameworks“. In: *2016 IEEE International Conference on Cluster Computing (CLUSTER)*. 2016, S. 433–442 (Zitiert auf S. 169).

[MCB+11] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, A. H. Byers. *Big data: The next frontier for innovation, competition, and productivity*. McKinsey Global Institute, Mai 2011 (Zitiert auf S. 19).

[Meu95] R. Meunier. „The pipes and filters architecture“. In: *Pattern languages of program design*. ACM Press/Addison-Wesley Publishing Co. 1995, S. 427–440 (Zitiert auf S. 72).

[MG+11] P. Mell, T. Grance et al. „The NIST definition of cloud computing“. In: (2011) (Zitiert auf S. 30, 37, 48, 50).

[MW15] N. Marz, J. Warren. *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications Co., 2015 (Zitiert auf S. 39).

[Neu17] N. Neugebauer. *Dynamische Teilausführung von Workflows zur Modellierungszeit*. Masterarbeit. 2017 (Zitiert auf S. 177, 192).

[Nor16] A. Nordrum. *Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated*. online. Aug. 2016. URL: http://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated (Zitiert auf S. 20, 59).

[OAS13a] OASIS. *Topology and Orchestration Specification for Cloud Applications*. Version 1.0. Advancing open standards for the information society, 2013. URL: http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.pdf (Zitiert auf S. 30, 36, 38, 53, 56, 64).

[OAS13b] OASIS. *TOSCA Primer*. Online. Version 1.0. Nov. 2013. URL: http://docs.oasis-open.org/tosca/tosca-primer/v1.0/cnd01/tosca-primer-v1.0-cnd01.pdf (Zitiert auf S. 36, 38).

[Old] U. Oldenburg. *Odysseus Complex Event System*. online. URL: http://odysseus.informatik.uni-oldenburg.de/ (Zitiert auf S. 212).

[OPP01] S. Orlando, P. Palmerini, R. Perego. „Enhancing the apriori algorithm for frequent set counting“. In: *DaWaK*. Bd. 1. Springer. 2001, S. 71–82 (Zitiert auf S. 80).

[ORe05] T. O'Reilly. „What Is Web 2.0? Design Patterns and Business Models for the Next Generation of Software.“ In: (2005). URL: http://oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html (Zitiert auf S. 21).

[PA03] C. Pautasso, G. Alonso. „Visual composition of web services“. In: *2003 IEEE Symposium on Human Centric Computing Languages and Environments (HCC 2003), 28-31 October 2003, Auckland, New Zealand*. 2003, S. 92–99. URL: http://dx.doi.org/10.1109/HCC.2003.1260208 (Zitiert auf S. 181).

[PA05] C. Pautasso, G. Alonso. „The JOpera visual composition language“. In: *J. Vis. Lang. Comput.* 16.1-2 (2005), S. 119–152. URL: http://dx.doi.org/10.1016/j.jvlc.2004.08.004 (Zitiert auf S. 87, 181).

[Pau08] C. Pautasso. „BPEL for REST“. In: *Business Process Management: 6th International Conference, BPM 2008, Milan, Italy, September 2-4, 2008. Proceedings*. Hrsg. von M. Dumas, M. Reichert, M.-C. Shan. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, S. 278–293 (Zitiert auf S. 114, 120).

[PD16] C. Pautasso, F. Daniel. „Challenge Outcome and Conclusion“. In: *Rapid Mashup Development Tools: First International Rapid Mashup Challenge, RMC 2015, Rotterdam, The Netherlands, June 23, 2015, Revised Selected Papers*. Hrsg. von F. Daniel, C. Pautasso. Cham: Springer International Publishing, 2016, S. 118–122 (Zitiert auf S. 198, 200).

[PJ03] S. Park, S. Jayaraman. „Enhancing the quality of life through wearable technology“. In: *IEEE Engineering in Medicine and Biology Magazine* 22.3 (Mai 2003), S. 41–48 (Zitiert auf S. 20).

[Pru07] M. Pruett. *Yahoo! Pipes*. First. O'Reilly, 2007 (Zitiert auf S. 22).

[RCD+14] C. Rodriguez, S. R. Chowdhury, F. Daniel, H. R. M. Nezhad, F. Casati. „Assisted Mashup Development: On the Discovery and Recommendation of Mashup Composition Knowledge“. In: *Web Services Foundations*. Hrsg. von A. Bouguettaya, Q. Z. Sheng, F. Daniel. New York, NY: Springer New York, 2014, S. 683–708 (Zitiert auf S. 85).

[RCN+13] S. Roy Chowdhury, O. Chudnovskyy, M. Niederhausen, S. Pietschmann, P. Sharples, F. Daniel, M. Gaedke. „Complementary Assistance Mechanisms for End User Mashup Composition“. In: *Proceedings of the 22Nd International Conference on World Wide Web*. WWW '13 Companion. Rio de Janeiro, Brazil: ACM, 2013, S. 269–272. URL:

http://doi.acm.org/10.1145/2487788.2487919 (Zitiert auf S. 83, 85).

[RDC11] S. Roy Chowdhury, F. Daniel, F. Casati. „Efficient, Interactive Recommendation of Mashup Composition Knowledge“. In: *Service-Oriented Computing: 9th International Conference, ICSOC 2011, Paphos, Cyprus, December 5-8, 2011 Proceedings*. Hrsg. von G. Kappel, Z. Maamar, H. R. Motahari-Nezhad. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, S. 374–388 (Zitiert auf S. 85).

[RDC16] C. Rodriguez, F. Daniel, F. Casati. „Mining and Quality Assessment of Mashup Model Patterns with the Crowd: A Feasibility Study“. In: *ACM Trans. Internet Technol.* 16.3 (Juni 2016), 17:1–17:27. URL: http://doi.acm.org/10.1145/2903138 (Zitiert auf S. 85).

[Rei17] P. Reimann. „Data provisioning in simulation workflows“. Diss. University of Stuttgart, Germany, 2017. URL: http://nbn-resolving.de/urn:nbn:de:bsz:93-opus-ds-90220 (Zitiert auf S. 74, 76, 84).

[RRDC12] S. Roy Chowdhury, C. Rodriguez, F. Daniel, F. Casati. „Baya: Assisted Mashup Development As a Service“. In: *Proceedings of the 21st International Conference on World Wide Web*. WWW '12 Companion. Lyon, France: ACM, 2012, S. 409–412. URL: http://doi.acm.org/10.1145/2187980.2188061 (Zitiert auf S. 83, 85).

[Sar17] S. Sarangi. *Optimizing the efficiency of data-intensive Data Mashups using Map-Reduce*. Masterarbeit. 2017 (Zitiert auf S. 169–176).

[Seb13] L. Sebastian-Coleman. „Chapter 4 - Data Quality and Measurement“. In: *Measuring Data Quality for Ongoing Improvement*. Hrsg. von L. Sebastian-Coleman. MK Series on Business Intelligence. Boston: Morgan Kaufmann, 2013, S. 39–53 (Zitiert auf S. 156).

[Sen13] U. Sendler. „Industrie 4.0– Beherrschung der industriellen Komplexität mit SysLM (Systems Lifecycle Management)“. In: *Industrie 4.0: Beherrschung der industriellen Komplexität mit SysLM*. Hrsg. von U. Sendler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, S. 1–19 (Zitiert auf S. 32).

[SHLP05] M. T. Schmidt, B. Hutchison, P. Lambros, R. Phippen. „The Enterprise Service Bus: Making service-oriented architecture real“. In: *IBM Systems Journal* 44.4 (2005), S. 781–797 (Zitiert auf S. 47).

[Shn96] B. Shneiderman. „The eyes have it: a task by data type taxonomy for information visualizations“. In: *Proceedings 1996 IEEE Symposium on Visual Languages*. Sep. 1996, S. 336–343 (Zitiert auf S. 73, 75).

[SK11] M. Sonntag, D. Karastoyanova. „Enforcing the Repeated Execution of Logic in Workflows“. In: *Proceedings of the 1st International Conference on Business Intelligence and Technology (BUSTECH 2011), Rome, Italy, 2011*. IARIA, Sep. 2011, S. 1–6 (Zitiert auf S. 178, 180).

[SK12] M. Sonntag, D. Karastoyanova. „Ad hoc Iteration and Re-execution of Activities in Workflows“. In: *International Journal On Advances in Software* 5.1 & 2 (Juli 2012), S. 91–109 (Zitiert auf S. 178, 180).

[SK13] M. Sonntag, D. Karastoyanova. „Model-as-you-go: An Approach for an Advanced Infrastructure for Scientific Workflows“. In: *Journal of Grid Computing* 11.3 (Sep. 2013), S. 553–583 (Zitiert auf S. 178, 180).

[SKLS11] D. Schumm, D. Karastoyanova, F. Leymann, S. Strauch. „Fragmento: Advanced Process Fragment Library“. In: *Information Systems Development: Business Systems and Services: Modeling and Development*. Hrsg. von J. Pokorny, V. Repa, K. Richta, W. Wojtkowski, H. Linger, C. Barry, M. Lang. New York, NY: Springer New York, 2011, S. 659–670 (Zitiert auf S. 108, 115).

[Spa15] A. Spark. *Apache Spark: Lightning-fast cluster computing*. online. 2015. URL: `http://spark.apache.org/` (Zitiert auf S. 30, 37, 40, 168).

[Sto14] A. Storm. „Storm, distributed and fault-tolerant realtime computation“. In: (2014). URL: `http://storm.apache.org/` (Zitiert auf S. 42).

[Sun+14] Y. Sun et al. „Modeling Data for Business Processes“. In: *Proceedings of the 30th IEEE International Conference on Data Engineering (ICDE), Chicago, USA*. 2014 (Zitiert auf S. 35, 88).

[TSJ+09] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, R. Murthy. „Hive: A Warehousing Solution over a Map-reduce Framework“. In: *Proc. VLDB Endow.* 2.2 (Aug. 2009), S. 1626–1629. URL: https://doi.org/10.14778/1687553.1687609 (Zitiert auf S. 37, 40).

[VB15] B. Varanasi, S. Belida. „HATEOAS“. In: *Spring REST*. Berkeley, CA: Apress, 2015, S. 165–174 (Zitiert auf S. 153).

[VF13] O. Vermesan, P. Friess. *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*. River Publishers, 2013 (Zitiert auf S. 59).

[VF14] O. Vermesan, P. Friess. *Internet of things-from research and innovation to market deployment*. River Publishers Aalborg, 2014 (Zitiert auf S. 19, 32).

[VSID16] M. Vögler, J. M. Schleicher, C. Inzinger, S. Dustdar. „A scalable framework for provisioning large-scale iot deployments“. In: *ACM Transactions on Internet Technology (TOIT)* 16.2 (2016), S. 11 (Zitiert auf S. 161).

[WBB+14] J. Wettinger, T. Binz, U. Breitenbücher, O. Kopp, F. Leymann, M. Zimmermann. „Unified Invocation of Scripts and Services for Provisioning, Deployment, and Management of Cloud Applications Based on TOSCA“. Englisch. In: *Proceedings of the 4th International Conference on Cloud Computing and Services Science (CLOSER 2014)*. SciTePress, Apr. 2014, S. 559–568 (Zitiert auf S. 110).

[WCL+05] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, D. F. Ferguson. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005 (Zitiert auf S. 46, 47, 55, 100, 102, 107).

[Wet17] J. Wettinger. „Gathering solutions and providing APIs for their orchestration to implement continuous software delivery“. Diss. Universität Stuttgart, 2017 (Zitiert auf S. 139).

[WFHP16] I. H. Witten, E. Frank, M. A. Hall, C. J. Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016 (Zitiert auf S. 80).

[WHW15] S. Woodman, H. Hiden, P. Watson. „Workflow Provenance: An Analysis of Long Term Storage Costs“. In: *Proceedings of the 10th Workshop on Workflows in Support of Large-Scale Science*. WORKS '15. Austin, Texas: ACM, 2015, 9:1–9:9. URL: http://doi.acm.org/10.1145/2822332.2822341 (Zitiert auf S. 47).

[WKQ+08] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. Mc Lachlan, A. Ng, B. Liu, S. Y. Philip et al. „Top 10 algorithms in data mining“. In: *Knowledge and information systems* 14.1 (2008), S. 1–37 (Zitiert auf S. 80).

[WSBL15] M. Wieland, H. Schwarz, U. Breitenbücher, F. Leymann. „Towards Situation-Aware Adaptive Workflows“. Englisch. In: *Proceedings of the 13th Annual IEEE Intl. Conference on Pervasive Computing and Communications Workshops: 11th Workshop on Context and Activity Modeling and Recognition*. St. Louis, Missouri, USA: IEEE, März 2015, S. 32–37 (Zitiert auf S. 203).

[WSO] WSO2. *Complex Event Processor*. online. URL: https://wso2.com/products/complex-event-processor/ (Zitiert auf S. 212).

[WWB+13] T. Waizenegger, M. Wieland, T. Binz, U. Breitenbücher, F. Haupt, O. Kopp, F. Leymann, B. Mitschang, A. Nowak, S. Wagner. „Policy4TOSCA: A Policy-Aware Cloud Service Provisioning Approach to Enable Secure Cloud Computing“. In: *On the Move to Meaningful Internet Systems: OTM 2013 Conferences: Confederated International Conferences: CoopIS, DOA-Trusted Cloud, and ODBASE 2013, Graz, Austria, September 9-13, 2013. Proceedings*. Hrsg. von R. Meersman, H. Panetto, T. Dillon, J. Eder, Z. Bellahsene, N. Ritter, P. De Leenheer, D. Dou. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, S. 360–376 (Zitiert auf S. 127).

[WZGP04] X. Wang, D. Q. Zhang, T. Gu, H. Pung. „Ontology based context modeling and reasoning using OWL“. In: *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*. 2004 (Zitiert auf S. 213).

[YC06] Y. Ye, C.-C. Chiang. „A Parallel Apriori Algorithm for Frequent Itemsets Mining“. In: *Fourth International Conference on Software Enginee-*

ring Research, Management and Applications (SERA'06). Aug. 2006, S. 87–94 (Zitiert auf S. 81).

[ZHKL09] O. Zweigle, K. Häussermann, U.-P. Käppeler, P. Levi. „Supervised Learning Algorithm for Automatic Adaption of Situation Templates Using Uncertain Data". In: *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*. Seoul, Korea, 2009 (Zitiert auf S. 206, 213).

[Zie17] J. Ziegler. *Ein Sicherheitskonzept für verteilte Datenverarbeitungssysteme*. Masterarbeit. 2017 (Zitiert auf S. 122).

[Zwe11] O. Zweigle. „Erweiterung kognitiver Fähigkeiten in Multiagentensystemen durch Kommunikation, Rollenverteilung und Situationsanalyse". Diss. University of Stuttgart, 2011 (Zitiert auf S. 206, 213).

Alle URLs wurden zuletzt am 18.06.2018 geprüft.

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Definitionsverzeichnis