Einheitliches Management serviceorientierter Systeme in einer Multi-Provider-Umgebung

Dissertation

zur Erlangung des Grades eines

D o k t o r s d e r N a t u r w i s s e n s c h a f t e n

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Christoph Fiehe

Dortmund

2019

Tag der mündlichen Prüfung:	03.09.2019
Dekan:	Prof. Dr.-Ing. Gernot A. Fink
Gutachter:	Prof. Dr. Heiko Krumm
	Prof. Dr. Kurt Geihs

Kurzfassung

Die zunehmende Digitalisierung der Geschäfts- und Alltagswelt stellt die heutige Unternehmens-IT vor immer größer werdende Herausforderungen. Die Unternehmen sind gezwungen, ihre Prozesse kontinuierlich zu optimieren und an veränderte Rahmen- und Marktbedingungen anzupassen. Die IT muss mit diesem Wandel Schritt halten. Als ein strategisches IT-Konzept bietet das Cloud-Computing die Möglichkeit, die IT-Landschaft bedarfsorientiert nach dem Baukastenprinzip zusammenzustellen. In den seltensten Fällen wird aber ein einzelner Anbieter über ein passendes Leistungsangebot verfügen, das sämtliche funktionalen und nichtfunktionalen Anforderungen abdeckt. Der Weg hin zu einer Multi-Provider-Umgebung ist somit vorgezeichnet und bereits durch Trends belegt. Allerdings stellt das einheitliche Management einer Multi-Provider-Umgebung, die neben cloudbasierten auch virtuelle und physikalische Umgebungen umfasst, eine Herausforderung dar. Die anforderungsgerechte Bereitstellung und der gütegesicherte Betrieb von Services erfordern den flexiblen Einsatz aller am Verbund beteiligten Ausführungsumgebungen. Im Rahmen dieser Arbeit wird dafür eine Lösung entwickelt. Die Grundlage bildet ein Informationsmodell, das managementrelevante Ressourcen durch Managementobjekte einheitlich repräsentiert. Dazu werden Managementobjektklassen und ihre Beziehungen untereinander festgelegt. Managementobjektklassen verfügen über öffentliche Eigenschaften, die in Form von Managementvariablen modelliert werden. Mit Hilfe von Statusvariablen kann sich der Manager über den Ressourcenzustand informieren, und mit Hilfe von Konfigurationsvariablen kann er auf den Ressourcenzustand einwirken. Das Management einer Multi-Provider-Umgebung erfordert den Einsatz eines Managementsystems, das den fehlerfreien Servicebetrieb sicherstellt. Dazu gilt es, die vom Informationsmodell festgelegten Managementobjekte zur Laufzeit bereitzustellen und zu verwalten. Die Umsetzung wird dadurch erschwert, dass nicht nur eine einzelne Managementarchitektur zum Einsatz kommt, sondern zumeist mehrere. Dies setzt den Einsatz einer Datenstruktur voraus, die zur Informationsintegration verschiedenste Datenquellen anbinden kann. Dadurch lässt sich die Heterogenität überwinden und eine einheitliche Sicht auf die Managementinformationen erzeugen. Zur Gewährleistung der nicht-funktionalen Eigenschaften bedarf es neben der kontinuierlichen Überprüfung der Zieleinhaltung auch des Einsatzes adaptiver Maßnahmen, um den sich abzeichnenden Zielverfehlungen entgegenzuwirken. Hierfür kommen Policy-Regeln zum Einsatz, die die Multi-Provider-Umgebung überwachen und steuern. Im Rahmen eines Anwendungsfalls wird der experimentelle Nachweis erbracht, dass sich nicht-interaktive Services auf Basis des Informationsmodells und der Policy-Regeln in einem Verbund von heterogenen Ausführungsumgebungen flexibel bereitstellen und gütegesichert erbringen lassen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Unternehmen stehen heute in einem globalen Wettbewerb und sehen sich einer verschärften Konkurrenzsituation ausgesetzt. Dies erfordert eine stetige Weiterentwicklung der Geschäftsmodelle und deren Prozesse. Dabei wird die digitale Transformation vermehrt als Chance betrachtet, um die Zukunftsfähigkeit der Unternehmen zu sichern. Ziel ist es, die Geschäftsprozesse agiler und flexibler zu gestalten, um schneller als bisher auf Veränderungen reagieren zu können. Die digitale Transformation kann letztlich aber nur dann gelingen, wenn das operative Geschäft den Wandel vorantreibt und wenn sie die Informationstechnologie (IT) mit Methoden und Lösungen unterstützt. Im Folgenden soll diese Thematik näher betrachtet werden. Zunächst gilt es, die Motivation der Arbeit darzustellen und das Thema in einen größeren Zusammenhang einzuordnen. Im Anschluss erfolgt eine Beschreibung der Problemstellung und Zielsetzung. Zudem werden der Forschungsbeitrag und die zentralen wissenschaftlichen Veröffentlichungen aufgeführt. Abschließend wird der Aufbau der Arbeit erläutert.

1.1 Motivation

Die zunehmende Digitalisierung unserer Geschäfts- und Alltagswelt stellt die Unternehmens-IT vor immer größer werdende Herausforderungen [Rot16]. Die Zeiten, in denen die IT ein Nischendasein fristete, sind längst vorüber. Heutzutage tragen IT-Systeme maßgeblich zum Unternehmenserfolg bei und sind ein wichtiger Bestandteil der Wertschöpfungskette [DS14]. Die immer kürzeren Produktzyklen sowie der stetig wachsende Konkurrenzdruck durch nationale und internationale Mitbewerber erfordern Innovationen und neue Lösungsstrategien, wenn sich ein Unternehmen langfristig am Markt behaupten will. Angebote und Dienstleistungen sind in immer kürzeren Zeitintervallen zu entwickeln und marktreif zu machen. Dabei lassen sich die Unternehmensziele nur noch durch automatisierte und qualitätsgesicherte Geschäftsprozesse erreichen. Die Unternehmen sind gezwungen, ihre Abläufe kontinuierlich zu optimieren und an veränderte Rahmen- und Marktbedingungen anzupassen [Poh09]. Die IT muss mit diesem Wandel Schritt halten, um die strategischen Zielsetzungen auch tatsächlich erreichen zu können. Dafür ist eine IT-Landschaft erforderlich, die sich flexibel aus unterschiedlichen Angeboten zusammenstellen lässt. Einzig eine direkt an den Geschäftsprozessen ausgerichtete Mischung

aus unternehmensinternen sowie -externen Anbietern kann diesen Anforderungen gerecht werden. Die Unternehmens-IT ist mehr denn je in der Verantwortung, die Innovationssprünge im Geschäftsmodell und in der Produktentwicklung zu unterstützen.

Cloud-Computing als Schlüsseltechnologie

Als ein strategisches Konzept bietet das Cloud-Computing Unternehmen die Möglichkeit, ihre IT-Landschaft bedarfsorientiert nach dem Baukastenprinzip zusammenzustellen. Es ist Aufgabe der IT-Verantwortlichen festzulegen, von welchen Anbietern sie welche Dienstleistungen in welchem Umfang beziehen möchten. Dadurch lassen sich bei gleichzeitiger Steigerung der Servicequalität Betriebskosten, Kapitalausgaben und Risiken oftmals deutlich senken. Beim Cloud-Computing steht Unternehmen ein hoch skalierbarer, nahezu unerschöpflicher Ressourcenpool zur Verfügung, der es möglich macht, die Unternehmens-IT annähernd unbegrenzt zu erweitern. In den seltensten Fällen wird aber ein einzelner Anbieter über ein passendes Leistungsspektrum verfügen, das sämtliche funktionalen und nicht-funktionalen Anforderungen abdeckt. Es geht vielmehr darum, für eine Anforderung das jeweils beste Angebot auszuwählen, um so eine IT-Landschaft zu definieren, die bestmöglich auf die eigenen Erfordernisse abgestimmt ist. Sollte sich ein Unternehmen auf nur wenige Anbieter beschränken und zumeist anbieterspezifische Services einsetzen, droht die Gefahr eines Vendor-Lock-in [RMVG+10, PRS17]. Dadurch geht die Möglichkeit verloren, Abläufe und Prozesse eigenverantwortlich zu gestalten. In Zeiten sich rasch ändernder Rahmen- und Marktbedingungen stellt dies eine Fehlentwicklung dar. Heutzutage gelten Wandlungsfähigkeit, Agilität und Flexibilität als drei wesentliche Erfolgsfaktoren für Unternehmen.

Die Zukunft gehört der Multi-Cloud

Die Konsequenz einer solchen Entwicklung ist der Einsatz einer Multi-Provider-Umgebung mit dem Ziel, Infrastrukturen, Plattformen und Anwendungen anbieterübergreifend frei kombinieren und austauschen zu können. Dieser Trend wird von einer Studie [HJS17] belegt, die Crisp Research im Auftrag der QSC AG durchgeführt hat. Dazu wurden mehr als 300 Entscheider deutscher Unternehmen unterschiedlicher Branchen und Größen hinsichtlich ihrer Cloud- und Multi-Cloud-Planungen befragt. Das Ergebnis ist in der Abbildung 1.1 dargestellt. Grundsätzlich wird Cloud-Computing für mehr als 80 % der Befragten zukünftig innerhalb der IT-Strategie eine tragende Säule darstellen. Für 17 % der mittleren und großen Unternehmen ist Cloud-

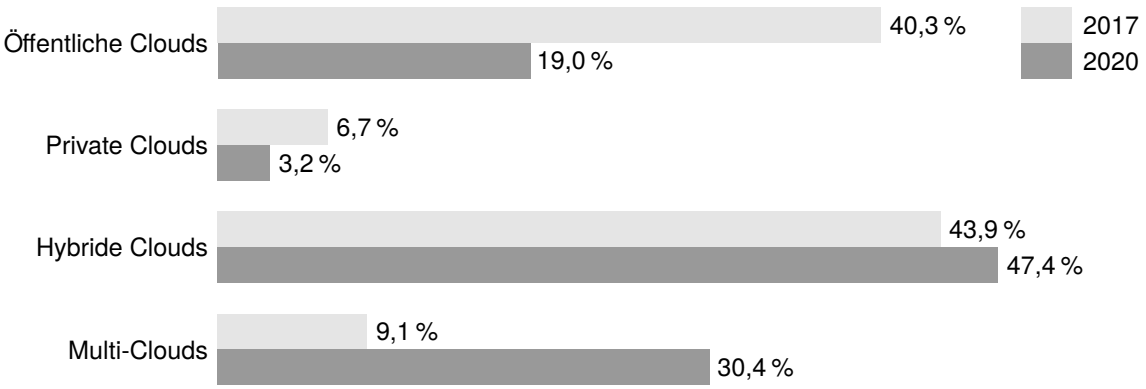


Abbildung 1.1: Favorisiertes IT-Betriebsmodell im Jahre 2017 und zukünftig

Computing bereits fester Bestandteil ihrer IT-Strategie und tagtäglich im Einsatz. Laut Umfrage wandern schon heute 41 % der IT-Budgets in den Aufbau moderner Cloud-Infrastrukturen, vor allem Multi-Cloud-Umgebungen. Knapp zwei Drittel (62 %) setzen eine Kombination aus öffentlichen und privaten Clouds ein oder planen deren Aufbau. Ziel bis 2020 ist für mehr als 70 % der Befragten der Einsatz einer Hybrid-/Multi-Cloud-Umgebung, in der unterschiedliche Lösungen zu einer ganzheitlichen Cloud-Architektur verbunden werden. Unternehmen profitieren somit vom Leistungsangebot und Innovationspotential aller sich auf dem Markt befindlichen Anbieter. Dadurch lassen sich Skalierbarkeit und Stabilität der eigenen Anwendungen und Prozesse mitunter deutlich verbessern. Die Neuausrichtung der IT-Landschaft zeigt sich auch darin, dass reine private und öffentliche Cloud-Lösungen zukünftig an Bedeutung verlieren werden. Derzeit stellen reine öffentliche oder hybride Cloud-Umgebungen noch die meist favorisierten IT-Betriebsmodelle dar, die zukünftige Entwicklung bewegt sich aber eindeutig in Richtung von Multi-Provider-Umgebungen.

1.2 Problemstellung

Derzeit stellt das Management einer Multi-Provider-Umgebung, die neben cloudbasierten auch virtuelle und physikalische Umgebungen umfasst, eine Herausforderung dar. Der Umgebungsverbund besteht aus Ausführungsumgebungen, die sich hinsichtlich Anbieter, Betriebsmodell, Technologie und Managementschnittstelle unterscheiden. Er muss sich jederzeit an das Marktangebot und die unternehmensspezifische Anbieterauswahl anpassen lassen. Dabei darf die Servicebereitstellung nicht auf nur eine einzelne Ausführungsumgebung beschränkt bleiben. Vielmehr müssen sich die an einer Multi-Provider-Umgebung beteiligten Ausführungsumgebungen flexibel und bedarfsgerecht einsetzen lassen. Dadurch können die Vorteile einer jeden Ausführungsumgebung ausgenutzt werden. Ziel dieser Arbeit ist die Entwicklung eines Lösungsansatzes, mit dem sich eine Multi-Provider-Umgebung einheitlich überwachen und steuern lässt. Dabei geht es insbesondere um die Gewährleistung von Servicegüteanforderungen hinsichtlich Bereitstellung und Betrieb von Services.

1.3 Forschungsbeitrag

Im Zentrum der Betrachtung steht die Überwindung der semantischen und technischen Heterogenität der Managementinformationen und Datenquellen. Der Forschungsbeitrag besitzt folgende Schwerpunkte:

Einheitliches Informationsmodell

Die Arbeit befasst sich mit dem Entwurf eines Informationsmodells für eine Multi-Provider-Umgebung. Ziel ist die Identifikation der managementrelevanten Ressourcen und ihre einheitliche Beschreibung durch Managementobjekte. In diesem Zusammenhang wird die Multi-Provider-Umgebung selbst durch ein Managementobjekt beschrieben, dem wiederum Managementobjekte untergeordnet sind, die die am Verbund beteiligten Ausführungsumgebungen repräsentieren. Im Rahmen von Projekten werden Ressourcen erfasst, die der Infrastruktur-, Plattform- und Anwendungsschicht entstammen. Das Informationsmodell basiert auf einem objektorientierten Modellierungsansatz und legt die Managementobjektklassen fest. Indem physikalische, virtuelle und cloudbasierte Umgebungen jeweils durch dieselbe Managementobjektklasse beschrieben werden, lässt sich die semantische Heterogenität in den Informati-

onsmodellen der Ausführungsumgebungen überwinden. Das Managementsystem verdeckt die technische Heterogenität und bietet eine vereinheitlichte Sicht auf die Managementinformationen.

Anforderungsgerechte Servicebereitstellung

Im Rahmen eines Anwendungsfalls wird ein Lösungsansatz experimentell erprobt, der es ermöglicht, nicht-interaktive Services in einer Multi-Provider-Umgebung anforderungsgerecht bereitzustellen. Darunter fallen Services, die sich während der Verarbeitung nicht mehr rekonfigurieren lassen. Die Ausführung terminiert entweder mit einem Ergebnis oder endet vorzeitig aufgrund eines Fehlers. Seitens des Service-Providers gilt es, in Abhängigkeit zu den Servicegüteanforderungen, einen Verarbeitungsknoten einzusetzen, der zur Ausführung des Services am geeignetsten erscheint. Bei der Entscheidungsfindung stehen Zeit- und Kostenaspekte im Vordergrund. Dementsprechend müssen Informationen über die Leistungsfähigkeit der Verarbeitungsknoten im Bezug zum Service vorliegen. Auf Grundlage dieser Daten kann der Managementprozess dann eigenständig entscheiden, ob und mit welchen Mitteln sich die Leistungsanforderungen erfüllen lassen.

Gütegesicherter Servicebetrieb

Zudem wird aufgezeigt, wie sich durch den flexiblen Einsatz der an einer Multi-Provider-Umgebung beteiligten Ausführungsumgebungen ein gütegesicherter Servicebetrieb realisieren lässt. Während der Ausführung eines Services können nicht alle Einflussgrößen kontrolliert werden, die sich negativ auf seine Leistungswerte auswirken. Grundsätzlich findet der Servicebetrieb immer unter gewissen Unsicherheiten und Risiken statt. Die vereinheitlichten Managementobjekte bilden die Grundlage einer autonomen Kontrollschleife zur Selbstadaption. Im Vordergrund steht die Servicemigration als korrektive bzw. perfektive Maßnahme. Dabei wird der Managementprozess ausschließlich von Policy-Regeln ausgeführt, die mit Hilfe einfacher Modifikationsfunktionen Änderungen an den Managementobjekten durchführen und dadurch die Multi-Provider-Umgebung steuern.

1.4 Forschungskooperation

Die Arbeit ist im Rahmen einer Forschungskooperation zwischen der Fachgruppe Rechnernetze und verteilte Systeme des Lehrstuhls IV für praktische Informatik der Technischen Universität Dortmund und dem Innovation-Center der Materna Information & Communications SE entstanden. Die Arbeitsgruppe hat gemeinsam in den internationalen ITEA/BMBF-Forschungsprojekten *OSAmI* (Open Source Ambient Intelligence Commons), *EASI-CLOUDS* (Extensible Architecture and Service Infrastructure for Cloud-aware Software), *BaaS* (Building-as-a-Service) und *Medolution* (Medical Care Evolution) wissenschaftliche Artikel und Konferenzbeiträge veröffentlicht. Eine zeitliche Einordnung der Forschungsprojekte ist in der Abbildung 1.2 dargestellt. Innerhalb des prämierten Forschungsprojekts EASI-CLOUDS (ITEA Award of Excellence in der Kategorie „Business Impact“, 2015) wurde ein Großteil der eigenen Forschungs- und Entwicklungsarbeiten geleistet. Im Zuge dessen sind das Informationsmodell, die Struktur- und Steuerungsmuster sowie das Managementsystem entstanden. Die erzielten (Teil-)Ergebnisse wurden in den folgenden Publikationen vorgestellt:

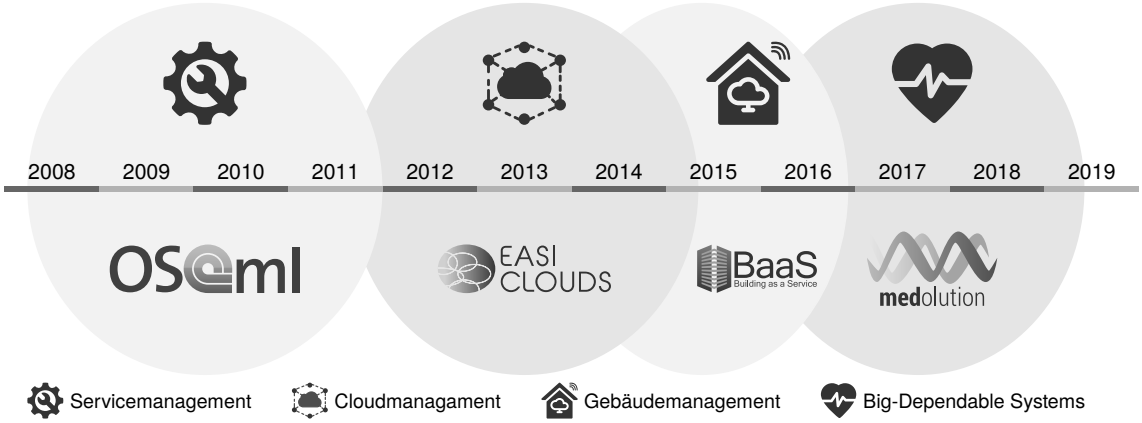


Abbildung 1.2: Zeitliche Einordnung der Forschungsprojekte

- *Scalable Monitoring System for Clouds* [BFL+13] auf der Konferenz *IEEE/ACM 6th International Conference on Utility and Cloud Computing* (UCC 2013)
- *Building a Medical Research Cloud in the EASI-CLOUDS Project* [FLT+15] in der Fachzeitschrift *Concurrency and Computation: Practice and Experience* (2015)

Die Arbeiten und Beiträge zum adaptiven Management sind insbesondere in die folgenden Publikationen eingegangen:

- *Lightweight Policy-Based Management of Quality-Assured, Device-Based Service Systems* [DKK+10b] auf der Konferenz *IEEE 24th International Conference on Advanced Information Networking and Applications* (AINA 2010)
- *Policy-Based Management for Resource-Constrained Devices and Systems* auf der Konferenz *IEEE International Symposium on Policies for Distributed Systems and Networks* [DKK+10a] (POLICY 2010)
- *Tool-Supported Refinement of High-Level Requirements and Constraints into Low-Level Policies* [DKK+11b] auf der Konferenz *IEEE International Symposium on Policies for Distributed Systems and Networks* (POLICY 2011)
- *Adaptive and Reliable Binding in Ambient Service Systems* [DKK+11a] auf der Konferenz *IEEE 16th International Conference on Emerging Technologies and Factory Automation* (ETFA 2011)
- *Networked Devices Meeting Dependability while Supporting the Medical Supervision of Cardiac Patients under Rehabilitation* [BSK+15] in der Fachzeitschrift *International Federation of Automatic Controls* (IFAC-PapersOnLine, 2015)
- *Rule-based technical management for the dependable operation of networked Building Automation Systems* [BVKF17] auf der Konferenz *IFIP/IEEE Symposium on Integrated Network and Service Management* (IM 2017)

Der Eigenanteil der unter Co-Autorenschaft entstandenen Publikationen umfasst das innerhalb des Promotionsvorhabens entwickelte policybasierte Managementsystem, die adaptiven Steuerungskonzepte zur Systemanpassung sowie die Grundbausteine des Informationsmodells.

1.5 Aufbau der Arbeit

Die Arbeit gliedert sich insgesamt in drei Teile. Der erste Teil befasst sich mit den informationstechnischen Grundlagen und dem Forschungsumfeld. Es werden Systemklassen, Ausführungsumgebungen und das technische Management vernetzter Systeme vorgestellt. Unter dem Gesichtspunkt eines einheitlichen Informationsmodells für eine Multi-Provider-Umgebung erfolgt eine Darstellung ausgewählter Forschungs- und Standardisierungsarbeiten. Der zweite Teil umfasst die Problemanalyse sowie den Lösungsansatz. Dabei werden die Ausgangssituation, die Zielsetzung und die Anforderungen erläutert. Im Anschluss folgt der Lösungsentwurf, bei dessen Darstellung zugleich auch die zentralen Herausforderungen und Neuerungen aufgeführt werden. Den Schwerpunkt der Arbeit bildet das Informationsmodell für das einheitliche Management einer Multi-Provider-Umgebung. Es wird eine Grundstruktur entworfen, auf deren Basis eine Umgebungs-, Projekt-, Servicegüte-, Funktions-, Policy-, Domänen-, Benutzer- und Berechtigungsmodellierung erfolgt. Die Managementobjekte bilden die Basis für die Spezifikation von Struktur- und Steuerungsmustern zur Kontrolle des Umgebungsverbunds. Die Managementinformationen gilt es, zur Laufzeit bereitzustellen und zu verwalten. Zu diesem Zweck wird ein Managementsystem entworfen, das eine Datenstruktur zur Informationsintegration sowie eine Laufzeitumgebung für den Überwachungs- und Steuerungsprozess umfasst. Der dritte Teil beinhaltet die experimentelle Erprobung des Lösungsansatzes anhand eines Anwendungsfalls, der dem Forschungsprojekt EASI-CLOUDS entstammt. Dabei wird aufgezeigt, wie sich nicht-interaktive Services mit Hilfe des Informationsmodells und der Steuerungsmuster gütegesichert erbringen lassen. Abschließend werden die Ergebnisse zusammengefasst und ein Ausblick für zukünftige Arbeiten gegeben.

2

Grundlagen

Das Kapitel befasst sich mit den informationstechnischen Grundlagen, die für das Management serviceorientierter Systeme in einer Multi-Provider-Umgebung relevant sind. Die nachfolgenden Abschnitte erläutern das thematische und begriffliche Umfeld der Arbeit. Zunächst werden Systemklassen und Architekturen behandelt. Den Schwerpunkt bilden verteilte, serviceorientierte und adaptive Systeme. Des Weiteren ist der Begriff der Multi-Provider-Umgebung festzulegen. Dazu werden Ausführungsumgebungen in Form physikalischer, virtueller und cloudbasierter Umgebungen betrachtet. Abschließend folgt eine Beschreibung des technischen Managements vernetzter Systeme.

2.1 Verteilte Systeme

Ein *verteiltes System* ist eine Ansammlung unabhängiger Computer, die den Benutzern wie ein einzelnes kohärentes System erscheinen [TS08]. Die Hardware- und Softwarekomponenten befinden sich auf vernetzten Computern und kommunizieren miteinander über den Austausch von Nachrichten [CDKB11]. Die Funktionalität des Gesamtsystems kann dabei nicht von nur einer einzelnen Komponente erbracht werden, sondern ergibt sich erst aus dem Zusammenwirken aller Komponenten. Eng damit verbunden ist der Begriff der verteilten Anwendung, die den eigentlichen Mehrwert eines verteilten Systems ausmacht und zumeist auf einer Middleware aufsetzt:

Verteilte Anwendung Eine *verteilte Anwendung* nutzt ein verteiltes System, um Anwendern eine in sich geschlossene fachliche Funktionalität zur Verfügung zu stellen [Ham05]. Die Anwendungslogik ist auf einzelne, weitestgehend unabhängige Softwarekomponenten aufgeteilt, die auf unterschiedlichen Verarbeitungsknoten ausgeführt werden. Eine verteilte Anwendung bezieht sich auf die Gesamtheit der Anwendungslogik und baut auf der Fähigkeit des verteilten Systems auf, Nachrichten zwischen Komponenten auszutauschen.

Middleware Eine *Middleware* unterstützt und erleichtert die Entwicklung einer verteilten Anwendung. Es handelt sich um eine Softwareschicht zwischen Betriebssystem, Netzwerk und Anwendung. Ihre Aufgabe besteht darin, die Anwendung von den Aspekten

der Netzwerkprogrammierung zu entkoppeln. Dazu stellt eine *kommunikationsorientierte Middleware* eine übergeordnete Kommunikations- und Koordinationsschicht zur Verfügung. Diese umfasst Protokolle zur Datenübertragung, Techniken zur Datentransformation und Mechanismen zur Fehlerbehandlung. Eine *anwendungsorientierte Middleware* erweitert eine kommunikationsorientierte Middleware um eine Laufzeitumgebung, ein Komponentenmodell und Funktionen zur Komponentenverwaltung.

2.1.1 Cluster-Computing

Unter einem *Cluster* versteht man eine Menge eigenständiger Rechner, die über ein lokales Hochgeschwindigkeitsnetzwerk miteinander verbunden sind und gemeinsam eine Funktion erbringen [BM05, TS08]. Die Rechner werden auch als *Knoten* bezeichnet. Sie lassen sich ausschließlich als Einheit ansprechen und nur im Verbund nutzen. Das Ziel des Cluster-Computings ist die Bereitstellung einer hohen Rechenleistung sowie einer ausfallsicheren Betriebsumgebung. Zu diesem Zweck kommen homogene und heterogene Cluster zur Anwendung. *Homogene Cluster* bestehen aus identischen Knoten, *heterogene Cluster* hingegen aus unterschiedlichen. Zudem existieren drei Arten von Cluster-Systemen:

Hochverfügbarkeitscluster Zur Steigerung der Verfügbarkeit sowie zur Verbesserung der Ausfallsicherheit werden *Hochverfügbarkeitscluster* gebildet. Sie sind ausgestattet mit Maßnahmen zur Fehlererkennung und -beseitigung. Typischerweise werden dazu redundante Knoten im *Hot-* und *Cold-Standby* [Ech90, SP92] eingesetzt.

Lastverteilungscluster In einem *Lastverteilungscluster* werden gleichartige Funktionen zur Leistungssteigerung mehrfach angeboten. Ein *Lastverteiler* (engl. *Load-Balancer*) ermittelt, basierend auf dem Auslastungsgrad der Knoten, denjenigen mit der voraussichtlich besten Performance zur Anfragebearbeitung.

Hochgeschwindigkeitscluster Der *Hochgeschwindigkeitscluster* ist auf spezielle Leistungsparameter optimiert. Dazu werden Anfragen in *Jobs* gegliedert und auf mehrere Knoten zur parallelen Abarbeitung verteilt.

2.1.2 Grid-Computing

Während ein Cluster eine begrenzte räumliche Ausdehnung hat und sich in der Regel über ein einzelnes Rechenzentrum erstreckt, ist ein *Grid* stark dezentralisiert. [FK03] definiert ein Grid als ein System, das verteilte Ressourcen unter Verwendung standardisierter offener Protokolle und Schnittstellen koordiniert und eine hohe Servicequalität bietet. Es umspannt mehrere administrative Verantwortungsbereiche, und die Kommunikation erfolgt über *Wide Area Networks* (WANs) anstelle von *Local Area Networks* (LANs). Während ein Cluster für einen bestimmten Einsatzzweck vorgesehen ist, übernehmen Grid-Knoten im Normalbetrieb andere Aufgaben. Sie können sich jederzeit in das Grid ein- bzw. ausklinken. Das dem Grid-Computing zugrundeliegende Geschäftsmodell ist kollaborativ, d. h. Teilnehmer müssen Ressourcen zum Grid beisteuern, um das Recht zur weiteren Ressourcennutzung zu erhalten. Neben der gemeinschaftlichen Verwendung von Rechenleistung können auch andere Ressourcen geteilt werden. Ein Grid lässt sich hinsichtlich seines Einsatzzwecks als *Rechen-*, *Daten-* oder *Ressourcengrid* klassifizieren. Zudem unterscheidet [FK03] die folgenden Anwendungsaspekte:

On-Demand Computing In vielen Fällen sind Ressourcen zur Problemlösung nur für einen begrenzten Zeitraum erforderlich. Dabei kann es sich um Speicherplatz, Rechenleistung

oder Anwendungen handeln. Ziel des *On-Demand Computings* ist es, diese Dienstleistung auf Anfrage bereitzustellen und über den gesamten Nutzungszeitraum hinweg in gleichbleibender Qualität zu erbringen.

Distributed Supercomputing Die Lösung komplexer Aufgaben erfordert insbesondere im Bereich der Wissenschaft und Forschung den Einsatz leistungsstarker Rechensysteme, sogenannter *Supercomputer*. Allerdings verursachen Anschaffung und Betrieb solcher Systeme enorme Kosten. Eine Lösung bietet das *Distributed Supercomputing*. Dabei werden Einzelsysteme, die nur über beschränkte Rechenkapazitäten verfügen, zu einem verteilten Supercomputer zusammengeschaltet.

High-Throughput Computing Lässt sich ein Problem in unabhängige Teilaufgaben zerlegen, ist es das Ziel des *High-Throughput Computings* (HTC), möglichst viele dieser Aufgaben in kurzer Zeit zu bearbeiten. Es entfällt die Notwendigkeit der gleichzeitigen Bereitstellung aller Ressourcen an zentraler Stelle. Der maximal erreichbare Durchsatz hängt maßgeblich vom Parallelisierungsgrad des Problems ab und kann bei gegebener Rechenleistung nicht beliebig gesteigert werden.

Data-Intensive Computing Komplexe Datenanalysen erfordern die Auswertung großer Datenmengen, die sich meist über geografisch verteilte Datenspeicher erstrecken. Eine vollständige Übertragung des Datenbestands auf einen zentralen Analyseknoten ist in der Regel entweder aus zeitlichen Gründen oder wegen Speichermangels nicht möglich. Das *Data-Intensive Computing* verwaltet einen verteilten Datenbestand, so dass sich Analysen und Suchanfragen dezentral durchführen lassen.

Collaborative Computing Das *Collaborative Computing* verbindet Benutzer und Applikationen zu Arbeitsgruppen und ermöglicht eine Echtzeit-Interaktion auf einem virtuellen Arbeitsfeld. Es zielt darauf ab, die Koordination und Kommunikation zwischen Menschen durch verteiltes Rechnen und den Einsatz von Multimedia-Technologien zu verbessern.

2.1.3 Utility-Computing

Das *Utility-Computing* entspricht einem Bereitstellungs- und Nutzungsmodell, bei dem sich IT-Services in gleichbleibend hoher Qualität jederzeit und nach Bedarf anfordern und verwenden lassen [Rap04]. Das Utility-Computing stellt eine Erweiterung des Grid-Computings dar und verfolgt das Ziel, IT-Services wie „Strom aus der Steckdose“ anzubieten. Zentrales Merkmal ist die verbrauchsorientierte Abrechnung der in Anspruch genommenen Leistung. Dieses Nutzungsmodell wird als *Pay-per-Use* (PPU) [Cus08] bezeichnet und setzt ein Preismodell voraus sowie die Möglichkeit der verbrauchsgenauen Kostenerfassung [YVCB10]. Als *Utility* bezeichnet man im Allgemeinen die von Unternehmen bereitgestellten Versorgungsgüter wie Gas, Strom, Wasser und Wärme. Nach [Rap04] besitzt ein Utility die folgenden Merkmale:

Erforderlichkeit Ein Utility dient der Befriedigung eines nutzerseitigen Grundbedürfnisses. Sein Fehlen wird als Minderung der Lebensqualität empfunden.

Zuverlässigkeit Die vom Utility bereitgestellte Dienstleistung muss jederzeit verfügbar und abrufbar sein. Fehler oder Ausfälle werden von Nutzern nicht akzeptiert.

Gebrauchsfreundlichkeit Unabhängig von der technologischen Komplexität muss die Nutzung des Utilitys für den Endanwender möglichst einfach sein.

Schwankende Nutzungsintensität Ein Utility wird je nach Nutzerkontext unterschiedlich häufig in Anspruch genommen, dabei ist die Nutzungsintensität nur schwer vorhersehbar. Die vorgehaltene Kapazität orientiert sich in der Regel an Maximalwerten, um auch Lastspitzen bedienen zu können.

Skalierbarkeit Utilitys entsprechen Versorgungsgütern und sind aus Sicht des Endanwenders unbegrenzt und jederzeit abrufbar. Dabei führt ein Überangebot zu sinkenden Kosten, eine Knappheit zu steigenden Kosten.

Exklusivität Staatliche Stellen können regionale Monopole festlegen, um die Versorgungssicherheit in bestimmten Gebieten zu gewährleisten.

2.2 Serviceorientierte Systeme

Nach dem Referenzmodell für *serviceorientierte Architekturen* (SOA) der *Organization for the Advancement of Structured Information Standards* (OASIS) ist ein *Dienst* (engl. *Service*) eine Komponente, die eine bestimmte Funktionalität über wohldefinierte Schnittstellen und unter Beachtung spezifizierter Randbedingungen und Richtlinien anderen Services oder Anwendungen zugänglich macht [MLM+06]. Services werden von einem *Serviceanbieter* (engl. *Service-Provider*) bereitgestellt und von einem *Servicenutzer* (engl. *Service-Consumer*) verwendet. Dazu veröffentlicht der Service-Provider seine Services bei einem *Servicevermittler* (engl. *Service-Broker*). Es handelt sich hierbei um einen Verzeichnisdienst, der eine Beschreibung aller veröffentlichten Services enthält [HS05]. In Abbildung 2.1 sind diese drei Rollen sowie ihre Interaktionsbeziehungen dargestellt. Nach [Erl05, BSI08] besitzt ein Service die folgenden Merkmale:

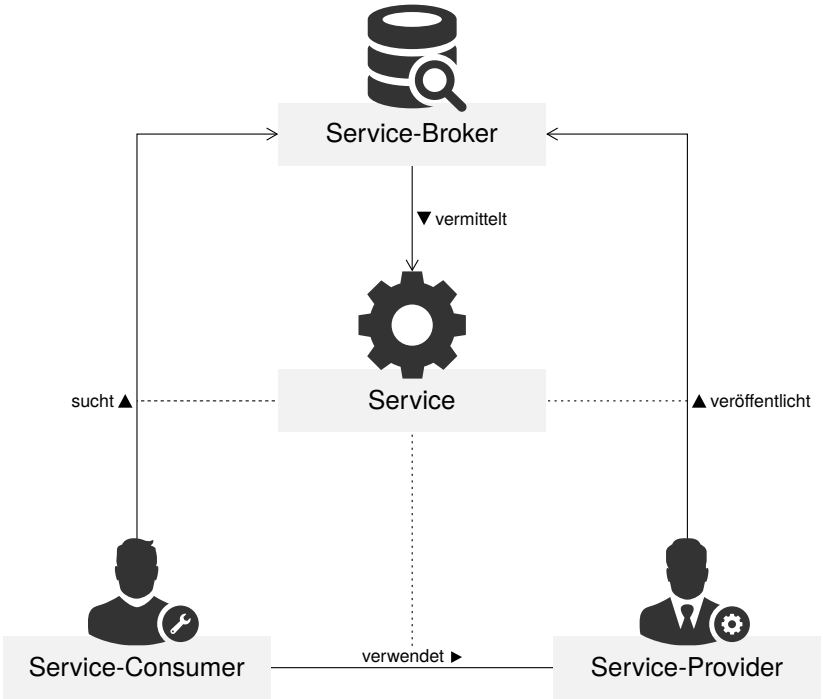


Abbildung 2.1: Rollen und Interaktion in einer serviceorientierten Architektur nach [Mel10]

Standardkonformität Damit Services unabhängig von ihrer Implementierung untereinander Nachrichten austauschen können, wird mit Hilfe offener Standards eine einheitliche Kommunikationsebene geschaffen. Der Mehrwert eines serviceorientierten Systems ergibt sich erst aus dem Zusammenwirken der beteiligten Services. Standards und die darauf basierende Interoperabilität bilden die Grundpfeiler eines jeden serviceorientierten Systems.

Vertrag Der Vertrag legt die Rahmenbedingungen der Servicenutzung fest und wird zwischen Nutzer und Anbieter geschlossen. Er beinhaltet eine Beschreibung der Funktion, der Schnittstelle sowie der nicht-funktionalen Eigenschaften. Die Übereinkunft wird als *Servicegütevereinbarung* (engl. *Service-Level-Agreement*, SLA) [PTDL07] bezeichnet. Sie enthält qualitative Anforderungen für Antwortzeiten, Durchsatzgeschwindigkeiten, Verfügbarkeiten und Wiederherstellungszeiten [OMB07, AGAF16].

Lose Koppelung Services gelten als lose gekoppelt, falls Abhängigkeits- und Nutzungsbeziehungen ausschließlich auf Informationen basieren, die in Serviceverträgen festgeschrieben sind. Dadurch können Services neu kombiniert und miteinander verschaltet werden. Zudem haben Änderungen nur lokale Auswirkungen, so dass sich Services und Implementierungen ohne größeren Aufwand austauschen lassen, ohne das gesamte Beziehungsgeflecht zu beeinflussen. Das Prinzip der losen Koppelung folgt dem Minimalitätsgrundsatz, wobei ein Service gerade über so viele Abhängigkeiten verfügen soll, wie zur Leistungserbringung minimal erforderlich sind. Gleichwohl gilt es, mit möglichst wenig Annahmen über den Ausführungskontext auszukommen.

Abstraktion Die Abstraktion ist eine Grundvoraussetzung der losen Koppelung und hat zum Ziel, möglichst viele Details eines Services vor den Nutzern zu verbergen. Ein Service wird als eine Black Box betrachtet, wobei für dessen Nutzung kein Wissen über die interne Struktur und Arbeitsweise erforderlich ist. Mit Hilfe eines Vertrags lässt sich von der Technologie, der Logik und der Funktionsumsetzung abstrahieren und die Sichtbarkeit für die Nutzer einschränken. Aspekte, die nicht Teil des Vertrags sind, können dementsprechend auch nicht vorausgesetzt werden.

Wiederverwendbarkeit Services bieten eine in sich abgeschlossene Funktionalität und müssen so beschaffen sein, dass sie sich in unterschiedlichen Anwendungsszenarien einsetzen lassen. Des Weiteren erlaubt dieses Grundprinzip eine stetige Verbesserung der Serviceimplementierung, da sich bereits vorhandene Funktionen wiederverwenden lassen.

Zustandslosigkeit Services folgen dem Dienstleistungsgedanken und erbringen eine vordefinierte Funktion. Sie sind zustandslos, so dass alle zur Ausführung erforderlichen Daten beim Aufruf zu übergeben sind. Die Verwaltung von Zustandsinformationen könnte ansonsten die Leistungsfähigkeit eines Services beeinträchtigen und gleichzeitig seine Skalierbarkeit einschränken.

Auffindbarkeit Unter der Auffindbarkeit von Services versteht man die Möglichkeit, gezielt nach Services suchen zu können. Grundlage dafür sind Verzeichnisdienste und Discovery-Mechanismen, die Such- und Abfragemöglichkeiten bereitstellen. Dies erfordert den Einsatz von Metadaten, die Informationen über die nicht-funktionalen Eigenschaften eines Services enthalten. Dadurch ist eine Eignungsprüfung hinsichtlich des Einsatzzwecks möglich.

Orchestrierbarkeit Bei der Orchestrierung lassen sich zusammengesetzte Services definieren. Dabei werden Services aus fachlichen Gesichtspunkten heraus zu größeren Einheiten verbunden. Die dadurch entstandenen *Services höherer Ordnung* lassen sich wiederum zu komplexen Prozessen verschalten.

Eine SOA stellt ein abstraktes Architekturkonzept dar und keine konkrete Umsetzung. Zentrales Prinzip ist das Anbieten, Suchen und Nutzen von Services. Es lassen sich Anwendungssysteme mit hoher Flexibilität und Verfügbarkeit erstellen. Dazu wird die Anwendungsfunktion in einzelne Services zerlegt, die mit Hilfe unterschiedlicher Programmiersprachen implementiert und auf verteilten Verarbeitungsknoten bereitgestellt werden.

Architekturstile

Im Wesentlichen existieren zur Realisierung eines serviceorientierten Systems drei Architekturstile [ST07]. Beim *schnittstellenorientierten Stil* liegt der Fokus auf der Schnittstelle und ihren Operationen, beim *nachrichtenorientierten Stil* stehen die ausgetauschten Informationen im Vordergrund und beim *ressourcenorientieren Stil* liegt der Schwerpunkt auf zustandslosen Ressourcen, die sich eindeutig identifizieren und adressieren lassen. Dieser Architekturstil wird auch als *ressourcenorientiere Architektur* (ROA) oder *Representational State Transfer* (REST) bezeichnet und geht zurück auf die Dissertation [Fie00] von Roy Fielding aus dem Jahr 2000.

2.3 Adaptive Systeme

Die zunehmende Komplexität von Softwaresystemen [Lil08] erschwert die Weiterentwicklung und die Wiederverwendung von (Teil-)Funktionen [SBD+10, VCD+13]. Mit Hilfe einer service- und komponentenorientierten Softwareentwicklung lassen sich zwar Fortschritte erzielen, sie reichen aber noch nicht aus, um den gestiegenen Anforderungen an Robustheit, Stabilität und Flexibilität gerecht zu werden [PTDL07]. In diesem Zusammenhang bietet sich der Einsatz adaptiver Systeme ein. Einen umfassenden Einstieg in das Thema bieten die beiden Arbeiten von [Kla04] und [Spr04]. Diese bilden zugleich auch die Grundlage für die nachfolgenden Ausführungen.

2.3.1 Adaption

Unter *Adaption* wird ein Vorgang verstanden, der sämtliche Maßnahmen umfasst, um den Systemzustand zielgerichtet an die verfügbaren Ressourcen, die gestellten Anforderungen und den umgebenden Ausführungskontext anzupassen [EHS+11]. Wie [Kla04] herausstellt, muss hierbei genau genommen von einem *Adaptionsprozess* gesprochen werden. Dieser ist in der Abbildung 2.2 dargestellt und hat folgende Bestandteile:

Adaptionssubjekt Die Adaption wird entweder von einem Menschen oder vom System selbst initiiert. Abhängig vom Automatisierungsgrad führt das System daraufhin die Operationen eigenständig durch oder sie unterliegen ganz oder zumindest teilweise der Verantwortung eines Menschen.

Adaptionsobjekt Beim Gegenstand der Adaption kann zwischen der Adaption der Inhalte, der Daten, der Kommunikation sowie der Softwarekomponenten unterschieden werden [Spr04]. Es besteht die Möglichkeit, Objekte im Bezug zur Systemsicht hinzuzufügen, anzupassen, auszutauschen und zu entfernen.

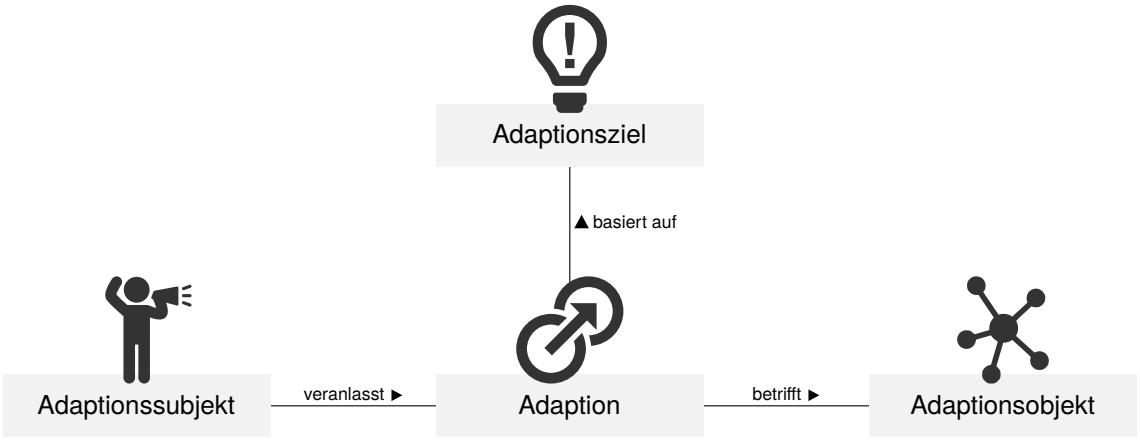


Abbildung 2.2: Schema eines Adaptionsprozesses nach [Kla04]

Adaptionsziel Anpassungen erfolgen stets zielgerichtet und müssen sich objektiv begründen lassen. Auf Grundlage einer Adaptionsstrategie wird in Abhängigkeit zum aktuellen Systemzustand und dem zeitlichen Verlauf entschieden, ob steuernd auf das System einzuwirken ist.

Ein System heißt *adaptierbar*, falls man es von außen statisch oder dynamisch auf die Anforderungen des Nutzungsszenarios abstimmen kann. Die *Adaptierbarkeit* eines Softwaresystems ist ein Maß für den Umfang und die Einfachheit, mit der sich ein System anpassen lässt. Ein System heißt *adaptiv*, wenn es sich selbstständig an veränderte Bedingungen anpasst. In diesem Zusammenhang wird auch von *selbstadaptiven* Systemen gesprochen, um zu betonen, dass das System selbst handlungsaktiv ist und den Adaptionsprozess eigenständig durchführt. Die *Adaptivität* setzt Adaptierbarkeit voraus und gilt als Maß dafür, inwieweit ein System im Vergleich zu nicht-adaptiven oder anderen adaptiven Systemen über Adaptionseigenschaften verfügt. Grundvoraussetzung für die Adaption ist die Bereitstellung von Zustandsinformationen. Die Gewinnung, Verarbeitung und Speicherung dieser Daten ist Teil des Forschungsgebiets *Context-Awareness* [ADOB98, ADB+99]. Der *Kontext* repräsentiert einen Teilausschnitt aus der Realität und macht implizit vorhandene Informationen zur maschinellen Verarbeitung sichtbar. Zum Kontext zählen sämtliche Informationen, die sich dazu nutzen lassen, um Situationen von Entitäten zu beschreiben [ADB+99]. Entitäten können sowohl reale als auch logische Objekte repräsentieren.

2.3.2 Self-X-Eigenschaften

Der Begriff der Self-X-Eigenschaften wurde von IBM im Zusammenhang mit autonomen Systemen geprägt [IBM06]. Sie umfassen grundlegende Eigenschaften, die Voraussetzung für ein adaptives Systemverhalten sind. Ursprünglich spezifizierte IBM die sogenannten *CHOP-Funktionen*: Selbstkonfiguration (engl. Self-Configuration), Selbstheilung (engl. Self-Healing), Selbstoptimierung (engl. Self-Optimization) und Selbstschutz (engl. Self-Protection). Im Laufe der Zeit kamen weitere Self-X-Eigenschaften wie Selbstorganisation (engl. Self-Organization) und Selbstbewusstsein (engl. Self-Awareness) hinzu:

Selbstkonfiguration Die Selbstkonfiguration ermöglicht es dem System, sich zur Laufzeit eigenständig an veränderte Situationen und Umgebungsbedingungen anzupassen. Dazu

müssen Veränderungen erkannt, analysiert und bewertet werden. Basierend auf einer Strategie passt der Adaptionsprozess den Systemzustand entsprechend an.

Selbstheilung Selbstheilende Systeme erkennen, analysieren und reagieren auf Ausfälle selbstständig und verbessern dadurch die Robustheit und die Verfügbarkeit des Systems. Fehlerhaft arbeitende Komponenten werden identifiziert, isoliert und nach Bedarf entfernt, angepasst oder ersetzt.

Selbstoptimierung Selbstoptimierende Systeme suchen kontinuierlich nach Möglichkeiten, ihre Eigenschaften zu verbessern. Ziel ist es, die zur Verfügung stehenden Ressourcen bestmöglich einzusetzen. Zielvorgaben beziehen sich zumeist nicht auf direkt messbare Größen, sondern auf unbekannte Extremwerte von Qualitätsmaßen. Es existiert kein fester Sollwert, vielmehr muss das System den Wert eigenständig ermitteln und beibehalten.

Selbstschutz Selbstschützende Systeme erkennen unautorisierte Zugriffe und wehren sich gegen Eindringlinge. Sie verteidigen das System gegen interne und externe Angriffe und verhindern eine Datenkorruption. Fehlerfälle oder kritische Situationen, die die Sicherheitsziele des Systems verletzen, werden erkannt bzw. vor ihrem Eintreten verhindert.

Selbstorganisation Das System organisiert sich selbst ohne Einfluss einer äußeren steuernden Instanz. Die internen Abläufe ergeben sich automatisch aus den Wechselwirkungen der eingesetzten Komponenten.

Selbstbewusstsein Das System ist sich seiner selbst, seiner Fähigkeiten sowie seines Umfelds bewusst und bezieht diese Informationen in die interne Ablaufplanung mit ein. Das Wissen umfasst die verfügbaren Komponenten, den eigenen Zustand, die bestehenden Verbindungen zu anderen Systemen und die nutzbaren Ressourcen.

2.3.3 Ziele

Gründe für System- und Softwareanpassungen sind vielfältig und in der Regel spezifisch für das jeweilige Adaptionsobjekt. Es lassen sich aber aus dem Bereich der *Softwarewartung* vier grundlegende Zielsetzungen herausstellen. [Swa76] benennt in diesem Zusammenhang die Ziele der korrektiven, perfektiven und adaptiven Adaption. Später überträgt [KBC02] diese Aspekte auf den Bereich der automatisierten Adaption komponentenorientierter Systeme und ergänzt sie um das Ziel der erweiternden Adaption. Die vier Adaptionsziele lassen sich wie folgt beschreiben:

Korrektive Adaption Ziel der korrektiven Adaption ist die Beseitigung einer Fehlfunktion. Tritt im System ein unerwünschtes Verhalten auf, so werden die dafür verantwortlichen Komponenten ermittelt und die Situation unter Anwendung eines Adaptionsmechanismus aufgelöst. Maßnahmen zur korrektiven Adaption müssen nicht auf den Problemverursacher beschränkt sein, sondern umfassen alle systemweit durchgeführten Schritte zur Fehlerkorrektur. Es handelt sich um eine reaktive Intervention, die erst dann erfolgt, wenn eine Fehlersituation vorliegt.

Perfektive Adaption Die perfektive Adaption dient der Optimierung einer bisher ordnungsgemäß funktionierenden Systemfunktion auf der Grundlage eines berechenbaren Qualitätsmaßes. Maßnahmen zur perfektiven Adaption umfassen den Einsatz verbesserter Algorithmen sowie die Vervielfältigung von Komponenten zur Lastverteilung.

Erweiternde Adaption Bei der erweiternden Adaption wird das System um zusätzliche Funktionen erweitert, die während seiner Entwicklungszeit noch nicht erforderlich waren. Vielfach sind veränderte Anforderungen und Rahmenbedingungen Auslöser entsprechender Maßnahmen. Dazu müssen dem System die zu integrierenden Komponenten bekannt gemacht werden. Des Weiteren ist eine Spezifikation erforderlich, die die übergeordnete Funktion beschreibt, die durch das Zusammenwirken dieser Komponenten erbracht wird. Eine Steigerung der erweiternden Adaption liegt vor, wenn das System die Fähigkeit besitzt, Funktionen je nach Systemzustand dynamisch zu aktivieren und zu deaktivieren.

Adaptive Adaption Mittels adaptiver Adaption reagiert das System auf Änderungen seiner Laufzeitumgebung. Es passt sich seiner Umwelt an. Auslöser dieser Art von Adaption ist immer der Ausführungskontext. Die Grenze zwischen korrektiver, perfektiver und erweiternder Adaption ist unscharf, da häufig Änderungen im Kontext unmittelbar Auswirkungen auf den Systemzustand haben. Dies kann Fehlersituationen oder Beeinträchtigungen der Systemfunktionen zur Folge haben, auf die das System entsprechend reagieren muss.

2.3.4 Strategien

Eine erste grundlegende Klassifikation von Adaptionsstrategien erfolgt in [Sat96]. Demnach lässt sich, wie die Abbildung 2.3 veranschaulicht, eine Adaptionsstrategie in einen Bereich einordnen, der durch zwei Extrema begrenzt wird:

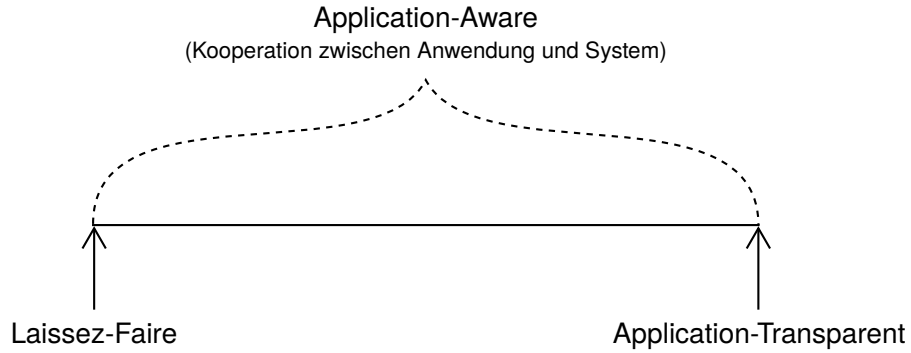
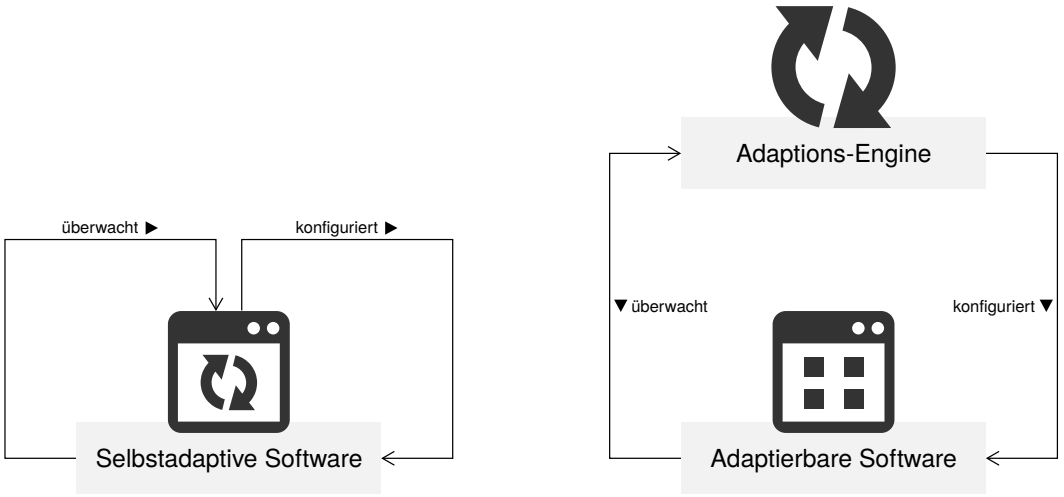


Abbildung 2.3: Adaptionsstrategien nach [Sat96, Spr04]

Laissez-Faire Die Anwendung ist ausschließlich selbst für ihre Adaption zuständig. Dieser Ansatz hat den Vorteil, dass keine zentrale Steuerung erforderlich ist und sich Entscheidungen zur Adaption dezentral treffen lassen. Eine Optimierung der Ressourcenvergabe zwischen konkurrierenden Zugriffen sowie eine übergeordnete Koordination ist nicht möglich. Des Weiteren erschwert dieser Ansatz den Entwicklungsprozess, da sich jede Anwendung selbstständig Informationen über den Systemzustand beschaffen muss. Die Planung des adaptiven Verhaltens erfolgt gänzlich zum Erstellungszeitpunkt und setzt voraus, dass alle zur Adaption erforderlichen Datenquellen bekannt sind.

Application-Transparent Die Verantwortung zur Adaption liegt ausschließlich bei dem System. Anwendungen müssen dazu nicht verändert werden, sondern können um ein adaptives Verhalten angereichert werden. Konkurrierende Ressourcenzugriffe lassen sich zentral unter Beachtung von Optimierungsvorgaben auflösen.

Adaptionsstrategien, die zwischen diesen beiden Extrema angesiedelt sind, setzen auf eine Kooperation zwischen Anwendung und System und werden auch als *anwendungszentrisch* [Sam02] oder *kollaborativ* bezeichnet. Auf diese Weise lassen sich die Vorteile beider Ansätze miteinander kombinieren. Adaptionsmaßnahmen, die ausschließlich anwendungsinternes Wissen erfordern, können direkt in den Applikationscode integriert werden. Darüber hinaus stellen Sensoren und Effektoren managementrelevante Zustände und Operationen zur Verfügung, die sich vom Adaptionsprozess zur koordinierten Systemanpassung verwenden lassen. Die Klassifikation nach [ST09] baut auf diesem Ordnungsprinzip auf, unterscheidet aber stattdessen zwischen *interner* und *externer Adaption*. Die Abbildung 2.4 zeigt diese beiden Adaptionsarten:



(a) Interne Adaption **(b)** Externe Adaption

Abbildung 2.4: Klassifikation von Adaptionsstrategien nach [ST09]

Interne Adaption Die Applikations- und Adaptionslogik sind, wie die Abbildung 2.4a zeigt, untrennbar miteinander verbunden. Der Ansatz der internen Adaption nutzt Eigenschaften der Programmiersprache in Form konditionaler Ausdrücke, Parametrisierungen sowie Möglichkeiten der Fehlerbehandlung [OGT+99, FHS+06]. Die für den Adaptionsprozess erforderlichen Sensoren und Effektoren sind Bestandteil des Programmcodes und müssen bereits zur Entwicklungszeit integriert werden. Lösungen nach dem Prinzip der internen Adaption bieten sich überall dort an, wo sich Adaptionsentscheidungen ausschließlich auf Grundlage lokal verfügbarer Zustandsinformationen treffen lassen. Als Beispiel sei die Anpassung der Fenstergröße des *Transmission Control Protocol* (TCP) an die aktuelle Lastsituation [APB09] genannt.

Externe Adaption Der Adaptionsprozess und die Adaptionslogik werden von einer externen Komponente außerhalb der Applikation implementiert. Die Abbildung 2.4b enthält dazu eine von der adaptierbaren Software unabhängige Adaptions-Engine, die Zustandswerte sammelt, analysiert und nach Bedarf steuernd eingreift. Lösungen der externen Adaption basieren häufig auf einer Middleware [FHS+06, Gei08] und erweitern diese um adaptive Funktionen. Dazu wird die Adaptionslogik auf Komponenten aufgeteilt und entsprechend bereitgestellt. Die Herausforderung besteht darin, die Komponenten zu verwalten, zu koordinieren und ihnen Zustandswerte zur Verfügung zu stellen, auf deren Grundlage sich Adaptionsentscheidungen treffen lassen. Der externe Ansatz hat den Vorteil, dass

sich die Adaptions-Engine und die Adaptionslogik wiederverwenden lassen. Des Weiteren ermöglicht der Austausch der Adaptionslogik eine Anpassung des Systemverhaltens an veränderte Anforderungen und Bedingungen.

In diesem Zusammenhang spricht [DC01] auch von *impliziter Adaption* und *expliziter Adaption*. Der Unterschied zur internen bzw. externen Adaption ist gering. Bei der internen bzw. externen Adaption wird die Richtung betont, von der die Initiative zur Adaption ausgeht, wohingegen bei der impliziten bzw. expliziten Adaption hervorgehoben wird, auf welche Weise die Adaption erfolgt und inwiefern sie von außen beeinflussbar ist. Darauf aufsetzend kann die Anpassbarkeit der Adaptionslogik selbst betrachtet werden. Es lassen sich folgende Fälle unterscheiden:

Statische Entscheidungsfindung Die Adaptionslogik wird zur Entwicklungszeit entworfen und implementiert. Zur Laufzeit folgt dann der Adaptionsprozess dem vorgegebenen Ablauf. Das adaptive Systemverhalten ist festgeschrieben und vorhersehbar. Anpassungen lassen sich anhand des Codes erklären und nachvollziehen. Nachteilig ist, dass alle Situationen, die Anpassungen erfordern, zur Entwicklungszeit explizit definiert und mit Reaktionen versehen sein müssen. Das System verfügt somit über eine Menge vorgefertigter Verhaltensmuster, die je nach Situation zur Anwendung kommen.

Dynamische Entscheidungsfindung Das adaptive Systemverhalten wird erreicht, indem Mechanismen aus der künstlichen Intelligenz eingesetzt werden. Das System erlernt Verhaltensmuster selbstständig und führt Anpassungen eigenständig durch. Je nach Lernstrategie können fehlerhafte oder unvollständige Muster revidiert und korrigiert werden. Das adaptive Verhalten solcher Systeme lässt sich nur schwer vorhersehen. Die Lernhistorie und der vergangene Systemablauf beeinflussen seinen „Erfahrungshorizont“ und wirken sich auf das Adaptionsverhalten zu einem späteren Zeitpunkt aus.

Adaptive Systeme können entweder *geschlossen-adaptiv* oder *offen-adaptiv* sein. Bei einem geschlossen-adaptiven System kann das Adaptionsverhalten nachträglich nicht mehr verändert werden. Bei einem offen-adaptiven System hingegen lassen sich Anpassungen und Erweiterungen der Adaptionslogik jederzeit vornehmen. Eine Steigerung der offen-adaptiven Eigenschaft wird erreicht, wenn nicht der Mensch, sondern die Software selbst neue Adaptionsregeln einbringt. Eine solche Software gilt als *lernfähig*. Systeme, die auf dem Prinzip der dynamischen Entscheidungsfindung basieren, sind in jedem Fall offen-adaptiv. Im Falle einer statischen Entscheidungsfindung muss genauer differenziert werden. Lässt sich die Adaptionslogik austauschen, so kann ebenfalls von einem offen-adaptiven System gesprochen werden. Erlaubt die Adaptionslogik aber keinen Austausch, ist das System geschlossen-adaptiv. In beiden Fällen bestimmen vorgefertigte Reaktionen das adaptive Systemverhalten. In Anlehnung an [End94] und [WF02] ist eine genauere Klassifikation möglich. Demnach lassen sich die *programmierte*, *spontane* und *selbstorganisierte Rekonfiguration* unterscheiden:

Programmierte Rekonfiguration Das adaptive Systemverhalten wird bereits zur Entwicklungszeit vollständig festgelegt, indem die kritischen Situationen vorab definiert werden, die den Einsatz adaptiver Maßnahmen erforderlich machen. Diese Eingriffe werden anschließend in Form von ausführbaren Regeln nach dem Prinzip von *Event-Condition-Action* (ECA) [DGG95] spezifiziert. Dabei stößt ein Ereignis die Überprüfung eines logischen Ausdrucks an und löst gegebenenfalls die Ausführung der „programmierten“ Aktion aus.

Spontane Rekonfiguration Das System verfügt zwar über Adaptionsmechanismen, die aber im Gegensatz zur programmierten Rekonfiguration nicht automatisiert zum Einsatz kommen. Die Ausführung erfolgt ausschließlich auf Initiative des Nutzers und setzt eine entsprechende Anfrage voraus. Grundsätzlich ist das adaptive Verhalten solcher Systeme zur Entwicklungszeit nicht vorhersehbar, da unbekannt ist, wann welche Adaptionsmaßnahme zur Anwendung kommt.

Selbstorganisierte Rekonfiguration Adaptionsmaßnahmen werden nicht in Form von Programmcode als feste Operationssequenz implementiert, vielmehr werden zur Entwicklungszeit ein Systemmodell und Verhaltensmuster beschrieben. Invarianten in Form von Zielvorgaben legen die Grenzen der Rekonfigurationen fest. Das System ist zur Laufzeit dafür verantwortlich, die zur Aufrechterhaltung des Sollzustands notwendige Operationssequenz abzuleiten und auszuführen [BLMR04].

Bezogen auf die Richtung des Informationsflusses zwischen Komponente und Adaptions-Engine lässt sich die *ereignisgesteuerte* von der *anfragebasierten Adaption* unterscheiden. Bei der ereignisgesteuerten Adaption informieren die Komponenten von sich aus die Adaptions-Engine nach dem Push-Prinzip über Zustandsänderungen. Bei der anfragebasierten Adaption geht die Initiative hingegen von der Adaptions-Engine aus [DL03]. Sie fragt die Zustandsinformationen nach dem Pull-Prinzip periodisch ab und überprüft sie auf Änderungen. Je nach Ausrichtung der Adaptions-Engine lässt sich ferner zwischen *spezifischer* und *generischer Adaption* unterscheiden. Eine spezifische Adaption liegt vor, wenn sich Adaptionen auf spezielle Anwendungsdomänen wie beispielsweise Datenbanken beschränken. Generische Lösungen sind unabhängig von der Anwendungsdomäne einsetzbar und bieten wiederverwendbare Adaptionsmechanismen. Des Weiteren kann eine Adaption bezogen auf ihren temporalen Aspekt klassifiziert werden. In Abhängigkeit vom Zeitpunkt der Anpassung lässt sich zwischen *reaktiver* und *proaktiver Adaption* unterscheiden:

Reaktive Adaption Erst beim Eintritt einer unerwünschten Situation, werden Anpassungen vorgenommen. Adaptionen erfolgen als Reaktion auf einen als kritisch klassifizierten Systemzustand. In den meisten Fällen verfolgen sie das Ziel, korrektiv auf das System einzuwirken, um einen als sicher geltenden Zustand herbeizuführen.

Proaktive Adaption Anpassungen erfolgen initiativ auf der Grundlage einer Vorausplanung. Ziel ist es, die Systementwicklung frühzeitig bereits dahingehend zu beeinflussen, dass eine als kritisch eingestufte Situation erst gar nicht entsteht.

Adaptive Systeme erfordern eine möglichst lückenlose Erfassung des Systemzustands. Auf dieser Grundlage lassen sich Werte aggregieren und Zustände abstrahieren. Dazu ist ein Überwachungsmechanismus erforderlich, der *kontinuierlich* oder *adaptiv* arbeitet. Eine kontinuierliche Überwachung bedeutet eine ununterbrochene vollständige Datensammlung. Im Gegensatz dazu arbeitet ein Überwachungsmechanismus adaptiv, wenn nicht fortwährend alle Werte, sondern lediglich ein Ausschnitt erfasst und verarbeitet wird. Dieser Ausschnitt lässt sich dann nach Bedarf ausweiten und wieder einschränken.

2.3.5 Steuerung

Das übergeordnete Ziel einer jeden Adaption ist die Gewährleistung von Robustheit und Stabilität [Men00, Lad01]. Adaptive Systeme zeichnen sich dadurch aus, dass Teile des Entscheidungsprozesses von der Entwicklungszeit in die Laufzeit verlagert werden. Auf Grundlage

des Systemzustands und der Ausführungsumgebung entscheidet das System selbst, wie sich unter den vorherrschenden Bedingungen ein bestmögliches Verhalten erreichen lässt. Prinzipien aus der Regelungstechnik helfen dabei, diese Anpassungsfähigkeit zu realisieren. Zu diesem Zweck kommen Regelkreise zum Einsatz, die – unabhängig ob sie aus elektronischen, mechanischen oder biologischen Übertragungssystemen zusammengesetzt sind – stets aus den folgenden Elementen bestehen:

- Regelstrecke (Strecke)
- Messeinrichtung (Messglied)
- Vergleichseinrichtung (Vergleicher)
- Regeleinrichtung (Regler)
- Stelleinrichtung (Stellglied)

Regelkreise führen vier grundlegende Operationen durch: die Datensammlung (Collect), die Datenanalyse und -aufbereitung (Analyze), die Lösungsentwicklung (Decide) und deren Umsetzung (Act). Sensoren erfassen den System- und Umgebungszustand. Die gesammelten Daten werden aufbereitet, gefiltert und zusammengeführt. In der Diagnosephase erfolgt eine Trend-, Fehler- und Symptomerkennung. Dazu wird der Systemzustand analysiert, bewertet und seine zukünftige Entwicklung abgeschätzt. Das Ergebnis führt gegebenenfalls zu einem steuernden Eingriff, indem mittels Effektoren auf das System eingewirkt wird. Der Regelkreis, der im Zusammenhang mit adaptiven Systemen auch als *autonomer Regelkreis* [DDF+06] oder *autonome Kontrollschleife* bezeichnet wird, ist in der Abbildung 2.5 dargestellt. Er entspricht einer Verallgemeinerung des aus der künstlichen Intelligenz stammenden Prinzips „Sense-Plan-Act“ [BFH84, AML89] zur Verhaltensbeschreibung eigenständig agierender Entitäten. Eine

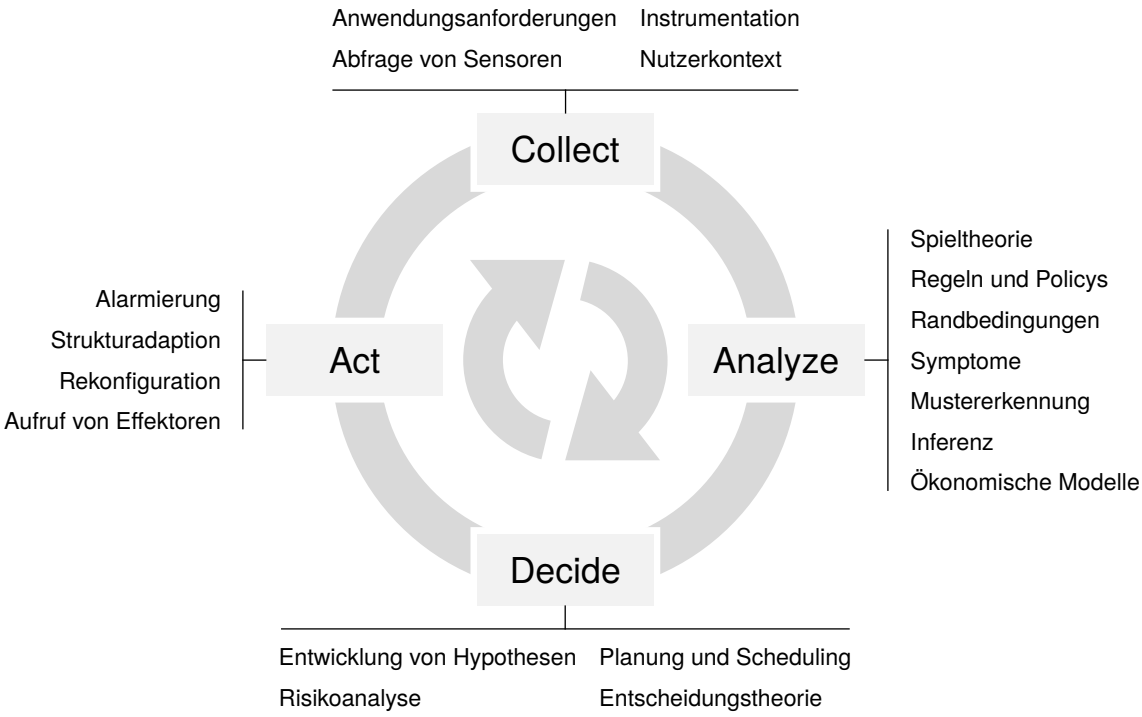


Abbildung 2.5: Autonomer Regelkreis nach [DDF+06]

solche Entität nimmt demnach eine Information auf (Sense), verarbeitet sie zielführend (Plan) und setzt anschließend die zuvor geplanten Schritte um (Act). Typischerweise besteht ein adaptives System aus einer Vielzahl von Regelkreisen, die untereinander in Beziehung stehen. Die Planung dieser Abhängigkeiten stellt beim Systementwurf eine Herausforderung dar.

2.3.5.1 Regelkreise in der Softwaretechnik

In der Softwaretechnik werden Regelkreise vielfältig eingesetzt. Sie bleiben aber bei der Systembeschreibung oftmals verborgen, so dass sich ihre Anwendung lediglich erahnen lässt [MPS08]. Entwurf- und Spezifikationssprachen wie die *Unified Modeling Language* (UML) bieten nur wenig Unterstützung bei der Beschreibung von Regelkreisen. Die rückkopplungsbasierte Steuerung ist aber ein grundlegendes Funktionsprinzip adaptiver Systeme und wirkt sich zumeist auf die gesamte Architektur aus. Der Einsatz von Regelkreisen bestimmt maßgeblich die Systemeigenschaften, wobei die fehlende explizite Repräsentation die Systemanalyse und -optimierung erschweren. [CLG+09, HGB10] fordern daher, Regelkreise explizit darzustellen und gleichwohl als First-Class-Entitäten aufzufassen. Explizite Regelkreise finden sowohl innerhalb von Prozessen zur Softwareentwicklung [Boe88] sowie beim IT-Servicemanagement [BCD+07] Verwendung. Innerhalb des IT-Servicemanagements ist der Managementprozess klar vom Entwicklungsprozess getrennt. Zudem werden beide Prozesse unabhängig voneinander geplant.

2.3.5.2 MAPE-K-Regelkreis

IBMs Initiative zum Autonomic Computing gilt als wichtiger Meilenstein bei dem Versuch, ein adaptives Systemverhalten mit Hilfe expliziter Regelkreise zu realisieren [IBM06]. Dazu wurde mit dem *MAPE-K-Regelkreis* (Monitor, Analyze, Plan, Execute und Knowledge) ein Referenzmodell für autonome Elemente aufgestellt. Es hat gemäß [Kir06] folgende Bestandteile:

Überwachung IT-Systeme liefern eine Vielzahl managementrelevanter Zustandsinformationen. Die Daten sind häufig unzusammenhängend und weisen kein einheitliches Format auf. Sie müssen gefiltert, aggregiert und für eine spätere Nutzung aufbereitet werden.

Analyse Die Daten gilt es, miteinander in Beziehung zu setzen. Zu diesem Zweck lassen sich Techniken zur Datenkorrelation einsetzen. Mittels Mustererkennung wird innerhalb der Datenbasis nach Strukturen gesucht, die einem Symptom zugeordnet sind. Reasoning Engines helfen dabei, diejenigen Operationen zu bestimmen, deren Ausführung zur Aufrechterhaltung der Zielvorgaben erforderlich ist.

Planung Die Aktionen werden in eine Ausführungsreihenfolge gebracht und mit den eingeplanten bzw. den derzeit stattfindenden Aktivitäten abgestimmt. Der Plan kann aus einer Einzelaktion oder einer Aktionssequenz bestehen.

Ausführung Die Änderungen werden in Form von Systemeingriffen durchgeführt. Ziel ist es, den Systemzustand dahingehend zu beeinflussen, dass alle Zielvorgaben erfüllt sind. Zu diesem Zweck kommen Befehle, Skripte oder Programme zum Einsatz, die über Effektoren die notwendigen Anpassungen veranlassen.

Wissen Die Phasen nutzen zu ihrer Durchführung eine Wissensdatenbank. Es handelt sich um eine Komponente, die Wissen in Form von Adaptionsmaßnahmen enthält sowie Informationen über den Systemzustand und die derzeitigen Managementaktivitäten.

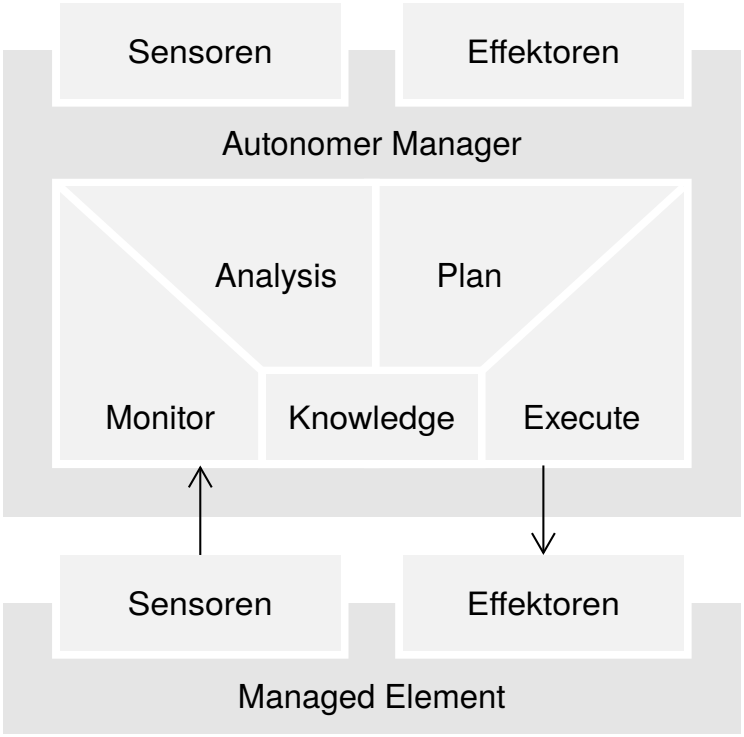


Abbildung 2.6: Autonomes Element von IBM nach [IBM06]

Gemäß der IBM-Referenzarchitektur bestehen autonome Systeme aus miteinander verschachtelten *autonomen Elementen*. Sie verfolgen ein gemeinsames Ziel und bilden die Bausteine zur Umsetzung der Self-X-Eigenschaften. Den Aufbau eines autonomen Elements zeigt die Abbildung 2.6. Es besteht aus einer zu managenden Ressource und einem autonomen Manager, der den MAPE-K-Regelkreis implementiert. Dabei entspricht der autonome Manager dem Regler, die zu managende Ressource der Regelstrecke und die Managementschnittstelle dem Stellglied. Über sie kann der autonome Manager Zustandsinformationen abrufen, auf deren Grundlage er entscheidet, ob der aktuelle Systemzustand den Einsatz adaptiver Maßnahmen erforderlich macht. Ein autonomer Manager stellt selbst wiederum eine zu managende Ressource dar, wobei über seine Managementschnittstelle Zustandsinformationen abrufbar sind und sich Anpassungen durchführen lassen. Besitzt ein Manager keine Effektoren, so ist eine Einflussnahme auf das Steuerungsverhalten zur Laufzeit nicht möglich. Auch wenn das autonome Element als richtungsweisende Neuerung zur Konstruktion selbstadaptiver Softwarekomponenten eingeführt wurde, handelt es sich im Grunde genommen um eine Übertragung der aus der klassischen Regelungstechnik bekannten Steuerungsansätze auf Softwaresysteme.

2.3.6 Mechanismen

Mechanismen zur Unterstützung heterogener und dynamischer Infrastrukturen werden seit der Verwendung von Rechnernetzen und verteilten Systemen intensiv untersucht und kontinuierlich weiterentwickelt [ST09]. Prinzipien wie das Autonomic- und Grid-Computing haben in diesem Zusammenhang wertvolle Erkenntnisse geliefert und dazu beigetragen, Adaptionsmechanismen für adaptive Systeme zu identifizieren [NA12]. Unter einem *Adaptionsmechanismus* wird ein Mechanismus verstanden, der bezüglich eines bestimmten Zustands der Ausführungsum-

gebung eine an diesen Zustand angepasste Verarbeitung realisiert bzw. deren Realisierung ermöglicht [Spr04].

2.3.6.1 Adaption der Softwarearchitektur

In der Literatur existieren unterschiedliche (formale) Ansätze zur Beschreibung von Softwarearchitekturen. Es existiert keine einheitliche Begrifflichkeit bezüglich der einzusetzenden Beschreibungselemente. [PW92] verwendet *Ausführungs-*, *Verbindungs-* und *Datenelemente*, [GMW00] hingegen *Komponenten*, *Konnektoren*, *Systeme*, *Ports*, *Rollen*, *Repräsentationen* und *Repräsentationstabellen*. Die von [Fie00] geprägten Begriffe *Komponenten*, *Konnektoren* und *Daten* sind heute gängig. Obwohl die Bezeichnungen Software- und Systemarchitektur meist synonym verwendet werden, gilt es diese im Zusammenhang mit adaptiven System zu unterscheiden. Die *Softwarearchitektur* bezieht sich auf die Organisation und die Struktur von Softwarebestandteilen in Form von Komponenten und Konnektoren. Die *Systemarchitektur* hingegen beschreibt die Eigenschaften und die Organisationsstruktur der Ressourcen auf der physikalischen Ebene und beinhaltet die Zuordnung der Komponenten auf die Verarbeitungsknoten. Sie beschreibt demzufolge ein System in Form einer konkreten Realisierung, das eine Gesamtfunktion erbringt. Es lassen sich grundlegende Basismechanismen zur Adaption unterscheiden, wobei sie jeweils bezogen auf die Software- bzw. Systemarchitektur verschiedenartige Bedeutungen haben.

2.3.6.2 Struktur- und Parameteradaption

Eine Strukturadaption bewirkt eine Veränderung der Komponentenstruktur, die aus logischer und physikalischer Sicht betrachtet werden kann. Veränderungen der Konnektoren ermöglichen Anpassungen der statischen und dynamischen Abhängigkeiten zwischen Komponenten. Die Parameteradaption ermöglicht die Rekonfiguration einer oder mehrerer Komponenten. In der Regel lassen sich Strukturadaptionen unabhängig vom Entwicklungsprozess planen und realisieren, wohingegen Parameteradaptionen eine explizite Unterstützung vom Entwickler erfordern und lediglich die Aspekte berücksichtigen können, die sich bei der Implementierung vorhersehen lassen. Die folgenden Operationen können gemäß [Kla04, Spr04] im Zusammenhang mit einer Struktur- und Parameteradaption unterschieden werden:

Hinzufügen einer Komponente Das Hinzufügen einer Komponente entspricht aus Sicht der Softwarearchitektur einer funktionalen Erweiterung und dient der erweiternden Adaption. An diesem Vorgang ist vorwiegend der Mensch beteiligt, der die neue Komponente implementiert und dem System bekannt macht. Hinsichtlich der Systemarchitektur führt das Einbringen einer Komponente zur Erzeugung einer (weiteren) Instanz, die einem Verarbeitungsknoten zugeordnet werden muss. Dadurch lassen sich u. a. die Fehlertoleranz und die Verfügbarkeit verbessern.

Replizieren einer Komponente Die Replikation einer Komponente erfolgt ausschließlich innerhalb der Systemarchitektur. Das Replikat ist eine vollständige Kopie seines Originals und weist zunächst denselben Zustand auf. Je nach zugrundeliegender Strategie können sich diese Zustände im Laufe der Zeit auseinander entwickeln.

Migrieren einer Komponente Die Migration einer Komponente ermöglicht die dynamische Änderung ihres Ausführungsortes. Dazu muss zunächst der aktuelle Zustand der Komponente erfasst werden. Anschließend wird die Komponente gestoppt, an ihrem neuen Ausführungsort gestartet und in den alten Zustand versetzt. Diese Änderung ist für die

Softwarearchitektur transparent, da die logischen Kommunikations- und Datenflüsse erhalten bleiben.

Entfernen einer Komponente Ähnlich wie beim Hinzufügen einer Komponente entspricht ihr Herausnehmen aus Sicht der Softwarearchitektur dem Entfernen von Funktionalität. Aus Sicht der Systemarchitektur wird die Komponente gestoppt und entfernt. In diesem Zusammenhang ist ein Mechanismus erforderlich, der ein kontrolliertes Herunterfahren möglich macht, ohne eine Fehlersituation im Anwendungsszenario hervorzurufen.

Austauschen einer Komponente Eine Komponente wird durch eine andere Implementierung ersetzt. Diese kann veränderte funktionale und nicht-funktionale Eigenschaften aufweisen, ist aber funktions- und schnittstellenkompatibel. Ähnlich wie bei der Migration einer Komponente, muss auch bei ihrem Austausch ein Zustandstransfer erfolgen. Des Weiteren muss sichergestellt sein, dass nicht beide Versionen gleichzeitig aktiv sind.

Anpassen eines Konnektors Konnektoren beschreiben den logischen bzw. physikalischen Datenfluss innerhalb des Systems. Durch das Anpassen der logischen Topologie kann eine vollständig andere Systemfunktionalität erreicht werden. Innerhalb eines verteilten Systems kommt der Anpassung der physikalischen Topologie besondere Bedeutung zu. Durch das Neubinden von Komponenten lassen sich unterbrochene oder beeinträchtigte Nachrichtenflüsse wiederherstellen bzw. beschleunigen.

Anpassen von Parametern Eine Komponente lässt sich mittels Parameteradaption rekonfigurieren. Dazu muss der Komponentenentwickler den entsprechenden Konfigurationsraum festlegen und spezifizieren. Rekonfigurationen erlauben es, auf das Verhalten einer Komponente steuernd einzuwirken. Je nach verfügbaren Parametern ist eine vollständige Verhaltensanpassung möglich.

2.3.6.3 Schnittstellenadaption

Eine Schnittstellenadaption kann vereinfacht als Folge der beiden Strukturadaptionen „Entfernen einer Komponente“ und „Hinzufügen einer Komponente“ aufgefasst werden. Dennoch ist eine differenzierte Betrachtung erforderlich, da sich mit Hilfe eines modularen Systementwurfs im begrenzten Maße eine Schnittstellenadaption realisieren lässt, ohne die Komponente während ihres Betriebs zu stoppen und wieder zu starten. Gemäß [Kla04, Spr04] lassen sich die folgenden Operationen unterscheiden:

Hinzufügen einer Schnittstelle Einer Komponente wird eine neue bzw. eine bereits im System vorhandene Schnittstelle hinzugefügt. Dazu muss die Implementierung der Funktion von der ihres Zugriffs getrennt erfolgt sein. Das Prinzip der Datenkapselung unterstützt dieses Anpassungsprinzip. Neue Zugriffsmöglichkeiten oder (veränderte) Protokollversionen lassen sich somit ohne Modifikation des Programmcodes in das System einbringen.

Entfernen einer Schnittstelle Das Entfernen einer Schnittstelle ermöglicht das Herausnehmen einer Zugriffsmöglichkeit. Gründe hierfür sind Fehler in der Implementierung der Schnittstelle, oder die von ihr bereitgestellte Zugriffsmöglichkeit ist für die Systemfunktion nicht (mehr) erforderlich.

Anpassen einer Schnittstelle Die Adaption der Schnittstellen ermöglicht eine Zusammenarbeit von Komponenten mit zunächst inkompatiblen Schnittstellen. Dazu kann eine

Operation hinzugefügt, entfernt oder ihre Signatur verändert werden. Die Signatur einer Operation umfasst neben der Bezeichnung und der Anzahl und Reihenfolge der Parameter auch den Datentyp der Rückgabe. Die automatisierte Anpassung der Schnittstelle ist schwierig und hat meist Auswirkung auf die dahinterliegende Semantik. Vielfach existieren explizite und implizite Anforderungen an die Repräsentation der Parameter, so dass automatisierte Anpassungen nur sehr eingeschränkt möglich sind.

Struktur-, Parameter- und Schnittstellenadaptionen können sowohl in der Anwendung als auch in der Adaptionsinfrastruktur erfolgen. Dazu muss die Adaptionsinfrastruktur den Adaptionsprozess in Form einer externen Adaption realisieren. Die Anwendung und die Adaptionsinfrastruktur liegen getrennt voneinander vor und lassen sich jeweils durch separate Software- und Systemarchitekturen beschreiben.

2.4 Ausführungsumgebungen

Um IT-Systeme betreiben zu können, müssen technische Ausführungsumgebungen zum Einsatz kommen. Wie die Abbildung 2.7 veranschaulicht, lassen sich physikalische, virtuelle und cloudbasierte Umgebungen von Umgebungsverbünden unterscheiden. Diese werden gemäß einem Betriebsmodell erbracht. Die nachfolgenden Abschnitte erläutern die Ausführungsumgebungen sowie die unterschiedlichen Betriebsmodelle.

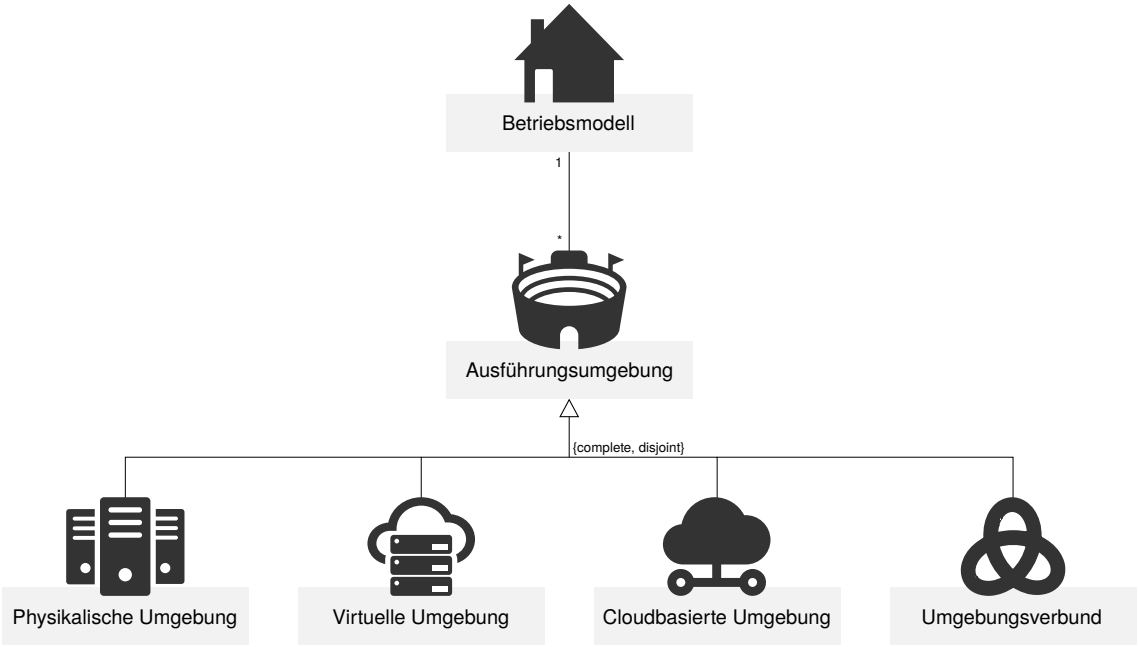


Abbildung 2.7: Klassifikation von Ausführungsumgebungen

2.4.1 Physikalische Umgebung

Eine physikalische Umgebung beinhaltet die Hardware und Software der IT-Infrastruktur, die für den Betrieb von (Anwendungs-)Software erforderlich sind. Dazu zählen Host-Systeme sowie Netzwerk- und Speicherkomponenten. Diese werden typischerweise in einem *Rechenzentrum* (RZ) oder *Datacenter* (DC) auf Basis einer physischen Infrastruktur betrieben. Bei einem

verteilten Rechenzentrum, auch *Distributed Datacenter* genannt, befinden sich Teile eines funktional einheitlichen Rechenzentrums an geografisch verteilten Standorten. Die RZ-Infrastruktur besteht aus Räumen, Stromversorgung, Klimatisierung, Sicherheitssystemen sowie einer Anbindung an Datennetze. Beim Betrieb eines Rechenzentrums müssen die Verfügbarkeit sowie die Daten- und Funktionssicherheit der Infrastrukturservices gewährleistet sein. Diese erfordern eine unterbrechungsfreie Stromversorgung, ausreichend Kühlung und redundant ausgelegte Systemkomponenten. Als Qualitätsnachweis gilt die Zertifizierung auf Basis des IT-Grundschutzes nach ISO 9001 [ISO15] und ISO/IEC 27001 [BSI14].

2.4.2 Virtuelle Umgebung

Mit dem Aufkommen leistungsfähiger Hard- und Softwaresysteme in den 60er Jahren beeinflusste auch der kosteneffiziente Einsatz den technologischen Fortschritt. Beim ATLAS-Computer, der als rechenstärkster Computer seiner Zeit am 7. Dezember 1962 in Betrieb ging, virtualisierte man aus Kostengründen erstmals den Hauptspeicher [Bel66]. Mitte der 60er Jahre entstand bereits der Vorläufer einer virtuellen Maschine. Die Virtualisierungstechnologie wurde über Jahrzehnte hinweg kontinuierlich weiterentwickelt und fortwährend verbessert [BG73, Gol73]. Sie kam aber nur vereinzelt zur Anwendung und fristete lange Zeit ein Schattendasein. Im Jahre 1999 erweckte sie erstmals öffentliches Interesse. Der Firma *VMware* gelang es mit *VMware Workstation* [BDR+12] eine x86-Architektur zu virtualisieren [Ros99]. Bis zu diesem Zeitpunkt ging man davon aus, dass aufgrund von Hardwarebeschränkungen nicht alle Rechnerarchitekturen virtualisierbar seien. Zwei Jahre später folgten die beiden Server-Versionen *GSX* und *ESX*, die den Einsatz von Virtualisierung auch auf Server-Hardware und damit im Umfeld von Rechenzentren ermöglichten.

2.4.2.1 Virtualisierung

Unter *Virtualisierung* versteht man die Emulation oder Simulation von Hardwareressourcen, bei der die Hardware vom Betriebssystem entkoppelt wird [BKL09]. Dadurch erhalten Ressourcen eine zentrale und einheitliche Repräsentation unabhängig von ihrem Standort und ihren spezifischen Eigenschaften. Virtualisierung ist demnach für den Anwender transparent. Die physischen Ressourcen werden dabei auch als *virtualisiert*, die logischen Ressourcen als *virtuell* bezeichnet. Häufig wird eine physische Ressource auf mehrere virtuelle Ressourcen abgebildet, beispielsweise um im Rahmen von Prozessorvirtualisierung die Auslastung der physischen Prozessoren zu erhöhen. Alternativ kommt Virtualisierung zum Einsatz, wenn eine Vielzahl heterogener physischer Ressourcen zu einer Menge homogener logischer Ressourcen zusammengefasst werden soll. In diesem Fall kann die Zahl der logischen Ressourcen deutlich kleiner sein als die der physischen. Je nachdem auf welcher Abstraktionsebene zwischen Hardware und Software die Virtualisierungsschicht platziert wird, können unterschiedliche Arten der Virtualisierung unterschieden werden:

Plattformvirtualisierung Ein *Virtual-Machine-Monitor* (VMM) oder *Hypervisor* erlaubt die Ausführung beliebiger Betriebssysteme oder Programme. Ein Hypervisor ist im Wesentlichen ein minimales Betriebssystem, welches die Hardwareressourcen für die Gastbetriebssysteme verwaltet und den Zugriff kontrolliert. Er kann entweder direkt auf der Hardware (Typ 1) oder auf einem regulären Betriebssystem (Typ 2) aufsetzen.

Speichervirtualisierung Der Speicher (Storage) wird in Form virtueller Laufwerke bereitgestellt. Dazu muss der physische Speicher zusammengefasst und logisch aufgeteilt

werden. Die Nutzer sind somit nicht mehr an die physischen Grenzen der Speichermedien gebunden.

Netzwerkvirtualisierung Ein *virtuelles lokales Netzwerk* (VLAN) ermöglicht die Zusammenfassung verteilter Ressourcen in einem gemeinsamen logischen Netzwerk. Dazu wird über das physikalische Netz eine zusätzliche virtuelle Schicht gelegt, so dass sich Komponenten transparent in ein bestehendes Netzwerk integrieren lassen.

Betriebssystemvirtualisierung Ein Betriebssystem bietet mehrere virtuelle Instanzen von sich selbst. Dazu erzeugt es Laufzeitumgebungen, sogenannte *Jails* oder *Container*, die für die laufenden Programme wie normale Betriebssysteme erscheinen. Die Betriebssystemvirtualisierung wird auch als *containerbasierte Virtualisierung* bezeichnet.

Anwendungsvirtualisierung Es werden mehrere virtuelle Instanzen einer Anwendung bereitgestellt, wobei jeder Nutzer der Anwendung scheinbar exklusiven Zugriff darauf hat.

Hardwarevirtualisierung Physikalische Hardware wird zu virtuellen Hardwarekomponenten abstrahiert und kann in gleicher Weise genutzt werden. Dies erlaubt einen Austausch der Hardware, solange die veränderte Umgebung in der Lage ist, die gleiche virtuelle Hardware bereitzustellen.

2.4.2.2 Merkmale

Virtuelle Umgebungen nutzen eine Virtualisierungstechnologie und stellen dem Anwender virtuelle Berechnungs-, Netzwerk- und Speicherressourcen zur Verfügung. Mit virtuellen Maschinen und Containern existieren derzeit zwei unterschiedliche Lösungen zur Bereitstellung virtueller Host-Systeme, auf denen sich Anwendungen betreiben lassen. Virtuelle Umgebungen setzen zumeist auf physikalischen Host-Systemen auf, grundsätzlich ist aber auch eine *verschachtelte Virtualisierung* möglich. Dabei wird die virtuelle Umgebung auf Basis virtueller Host-Systeme erbracht. Die Grenze zwischen virtuellen und cloudbasierten Umgebungen ist fließend. Als virtuelle Umgebung wird eine Ausführungsumgebung verstanden, die eine Virtualisierungstechnologie verwendet und mindestens eines der in Abschnitt 2.4.3.2 beschriebenen Cloud-Merkmale verletzt.

2.4.3 Cloudbasierte Umgebung

Mit der zunehmenden Verbreitung schneller Internetzugänge entwickelte sich Anfang der 2000er Jahre ein neues IT-Paradigma: *Cloud-Computing*. Es gilt mittlerweile als das Konzept der Zukunft für Bereitstellung, Betrieb und Nutzung von IT-Systemen. Cloudbasierte Umgebungen versprechen neue Möglichkeiten, Datenverarbeitungsprozesse zu organisieren und zu verwalten. Es stehen nicht mehr länger die softwaretechnischen Werkzeuge oder die physikalischen oder virtuellen Host-Systeme im Vordergrund, sondern die zur Problemlösung erforderlichen Services. Diese lassen sich nach Bedarf über das Internet beziehen und verbrauchsorientiert abrechnen. Dazu nutzen Anbieter oftmals Hardware und Software von Unterauftragnehmern, die weltweit verteilt sind. Blickt man zurück auf die Rechnerbetriebsmodelle der 60er und 70er Jahre, lässt sich feststellen, dass das Cloud-Computing der Wiederkehr eines bekannten Trends entspricht und dem Grundprinzip einer zentralisierten, gemeinsam nutzbaren Rechenleistung folgt [BHRL10].

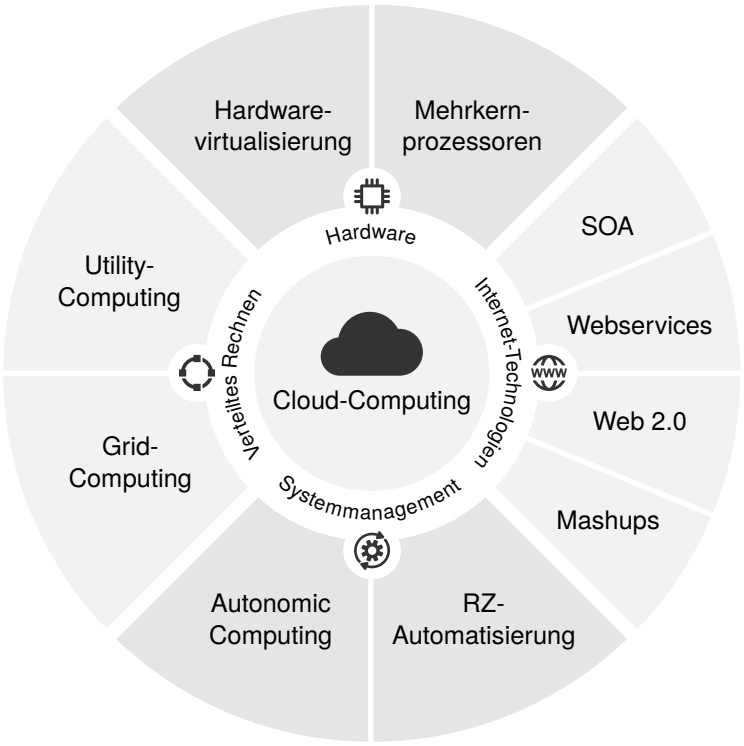


Abbildung 2.8: Cloud-Computing als Schnittmenge von Technologien nach [CG17]

2.4.3.1 Einordnung

Das Cloud-Computing beschreibt keine klar abgegrenzte Disziplin. Vielmehr werden unterschiedliche Technologien und Prinzipien miteinander verbunden und zu einem neuen Gesamtkonzept zusammengefügt. Diesen Zusammenhang zeigt die Abbildung 2.8. Das Cloud-Computing stellt zwar einen echten Paradigmenwechsel in der Nutzung und Bereitstellung von IT-Services dar, entspricht aus technischer Sicht aber eher einer Weiterentwicklung bestehender Lösungen. Lange Zeit hat sich für den Begriff des Cloud-Computings keine allgemein akzeptierte Definition finden lassen. In Publikationen und Forschungsarbeiten sind daher unterschiedliche Definitionen zu finden, die zwar grundsätzliche Ähnlichkeiten aufweisen, sich aber im Detail stark unterscheiden [VRMCL08]. Die meistgenutzten Definitionen stammen vom *Bundesamt für Sicherheit in der Informationstechnik* (BSI) und dem US-amerikanischen *National Institute of Standards and Technology* (NIST). Die Definition des NIST wird auch von der *European Network and Information Security Agency* (ENISA) verwendet. Demnach gilt:

Bundesamt für Sicherheit in der Informationstechnik Das BSI definiert das Cloud-Computing als das dynamisch an den Bedarf angepasste Anbieten, Nutzen und Abrechnen von IT-Dienstleistungen über ein Netzwerk. Angebot und Nutzung dieser Dienstleistungen erfolgen dabei ausschließlich über definierte technische Schnittstellen und Protokolle. Die Spannbreite der im Rahmen des Cloud-Computings angebotenen Dienstleistungen umfasst das komplette Spektrum der Informationstechnik und beinhaltet unter anderem Infrastruktur (z. B. Rechenleistung, Netzwerke, Speicherplatz), Plattformen und Anwendungen [EDM12].

National Institute of Standards and Technology Die Standardisierungsstelle NIST definiert das Cloud-Computing als ein Modell für einen ubiquitären, komfortablen und bedarfsbasierten Netzwerkzugriff auf einen gemeinsam genutzten Pool konfigurierbarer Ressourcen (z. B. Netzwerke, Server, Speicherkapazität, Anwendungen und Services), die schnell verfügbar gemacht und mit minimalem Verwaltungsaufwand oder minimaler Interaktion durch den Serviceanbieter bereitgestellt und freigegeben werden können [MG11].

2.4.3.2 Merkmale

Die folgenden fünf Merkmale sind gemäß NIST charakteristisch für cloudbasierte Umgebungen:

On-Demand Self-Service Nutzer von Services können selbstständig ohne weiteres Zutun des Anbieters zusätzliche Ressourcen (z. B. Rechenzeit, Bandbreite, Speicherkapazität) hinzu- oder abbuchen. Die Services müssen dabei Self-X-Eigenschaften besitzen, da ansonsten der automatisierte Bereitstellungs- und Nutzungsprozess auf Anbieterseite nicht realisierbar ist.

Broad Network-Access Alle Services sind über das Netzwerk mittels Standardprotokollen und -mechanismen wie HTTP, XML, JSON zugreifbar und können gleichermaßen von Thin- und Thick-Clients (z. B. Rechner, Smartphone, Tablet) genutzt werden. Zugang und Nutzung von Services müssen unabhängig von der Leistungsfähigkeit des Clients möglich sein. Dazu stellen Anbieter häufig Zugriffsschnittstellen als RESTful Webservices zur Verfügung. Um die volle Leistungsfähigkeit des Services ausschöpfen zu können, muss die Netzwerkverbindung zwischen Anwender und Anbieter eine hinreichend große Bandbreite besitzen.

Resource-Pooling Die Ressourcen des Cloud-Anbieters werden gebündelt und dynamisch an die Bedürfnisse des Anwenders angepasst. Dabei wissen diese in der Regel nicht, wo sich die Ressourcen tatsächlich befinden und wie sie beschafft wurden. Auf einer höheren Ebene können sie aber vertraglich sehr wohl den Ausführungs- und Speicherort (z. B. Region, Land, Rechenzentrum) beeinflussen. Hauptaufgabe des Resource-Pooling ist die Verteilung der Anfragen auf die zur Verfügung stehenden logischen und physischen Funktionseinheiten. Dadurch ist es dem Anbieter möglich, seine Ressourcen optimal einzusetzen.

Rapid Elasticity Services lassen sich, um den Bedürfnissen des Nutzers gerecht zu werden, zeitnah bereitstellen und wieder beenden. Dazu ist es notwendig, schnell nach oben und nach unten skalieren zu können. Eine Anpassung kann einen manuellen Eingriff erfordern oder vollständig automatisiert ablaufen. Aus Anwendersicht erscheinen die nutzbaren Ressourcen daher als annähernd unbegrenzt.

Measured Services Die Ressourcenverwendung muss fortwährend überwacht und kontrolliert werden, um sowohl Nutzern als auch Anbietern einen Überblick über die tatsächlich erbrachte Leistung zu geben. Die Nutzung erfolgt im Rahmen eines vereinbarten Abrechnungsmodells. Welche Messverfahren und -größen zum Einsatz kommen, hängt maßgeblich von der Art der Ressource sowie den Vertragsbedingungen ab. Dazu gilt es, die Messgrößen in Abhängigkeit zu den kundenspezifischen Anforderungen geeignet zu abstrahieren und zusammenzufassen.

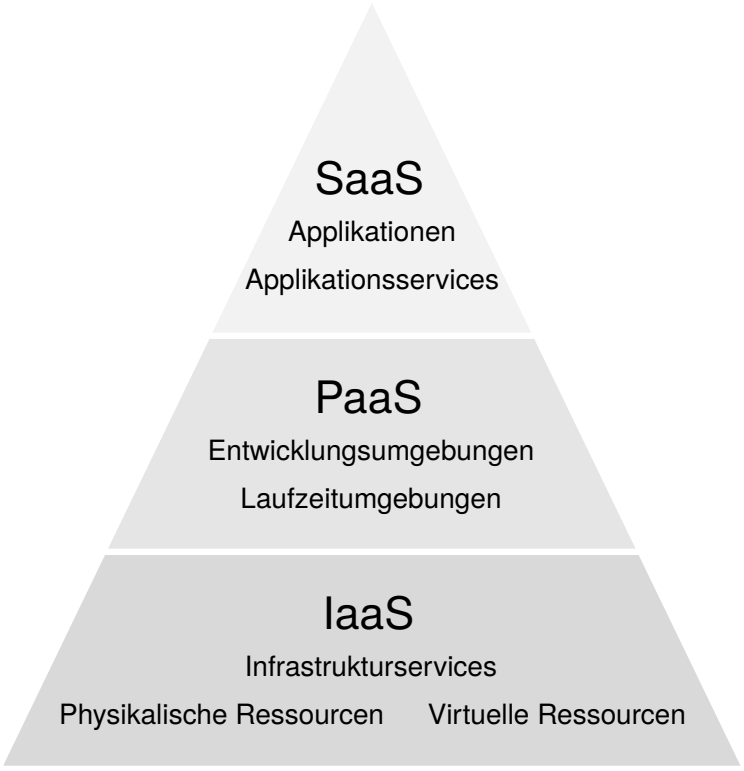


Abbildung 2.9: Schichtenmodell des Cloud-Computings nach [LKN+09]

2.4.3.3 Servicemodell

Das Grundprinzip des Cloud-Computings ist die Auffassung jeder Art von Dienstleistung als Service, die sich im Sinne von *Everything-as-a-Service* (XaaS) [DFZ+15] jederzeit bedarfs- und anforderungsgerecht bereitstellen lässt. XaaS definiert in diesem Zusammenhang die folgenden drei Kriterien:

- Die Dienstleistung oder Ressource wird durch einen Anbieter bevorratet und dem Anwender auf Anfrage zur Verfügung gestellt,
- sie wird mit kurzer Mindestnutzungsperiode angeboten und
- lässt sich durch den Anwender nach dem Self-Service-Prinzip jederzeit aktivieren, deaktivieren und anpassen.

Cloud-Services bilden eine Hierarchie, die als *Service-Stack* [LKN+09] bezeichnet wird und die in der Regel für die Anwender nicht sichtbar ist. Es existieren unterschiedliche Arten von Cloud-Services. Zu ihrer Klassifikation bietet der Service-Stack ein hierarchisches, dreischichtiges Modell. Eine Ausführungsumgebung muss zwar die XaaS-Kriterien vollständig erfüllen, jede ihrer Schichtimplementierungen aber hingegen nur teilweise. Services lassen sich üblicherweise einer dieser drei Schichten zuordnen und richten sich vornehmlich an bestimmte Nutzer bzw. Nutzergruppen. Dabei können Services höherer Schichten die von den unteren Schichten bereitgestellten Funktionen zu ihrer eigenen Realisierung nutzen. Sie bauen aufeinander auf und bilden immer spezifischere Services. Die einzelnen Schichten gruppieren die Services gemäß dem Einsatz- und Verwendungszweck. Lässt sich ein Service nicht eindeutig einer

Schicht zuordnen, wird dieser der höchsten Schicht zugeteilt. Die drei Schichten sind in der Abbildung 2.9 dargestellt. Es handelt sich um:

Infrastructure-as-a-Service Die unterste Schicht wird als *Infrastructure-as-a-Service* (IaaS) bezeichnet. Sie stellt Infrastrukturressourcen zur Verfügung und bietet Anwendern ein Höchstmaß an Flexibilität und Anpassbarkeit hinsichtlich Nutzung und Administration. Die IaaS-Schicht abstrahiert von den Hardwareressourcen und erlaubt es, Infrastrukturkomponenten über das Internet zu beziehen. Hierzu zählen Server-, Massenspeicher- und Netzwerkkomponenten. Typische Aufgaben sind das Starten und Stoppen von Host-Systemen, das Anlegen und Löschen von Betriebssystem-Abbildern sowie die Definition von Netzwerktopologien. Die IaaS-Schicht unterscheidet zudem zwischen *Ressourcen* und *Infrastrukturservices*. Ressourcen entsprechen entweder physikalischen oder virtuellen Ressourcen. *Physikalische Ressourcen* repräsentieren physikalische Infrastrukturkomponenten, wohingegen *virtuelle Ressourcen* mit Hilfe einer Virtualisierungstechnologie bereitgestellte Infrastrukturkomponenten darstellen. Infrastrukturservices besitzen einen engeren Anwenderfokus und dienen der Datenverarbeitung, -kommunikation und -speicherung.

Platform-as-a-Service Die mittlere Schicht wird als *Platform-as-a-Service* (PaaS) bezeichnet. Sie baut auf der IaaS-Schicht auf und bietet ihren Anwendern eine vollständig verwaltete Ausführungsplattform. PaaS-Angebote richten sich in der Regel nicht an den Endanwender, sondern vorrangig an Entwickler. Mit Hilfe von PaaS lassen sich eigene Anwendungen entwickeln. Die dafür notwendigen Werkzeuge werden durch den Cloud-Provider verwaltet. PaaS lässt sich weiter unterteilen in *Entwicklungsumgebungen* und *Laufzeitumgebungen*. Ressourcen der IaaS-Schicht werden abstrahiert und als Services zur Verfügung gestellt. Dadurch ist der Entwickler von der darunterliegenden Infrastruktur entkoppelt und kann ausschließlich auf die von der Plattform bereitgestellten Services zurückgreifen.

Software-as-a-Service Die höchste Schicht wird als *Software-as-a-Service* (SaaS) bezeichnet. Sie umfasst Anwendungsfunktionen, die sich direkt von Endanwendern nutzen lassen. Grundgedanke von SaaS ist es, Software nicht mehr länger auf einem lokalen Rechner bereitzustellen, sondern diese über das Internet einer Vielzahl von Anwendern gleichzeitig zugänglich zu machen. Seitens des Anwenders ist zur Nutzung keine eigene Ausführungsumgebung mehr erforderlich, Voraussetzung ist nur ein an das Internet angebundener Rechner. Verwaltung und Administration unterliegen dabei gänzlich dem Verantwortungsbereich des externen IT-Dienstleisters. Der Anwender selbst hat keinen direkten Einfluss auf den technischen Betrieb und verfügt zumeist über keinerlei Kenntnisse hinsichtlich des Ausführungsorts und der eingesetzten Technologien. Die SaaS-Schicht lässt sich weiter in *Applikationen* und *Applikationsservices* unterteilen. Bei Applikationsservices handelt es sich um unterstützende Services, die sich von Applikationen zur Funktionserbringung einsetzen lassen.

2.4.4 Umgebungsverbund

Beim Eintritt von Lastspitzen kann es vorkommen, dass ein einzelner Anbieter nicht über ausreichende Kapazitäten verfügt, um alle Serviceanfragen bedienen zu können. Insbesondere in Verbindung mit Servicegüteanforderungen kann die Überbuchung von Ressourcen negative Folgen haben und Forderungen von Seiten des Kunden nach sich ziehen. In solchen Überlastsituationen ist es sinnvoll, wenn auf Services anderer Anbieter zurückgegriffen werden kann. Voraussetzung dafür ist die Servicebereitstellung in einem *Umgebungsverbund*. Dabei handelt

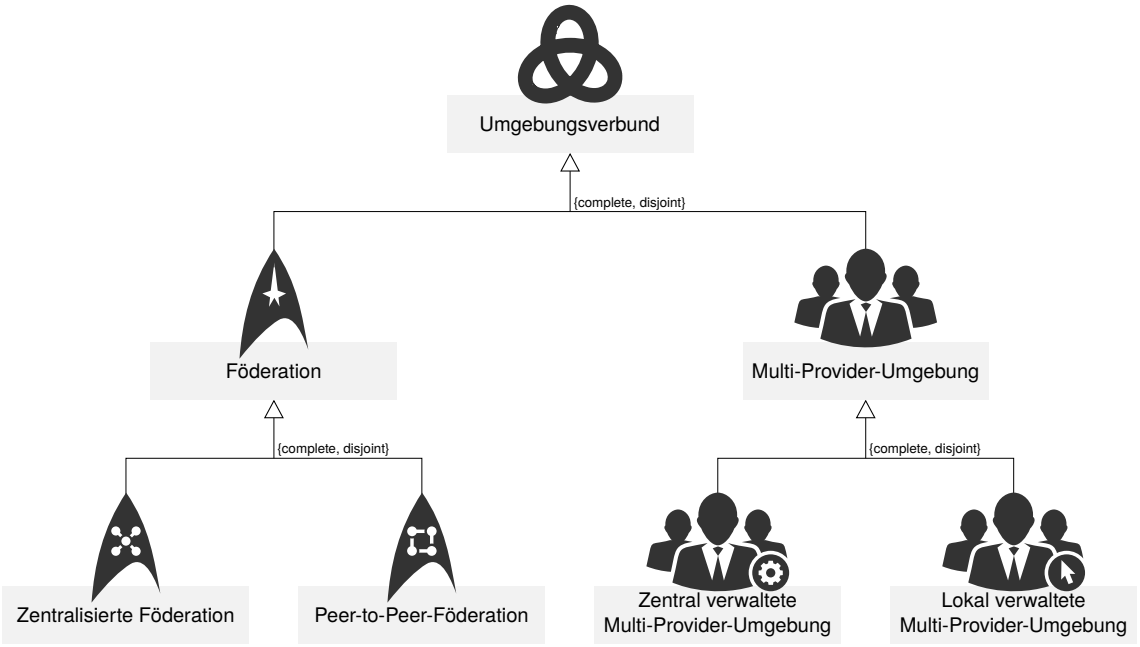


Abbildung 2.10: Klassifikation von Umgebungsverbünden

es sich um eine lose Koppelung von Ausführungsumgebungen unterschiedlicher Anbieter zu einem Netzwerk. Im Zusammenhang mit cloudbasierten Umgebungen wird auch der Begriff *Inter-Cloud* [KTMF09, RRB15] verwendet. Gemäß der Abbildung 2.10 entspricht ein Umgebungsverbund entweder einer Föderation oder einer Multi-Provider-Umgebung. Diese sind wie folgt definiert:

Föderation Eine *Föderation* ist ein Zusammenschluss von Ausführungsumgebungen mehrerer Anbieter und erfolgt auf freiwilliger Basis [VBR+12]. Üblicherweise besteht ein zuvor ausgehandelter Vertragsrahmen, der Rechte und Pflichten der Teilnehmer festlegt. Ziel ist die Ausweitung des eigenen Serviceangebots oder der Einsatz fremdbezogener Services zum Ausgleich von Lastspitzen [CTVP10]. Eine Föderation ist ein anbieterseitiger Zweckverbund, dessen Aufbau und Struktur dem Anwender verborgen bleibt. Für ihn erscheint die Föderation als eine einzelne Ausführungsumgebung, die entweder zentral oder dezentral organisiert ist.

Zentralisierte Föderation Eine *zentralisierte Föderation* verfügt über einen zentralen Broker, der in Abhängigkeit zur Anfrage entscheidet, in welchen Ausführungsumgebungen jeweils welche Services verwendet und bereitgestellt werden. Der Broker ist mit allen Ausführungsumgebungen verbunden und verantwortet ihren koordinierten Einsatz.

Peer-to-Peer-Föderation Bei einer *Peer-to-Peer-Föderation* existiert kein zentraler Broker, vielmehr entscheidet jeder Anbieter selbst, in welchen Ausführungsumgebungen welche Services bereitgestellt werden. Dieser Ansatz bietet den Anbietern ein Höchstmaß an Flexibilität und Anpassbarkeit, erhöht aber auch gleichzeitig den Verwaltungsaufwand zur Koordination von Anfragen und Services.

Multi-Provider-Umgebung Bei einer *Multi-Provider-Umgebung* werden Ausführungsumgebungen anwenderseitig miteinander verbunden. Im Unterschied zur Föderation besteht keine

direkte Beziehung zwischen den Anbietern. Es obliegt ausschließlich der Verantwortung des Anwenders, die Ausführungsumgebungen und Services aufeinander abzustimmen. Der Vorteil liegt darin, dass er jeweils das für seine Anforderungen am besten geeignete Angebot auswählen kann.

Zentral verwaltete Multi-Provider-Umgebung Bei einer *zentral verwalteten Multi-Provider-Umgebung* kommt seitens des Anwenders ein Service zum Einsatz, über den der Zugriff auf den Umgebungsverbund erfolgt. Dabei handelt es sich um einen Broker, der die Verteilung der Services auf die Ausführungsumgebungen verantwortet.

Lokal verwaltete Multi-Provider-Umgebung Bei einer *lokal verwalteten Multi-Provider-Umgebung* wird eine Softwarebibliothek verwendet, die die Multi-Provider-Umgebung verwaltet. In diesem Fall ist der Lebenszyklus des Umgebungsverbunds an die Softwarebibliothek gekoppelt, die ihrerseits Teil eines Ausführungskontextes ist. Terminiert dieser, hört zugleich auch die Multi-Provider-Umgebung auf zu existieren.

2.4.5 Betriebsmodell

Das Betriebsmodell umfasst den geografischen Betriebsort und das Nutzungsmodell einer Ausführungsumgebung. Eine Klassifikation ist in der Abbildung 2.11 dargestellt. Im Bezug zum Betriebsort lassen sich folgende Fälle unterscheiden:

On-Premises Unter *On-Premises* wird der lokale Betrieb der Ausführungsumgebung vor Ort in den Räumlichkeiten des Anwenders verstanden. Dabei verantwortet der Anwender den

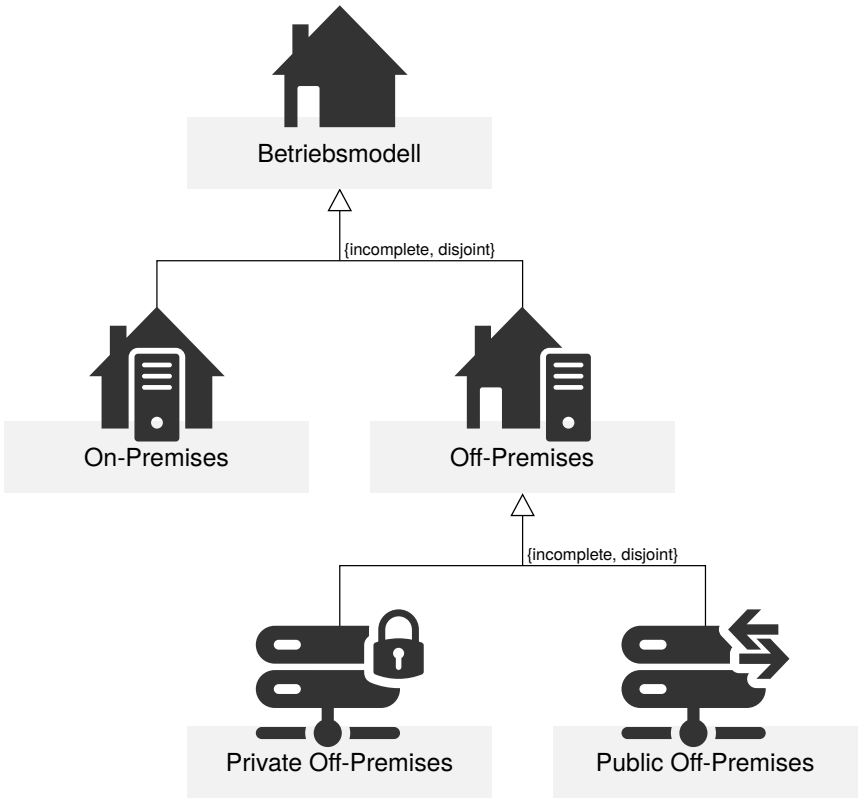


Abbildung 2.11: Klassifikation von Betriebsmodellen

korrekten Betrieb der physikalischen Hardwarekomponenten (Kühlung, Strom und Netzwerkanbindung), wohingegen der Anbieter die softwareseitige Betriebsverantwortung übernimmt.

Off-Premises Bei *Off-Premises* erfolgt der Betrieb der Ausführungsumgebung in Rechenzentren, die nicht der Verantwortung des Anwenders unterstehen. Dabei kann der Anwender, muss aber nicht, zugleich auch Eigentümer der physikalischen Hardwarekomponenten sein. Bei diesem Betriebsmodell übernimmt der Anbieter die hardware- und softwareseitige Betriebsverantwortung. In diesem Zusammenhang ist es nicht unüblich, die hardwareseitige Betriebsverantwortung an einen externen Dienstleister abzutreten.

Der Betrieb der Ausführungsumgebung in den Räumlichkeiten des Anwenders legt gleichzeitig auch das Nutzungsmodell fest. Die Ausführungsumgebung steht exklusiv dem Anwender zur Verfügung, eine öffentliche Nutzung durch betriebsfremde Dritte soll nicht möglich sein. Anders verhält es sich beim Off-Premises-Betrieb. Im Bezug zum Nutzungsmodell lassen sich folgende Fälle unterscheiden:

Private Off-Premises Unter *Private Off-Premises* wird eine private Ausführungsumgebung verstanden, die zwar in einem fremden Rechenzentrum betrieben wird, wobei der Anwender aber über ein exklusives Nutzungsrecht verfügt. Betriebsfremde Dritte haben keinen Zugriff auf die von der Ausführungsumgebung angebotenen Services.

Public Off-Premises Bei *Public Off-Premises* handelt es sich um eine öffentlich zugängliche Ausführungsumgebung, deren Services sich in der Regel gegen Zahlung eines Entgelts in Anspruch nehmen lassen. Die Anwender nutzen die Ausführungsumgebung gemeinsam und teilen sich das Leistungsangebot.

2.5 Technisches Management

Das technische Management befasst sich mit der Überwachung und Steuerung von IT-Systemen. Seine Aufgabe besteht darin, gleichzeitig sowohl die Anforderungen der Betreiber als auch die der Nutzer zu erfüllen [Slo95]. Dabei haben in den letzten Jahren die betreiber- und nutzerseitigen Ansprüche an die bereitgestellten Services stark zugenommen [RC07]. Diese Ansprüche umfassen neben den funktionalen vermehrt auch die nicht-funktionalen Eigenschaften [CGK+11]. Die Komplexität und Heterogenität verteilter IT-Umgebungen sowie der Einsatz immer leistungsfähigerer Kommunikationsnetzwerke haben dazu geführt, dass auch die Anforderungen an das technische Management stetig gestiegen sind [KKK96].

2.5.1 Disziplinen

Unter dem Begriff des *Managements* versteht [HAN99] die Gesamtheit aller Maßnahmen, die einen unternehmenszielorientierten effektiven und effizienten Betrieb eines verteilten Systems mitsamt seinen Ressourcen ermöglichen. Dabei lässt sich das Management je nach Art der zu verwaltenden Ressource in die folgenden Disziplinen einteilen:

Unternehmensmanagement Das Unternehmensmanagement fasst die Aufgaben des Finanz-, Personal-, Technologie- und Produktionsmanagements unter unternehmensweiten Gesichtspunkten zusammen. Daraus werden Zielvorgaben für die IT-Infrastruktur, die Betriebsprozesse, die Services, die Anwendungen und die Datenbestände abgeleitet. Das

Management wird mit dem gesamten Unternehmensumfeld in Beziehung gesetzt, so dass nicht nur technische, sondern auch organisatorische und betriebswirtschaftliche Aspekte Berücksichtigung finden.

Anwendungsmanagement Das Anwendungsmanagement ist zuständig für die Verwaltung verteilter Anwendungen und Services. Sie müssen überwacht, aktiviert bzw. deaktiviert sowie zur Laufzeit umverteilt werden. Sollen Services internen oder externen Kunden zur Verfügung stehen, sind diese anforderungsgerecht bereitzustellen und kundenbezogen zu verwalten. In diesen Fällen setzt auf dem Anwendungsmanagement noch ein Servicemanagement auf.

Informationsmanagement Das Informationsmanagement beschäftigt sich mit dem Entwurf und der Pflege unternehmensweiter Datenbestände zur Gewährleistung ihrer Konsistenz, Verfügbarkeit und Erreichbarkeit.

Systemmanagement Das Systemmanagement befasst sich mit der Verwaltung und Administration von Ressourcen innerhalb von Endsystemen und Systemverbünden. Ziel ist es, Benutzer und Betreiber während der Planung und des Betriebs zu unterstützen, um die angestrebten Güteanforderungen erreichen zu können. Verteilt erbrachte Systemfunktionen gilt es so zu organisieren, dass die beteiligten Einzelsysteme von außen betrachtet als ein virtuelles Gesamtsystem erscheinen.

Netzwerkmanagement Das Netzwerkmanagement umfasst die Verwaltung von Kommunikationsdiensten, Netzwerkkomponenten und ihren relevanten Parametern. Es soll für einen fehlerfreien Betrieb des Kommunikationsnetzwerks sorgen.

2.5.2 Integriertes Management

Gemäß [HAN99] erfordert das Management komplexer Systeme den Einsatz eines *integrierten Managements*. Dabei handelt es sich um einen interdisziplinären Ansatz für das einheitliche Management heterogener Ressourcen in einem verteilten IT-Umfeld. Das integrierte Management bewerkstelligt eine zentrale Kontrolle aller Ressourcen und gewährleistet die Einhaltung unternehmensweiter Richtlinien. Ziel ist die Gleichbehandlung der unterschiedlichen Managementobjekte und -aspekte. Die ursprünglich getrennten Disziplinen des Unternehmens-, Anwendungs-, Informations-, System- und Netzwerkmanagements verschmelzen zu einem ganzheitlichen Lösungsansatz, wodurch die Verfügbarkeit und die Leistungsfähigkeit des Gesamtsystems verbessert werden [HAN99]. Das integrierte Management deckt unterschiedliche Aspekte informationsverarbeitender Ressourcen ab. Dabei muss die Herstellervielfalt besondere Berücksichtigung finden. Dies setzt voraus, dass sich die von den Ressourcen bereitgestellten Managementinformationen herstellerunabhängig interpretieren und über standardisierte Schnittstellen und Protokolle abrufen lassen.

2.5.3 Managementarchitektur

Eine Managementarchitektur gilt nach [HAN99] als ein Rahmenwerk für managementrelevante Standards, das vier Teilmodelle umfasst. Das *Informationsmodell* ermöglicht die Beschreibung der Managementobjekte und legt die zur Informationsdarstellung erforderliche Syntax und Semantik fest. Das *Organisationsmodell* beschreibt Organisationsaspekte mit Hilfe von Rollen und Kooperationsformen, Domänen gruppieren Managementobjekte nach fachlichen oder verwaltungsbezogenen Gesichtspunkten. Im *Kommunikationsmodell* werden Zugriffsmechanismen

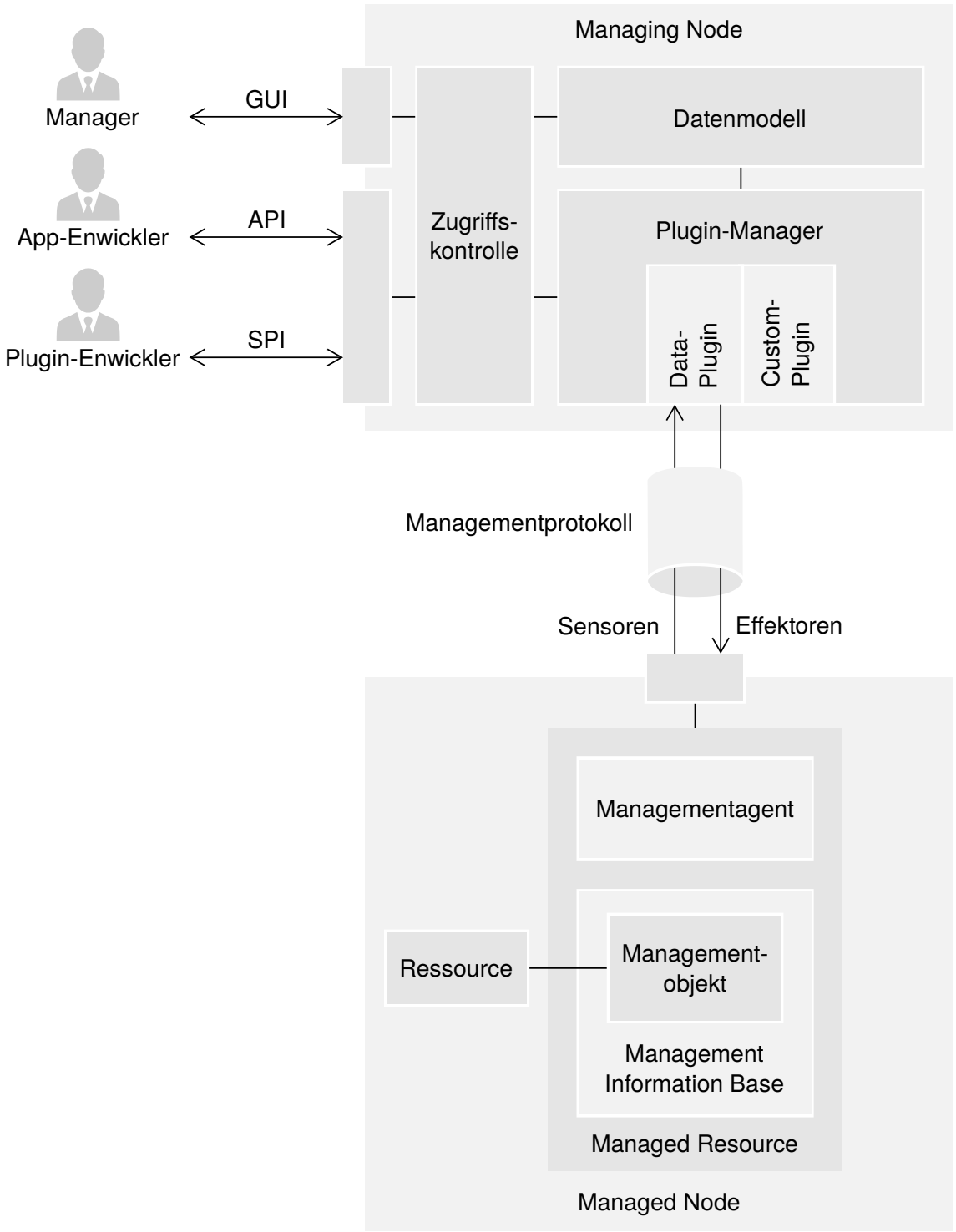


Abbildung 2.12: Referenzarchitektur für das technische Management

und -protokolle definiert. Das *Funktionsmodell* gliedert das Management in einzelne Teilbereiche und spezifiziert die Managementfunktionen. Die Abbildung 2.12 zeigt die Referenzarchitektur für das technische Management. Ihre Bestandteile sollen nachfolgend im Zusammenhang mit den vier Teilmodellen erläutert werden.

2.5.3.1 Informationsmodell

Das Informationsmodell stellt den Beschreibungsrahmen für Managementobjekte bereit und legt ein einheitliches Format für Managementinformationen fest [HAN99]. Dabei handelt es sich um Daten, die zu Managementzwecken ausgetauscht werden. *Managementobjekte* stellen Abstraktionen realer Ressourcen dar und entsprechen der Managementsicht auf eine Ressource [ISO89]. Managementobjekte sind innerhalb einer *Management Information Base* (MIB) organisiert, deren Struktur ebenfalls vom Informationsmodell festgelegt wird. Zur Interaktion bietet die MIB eine Reihe von Schnittstellen, die ein Agent einem Manager in Form eines *Managementservices* zugänglich macht. Das Informationsmodell verwendet eine spezifische Notation zur technologie- und herstellerunabhängigen Beschreibung der Managementinformationen. Neben den reinen Zustandsdaten stellt die MIB auch Informationen über die Organisationsstruktur zur Verfügung.

Managementobjekte

Die Definition eines Managementobjekts erfordert die Angabe seiner Attribute, Operationen, Ereignisse sowie seines Verhaltens [ISO89]. Attribute stellen Managementinformationen in Form von Parametern und Zustandsgrößen bereit. Operationen erlauben das Lesen und Setzen dieser Attribute und arbeiten nach dem *CRUD-Prinzip* (Create, Read, Update, Delete) [ISO89]. Ereignisse entsprechen Notifikationen, die beim Eintritt bestimmter Situationen erzeugt und versendet werden. Operationen lassen sich in *Sensoren* und *Effektoren* unterteilen [Mot03, IBM06]. Über Sensoren können Zustandsgrößen abgefragt und zu Analysezwecken weiterverarbeitet werden, über Effektoren ist ein Einwirken auf die Zustandsgröße möglich. Die Verhaltensdefinition beschreibt die Auswirkungen von Operationsaufrufen auf den Ressourcenzustand.

2.5.3.2 Organisationsmodell

Innerhalb des Organisationsmodells werden die am Managementprozess beteiligten Rollen und Elemente sowie ihre Domänenzugehörigkeit festgelegt.

Rollen und Akteure

Akteure, die steuernd auf andere Systeme einwirken, werden als *Manager* bezeichnet [HAN99]. Dabei kann es sich um ein technisches System oder einen Menschen handeln, dem typischerweise ein grafisches *Frontend* zur Verfügung steht. Das System, das die Ausübung dieser Rolle ermöglicht, wird als *Managing Node* bezeichnet. Dieses verfügt neben einer Ausführungsumgebung über ein internes *Datenmodell*, das sich mit Hilfe eines *Plugin-Managements* erweitern lässt. Dabei können zwei Typen von Plugins unterschieden werden. Ein *Data-Plugin* ermöglicht den Zugriff auf Managementobjekte, die zumeist unterschiedlichen Informationsmodellen angehören. Ein *Custom-Plugin* erlaubt eine Funktionserweiterung des Managing Nodes. Die Rolle des *Entwicklers* lässt sich unterteilen in die des *Anwendungsentwicklers* und die des *Plugin-Entwicklers*. Ihnen steht ein *Standard Development Kit* (SDK) zur Verfügung, das die

Implementierung von Applikationen und Plugins erlaubt. Da Managementinformationen schützenswerte Güter darstellen, sind Maßnahmen zur *Zugriffskontrolle* erforderlich [Eck09]. Neben der Rolle des Managers existiert die Rolle des *Managementagenten*. Es handelt sich vorrangig um einen Kommunikationsagenten, der dem Manager Zugriff auf die Managementobjekte ermöglicht. Grundsätzlich kann ein System auch beide Rollen gleichzeitig einnehmen. Der *Managed Node* stellt eine Ausführungsumgebung für den Agenten bereit und bietet Basisdienste zur Anbindung der MIB. Eine *Managed Resource* ist eine Ressource, die über ein oder mehrere Managementobjekte an eine MIB bzw. einen Managed Node gebunden ist und innerhalb des Managementprozesses sichtbar ist.

Domänen und Kooperationsformen

Domänen stellen eine aufgrund bestimmter Kriterien zusammengefasste Gruppe von Managementobjekten dar, die von einem oder mehreren Managern kontrolliert werden. Sie lassen sich nach Gesichtspunkten der Topologie, Geografie, Organisation oder Funktion strukturieren. Die Zuordnung ist entweder dynamisch oder statisch. Domänen ermöglichen die Bildung mehrstufiger Architekturen. Dabei lassen sich neben der direkten Manager-Agenten-Kommunikation die folgenden Kooperationsformen unterscheiden [HAN99]:

Zentralisiertes Management Die einfachste Kooperationsform folgt dem Modell eines *zentralisierten Managements*, das auf einer einstufigen Architektur basiert. Dabei unterstehen alle Managementobjekte der Kontrolle genau eines Managers. Sie befinden sich in einer gemeinsamen Domäne, die zentral verwaltet wird.

Hierarchisches Management Innerhalb des *hierarchischen Managements* kommen mehrere Domänen zum Einsatz, die jeweils unter der Kontrolle genau eines Managers stehen. Jedes Managementobjekt ist mindestens einer Domäne zugeordnet. Die Manager werden in einer hierarchischen Struktur angeordnet, wobei übergeordnete Manager die Verwaltung untergeordneter Manager übernehmen. Der logische Datenfluss zwischen zwei Managern verläuft stets entlang dieser Struktur.

Verteiltes Management Beim *verteilten Management* überwachen mehrere Manager eine Domäne, wobei jeder Manager einen dedizierten Aufgabenbereich abdeckt. Man spricht auch von einem *funktional verteilten Management* [DH08]. Im Gegensatz zum vorherigen Ansatz sind bei dieser Kooperationsform alle Manager gleichberechtigt.

Wird nicht strikt zwischen einem Managing Node und einem Managed Node unterschieden, sind weitere Kooperationsformen möglich. Diesen Ansätzen liegt die Auffassung zugrunde, dass Managed Nodes immer leistungsfähiger werden und sich zur Dezentralisierung des Managementprozesses zunehmend Funktionen dorthin verlagern lassen:

Management-by-Delegation Beim *Management-by-Delegation* [DH08] wird die Statik der Funktionszuweisung zwischen Manager und Agent aufgehoben, indem Aufgaben an andere Managing Nodes oder direkt an den betroffenen Managed Node weitergereicht werden. Dieses Prinzip ermöglicht die dynamische Zuordnung von Verantwortlichkeiten. Dazu wird auf dem Zielsystem der zu Managementzwecken erforderliche Ausführungscode verfügbar gemacht.

Management-by-Objectives Während im vorherigen Ansatz die Aktionen in Form von ausführbarem Code übertragen werden, erfolgt beim *Management-by-Objectives* [DH08]

eine Zuweisung von abstrakten Zielvorgaben. Es ist Aufgabe des verantwortlichen Systems, die zur Zielerreichung erforderlichen Aktionen selbstständig zu ermitteln und eigenverantwortlich auszuführen.

2.5.3.3 Kommunikationsmodell

Das Kommunikationsmodell definiert Konzepte und Prinzipien für den Austausch von Managementinformationen zwischen den im Organisationsmodell festgelegten Rollen. Dazu spezifiziert ein *Managementprotokoll* den Kommunikationsablauf sowie die ausgetauschten Nachrichten mitsamt ihren Formaten. Diese Ende-zu-Ende-Verbindung zwischen Manager und Agent ermöglicht den Aufruf von Operationen sowie den Versand von Ereignissen. Der Nachrichtenaustausch erfolgt entweder im Pull-, Push- oder hybriden Modus. Im *Pull-Modus* geht die Initiative vom Manager aus, der Informationen vom Agenten abfragt. Erfolgen solche Abfragen innerhalb bestimmter Zeitintervalle, spricht man auch vom *Polling*. Im *Push-Modus* sendet der Agent eigenständig und unaufgefordert Informationen an den Manager. Dieses Verfahren wird zur Ereignisübertragung eingesetzt.

Ereignisse

Ereignisse bilden die Grundlage des *ereignisbasierten Managements* und unterscheiden sich jeweils in Bedeutung und Einsatzzweck. Das ereignisbasierte Management erfordert einen zuverlässigen Nachrichtenaustausch zwischen Absender und Empfänger. Des Weiteren muss sichergestellt sein, dass ausschließlich autorisierten Empfängern Ereignisse zugestellt werden und während der Übertragung keine unautorisierte Informationsgewinnung möglich ist. Ereignisse lassen sich wie folgt klassifizieren:

Alarme Ein *Alarm* informiert den Manager über eine kritische Situation, die zumeist Steuerungseingriffe erforderlich macht. Jedem Alarm liegt die Verletzung einer Bedingung zugrunde, deren Verschwinden durch den Versand einer *Clearing-Nachricht* explizit mitzuteilen ist.

Schwellenwertverletzungen Im Gegensatz dazu weisen *Schwellenwertverletzungen* auf eine verminderte oder zumindest eingeschränkte Qualitäts-/Leistungsfähigkeit von Systemfunktionen hin. Die Über- bzw. Unterschreitungen von Zielvorgaben können, müssen jedoch nicht zu einer Intervention führen. Ebenso wie bei Alarmen erfordert ihr Verschwinden den Versand einer Clearing-Nachricht.

Änderungsereignisse *Änderungsereignisse* beschreiben Zustandsveränderungen innerhalb der MIB, die entweder vom Manager oder von der Ressource selbst ausgelöst wurden. Diese umfassen strukturelle Anpassungen in Form hinzugefügter und entfernter Managementobjekte sowie Änderungen von Attributwerten bestehender Managementobjekte. Neben der Modifikation der Datenbasis lässt sich auch der Zugriff auf die MIB über die Schnittstelle des Managementagenten als Zustandsänderung auffassen.

2.5.3.4 Funktionsmodell

Das Funktionsmodell gliedert das Management in Teilbereiche und legt die Managementfunktionen fest. Es entspricht einer Bibliothek oder einem Baukasten, der für den Entwurf und die Implementierung von Werkzeugen herangezogen werden kann. Innerhalb des Modells sind für die einzelnen Teilbereiche die Funktionalität, die Services sowie die Managementobjekte zu

definieren. Die Funktionen werden im Managing Node oder Managed Node implementiert und über Schnittstellen nutzbar gemacht. Teilbereiche nach dem *FCAPS-Modell* [ISO89] sind:

Fehlermanagement Das Fehlermanagement umfasst die Überwachung des Systemzustands, die Entgegennahme und Verarbeitung von Alarmen sowie die Einleitung von Maßnahmen zur Fehlerbeseitigung. Ziel ist das Erkennen, Isolieren, Beheben und Protokollieren von Fehlersituationen. Fehler stellen Abweichungen von Zielvorgaben dar. Als Fehlerquellen kommen Datenübertragungssysteme, Netzwerkkomponenten, Endsysteme, Software sowie Hardware in Betracht. Es ist Aufgabe des Fehlermanagements, die Ressourcen und Services durch eine schnelle Fehlerentdeckung und -beseitigung verfügbar zu halten. Sind Fehlersymptome erkennbar, ist meist schon der Normalbetrieb des Systems beeinträchtigt. Fehler gilt es somit, noch vor ihrem eigentlichen Eintreten zu erkennen, so dass das Fehlermanagement neben reaktiven auch proaktive Maßnahmen ergreifen kann.

Konfigurationsmanagement Der Begriff „Konfiguration“ besitzt unterschiedliche Bedeutungen. Zum einen umfasst er die Beschreibung eines verteilten Systems hinsichtlich der geografischen Ressourcenanordnung sowie ihrer logischen und physikalischen Beziehungen untereinander. Zum anderen bezeichnet er den Vorgang des Konfigurierens als Aktivität zur Manipulation der Struktur eines verteilten Systems. Dieser Vorgang beinhaltet das Setzen und Ändern von Parametern zur Systemsteuerung. Des Weiteren bezeichnet der Begriff das Ergebnis eines solchen Steuerungsvorgangs, beschrieben als Menge von Parameterwerten. Das Konfigurationsmanagement umfasst alle drei Bedeutungen und schließt das Setzen von Parametern, die Festlegung von Schwellenwerten sowie die Beschreibung des Systems und dessen Veränderungen mit ein. Konfigurieren bedeutet demnach die Anpassung von Systemen an veränderte Betriebs- und Rahmenbedingungen.

Abrechnungsmanagement Die Bereitstellung von Ressourcen zur Verarbeitung und Kommunikation führt zu Kosten, die auf die Verursacher umgelegt werden müssen. Die Aufteilung erfolgt gemäß einer vereinbarten Abrechnungspolitik zwischen Anbieter und Nutzer. Dies setzt ein internes Abrechnungsmodell voraus, das den Zusammenhang zwischen Leistungserbringung und Kosten definiert. Der Prozess der Daten- und Statistiksammlung kann helfen, den Überblick über die Verwaltung und Zuordnung der Ressourcen zu bewahren, so dass sich einer Ressourcenknappheit frühzeitig entgegensteuern lässt. Das Abrechnungsmanagement beinhaltet die Festlegung von Abrechnungsdaten, das Erfassen von Verbrauchsdaten, das Führen von Abrechnungskonten, die Zuordnung von Kosten zu Konten, die Verteilung und Überwachung von Kontingenten, das Führen von Verbrauchsstatistiken sowie die Konfiguration von Tarifen.

Leistungsmanagement Das Leistungsmanagement stellt eine konsequente Weiterführung des Fehlermanagements dar. Während das Fehlermanagement dafür sorgt, dass der System- und Servicebetrieb gewährleistet ist, ist es Aufgabe des Leistungsmanagements sicherzustellen, dass Systemfunktionen und Services gemäß vorab vereinbarter Gütekriterien erbracht werden. Die Überwachung versetzt den Anbieter in die Lage, Ressourcen optimal einzusetzen und Prognosen über die Entwicklung abzugeben. Auf dieser Grundlage kann wiederum ein Nutzer über seine zukünftigen Schritte entscheiden, beispielsweise ob sich durch den Zukauf weiterer Ressourcen Geschwindigkeitsvorteile ergeben. Das Leistungsmanagement umfasst die Festlegung von Gütekriterien, die Bestimmung von Parametern und Metriken, die Durchführung von Messungen, die Aggregation von Messdaten sowie die Durchführung der Leistungs- und Kapazitätsplanung.

Sicherheitsmanagement Das Sicherheitsmanagement enthält alle Maßnahmen, die für einen sicheren und geschützten Systembetrieb erforderlich sind. Sie richten sich sowohl gegen die unsachgemäße Verwendung als auch gegen die Bedrohung von Ressourcen. Bedrohungen können dabei allgemein als Angriffe gegen die Systemsicherheit aufgefasst werden, die entweder passiv oder aktiv ausgeübt werden. Eine systemweite Sicherheitspolitik legt die zugrundeliegenden Anforderungen fest, die die Ableitung geeigneter Maßnahmen ermöglicht. Aufgabe des Sicherheitsmanagements ist die Durchführung von Bedrohungsanalysen, die Erstellung und Durchsetzung der Sicherheitspolitik, die Durchführung der Zugriffskontrolle, die Gewährleistung von Vertraulichkeit und Integrität, die Überwachung des Systems bzgl. Angriffen sowie die kontinuierliche Berichterstattung.

2.5.4 Managementstandards

Für das Management verteilter Systeme existieren zahlreiche standardisierte Managementarchitekturen. Ziel ist die Gewährleistung von Interoperabilität zwischen unterschiedlichen Anbietern und Herstellern. Eine umfangreiche Übersicht bieten [Bla94, Sta98]. Stellvertretend seien an dieser Stelle folgende Managementstandards genannt:

Internet-Management Das *Internet-Management* [CFSD89] wird häufig auch als *SNMP-Management* bezeichnet. Das *Simple Network Management Protocol* (SNMP) wurde von der *Internet Engineering Task Force* (IETF) entwickelt und stellt das derzeit am häufigsten eingesetzte Managementprotokoll innerhalb IP-basierter Netzwerke dar. Das Internet-Management basiert auf einer Client-Server-Architektur, wobei der Client typischerweise als *Manager* und der Server als *Agent* bezeichnet wird. Die *Structure of Management Information* (SMI) legt den Aufbau und die Struktur der Managementobjekte fest. Sie werden in einer hierarchischen Baumstruktur angeordnet und in der Beschreibungssprache *Abstract Syntax Notation One* (ASN.1) [GS90] spezifiziert. Für die Belange des Netzwerkmanagements ist die sogenannte *MIB-II* [Ros90] von Bedeutung. Diese lässt sich zur Verwaltung bestimmter Geräte durch herstellerspezifische MIBs erweitern. Einzelne logisch verknüpfte Informationseinheiten werden zu MIB-Modulen zusammengefasst. Es existiert eine Vielzahl standardisierter MIB-Module, die in [PR98] aufgeführt sind.

OMA Device Management Das *OMA Device Management* (OMA DM) [Ope16b, Ope16c] wird von der *Device Management Working Group* innerhalb der *Open Mobile Alliance* (OMA) entwickelt. OMA DM ist speziell für das Management von Kleinstgeräten ausgelegt und berücksichtigt die besonderen Anforderungen kabelloser Umgebungen mit geringer Bandbreite und hoher Verzögerung. Es basiert auf einer Client-Server-Architektur, wobei ein zentraler *Managementserver* das Gerät konfiguriert. Im Rahmen eines Bootstrappings werden ihm Zugangsdaten mitgeteilt, die einen Verbindungsaufbau ermöglichen. OMA DM verwendet das seit 2002 ebenfalls von der OMA entwickelte *SyncML* [HTMP02], das zur Datenübertragung XML-kodierte Nachrichten verwendet. Das Datenmodell besteht aus *Knoten*, die in einer hierarchischen Baumstruktur, dem sogenannten *Device Management Tree* (DMT), angeordnet sind.

Web-Based Enterprise Management Als *Web-Based Enterprise Management* (WBEM) wird eine Reihe von Standards [DMT10] bezeichnet, die eine plattformübergreifende Basis zum Management verteilter Systeme zur Verfügung stellen [Hob04]. Das *Common Information Model* (CIM) [DMT17] beschreibt das Datenmodell von WBEM und definiert Managementinformationen für Systeme, Netzwerke, Policys, Applikationen und Services. *CIM-XML* ist

ein XML-basiertes Kommunikationsprotokoll, das CIM-konforme Managementinformationen über das HTTP-Protokoll überträgt [DMT13, DMT14b]. *WBEM-Discovery* [DMT14c] verwendet als Discovery-Mechanismus das *Service Location Protocol* (SLP), das es anderen Anwendungen ermöglicht, WBEM-Agenten dynamisch aufzufinden. Die *CIM Query Language* (CQL) [DMT07] ist eine Sprache zur Abfrage von Managementinformationen in CIM-basierten Systemen.

Web Services Distributed Management Das *Web Services Distributed Management* (WSDM) stellt einen Webservice-Standard dar und dient der Überwachung und Verwaltung von Services [OAS06]. Es wurde im Jahre 2006 vom OASIS-Konsortium veröffentlicht und beinhaltet die beiden Spezifikationen *WSDM Management Using Web Services* (MUWS) und *WSDM Management of Web Services* (MOWS). MUWS spezifiziert die Managementschnittstelle einer Ressource in Form eines Webservices. MOWS definiert, wie sich Webservices mittels MUWS verwalten lassen.

2.5.5 Managementsysteme

Managementarchitekturen bilden die Voraussetzung für den Entwurf von Managementsystemen. Implementierungen von Managementarchitekturen werden als *Managementplattformen* bezeichnet [HAN99]. Sie sind Trägersysteme für Managementanwendungen und bilden gemeinsam mit ihnen ein *Managementsystem*. Managementplattformen stellen Entwicklern grundlegende Basisdienste sowie Ausführungs- und Ablaufumgebungen zur Verfügung und unterstützen die vom Funktionsmodell festgelegten Teilbereiche. Grundlage dafür sind die einheitliche Darstellung, Speicherung und Verwaltung von Managementobjekten. Zur Anbindung unterschiedlicher Managementprotokolle lassen sich Plattformen um zusätzliche Kommunikationsmodule erweitern. Funktionen zur Datenaggregation und -analyse lassen sich im Sinne einer Datenvorverarbeitung auf andere Verarbeitungsknoten verlagern, so dass die Ausführung des Managementprozesses nicht mehr länger auf nur eine einzelne Plattform beschränkt bleibt.

2.5.6 Policys

Das Management von IT-Systemen muss an den Unternehmenszielen ausgerichtet sein. Es umfasst sämtliche Maßnahmen, die der Zielerreichung dienen und sorgt für die Einhaltung von (abstrakten) Regeln, Richtlinien und Vorschriften [Mar97, Dam02]. Diese Vorgaben werden als *Policys* bezeichnet [Slo94]. Sie sind aus der Notwendigkeit heraus entstanden, immer komplexer werdende Systeme beherrschen zu müssen. In diesem Zusammenhang nennt man das technische Management auch *policybasiertes Management*, das gemäß [MESW01, Ver00] folgende Bestandteile aufweist:

Policy Repository Das *Policy Repository* (PR) verwaltet die Policys und dient dem Managementprozess als zentrale Wissensbasis.

Policy Decision Point Der *Policy Decision Point* (PDP) trifft auf Grundlage der bereitgestellten Policys Steuerungsentscheidungen.

Policy Enforcement Point Der *Policy Enforcement Point* (PEP) ist verantwortlich für die Durchsetzung und Einhaltung der vom PDP getroffenen Entscheidungen.

Der Nachrichtenaustausch zwischen PDP und PEP kann über das *Common Open Policy Service Protocol* (COPS) [DBC+00] erfolgen. Des Weiteren sieht das in [DLSD01] beschriebene Vorgehen

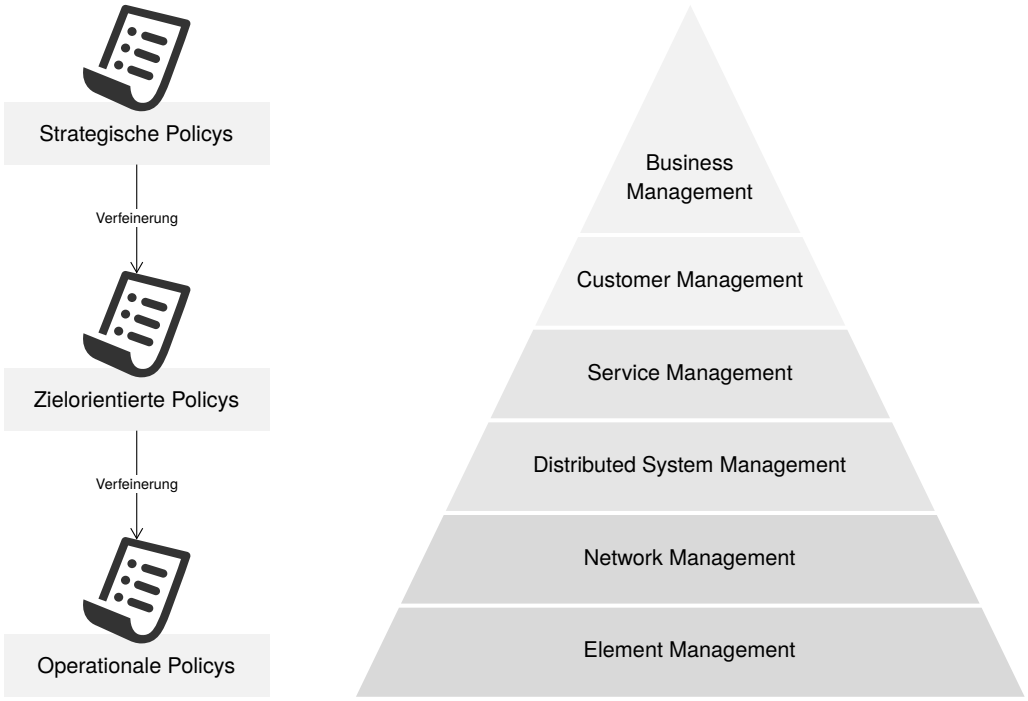


Abbildung 2.13: Policy-Hierarchie nach [Koc96] und Managementpyramide nach [HAN99]

vor, PDPs zur effizienteren Policyauswertung in der IT-Umgebung zu verteilen. In solchen Managementszenarien kommen zumeist mehrere PEPs gleichzeitig zum Einsatz, so dass der PDP zusätzlich festlegen muss, welcher PEP die Durchsetzung einer Policy zu verantworten hat. Für die Policy-Spezifikation lassen sich unterschiedliche Sprachen einsetzen. Stellvertretend seien an dieser Stelle *Ponder2* [DTSL08], die *Autonomic Computing Policy Language* (ACPL) [ALL05] und *WS-Policy* [BGP06, Wor07] genannt. Eine umfangreiche Übersicht bietet [Dam02].

Policy-Hierarchie

Policys unterscheiden sich bezüglich Abstraktionsniveau und Detaillierungsgrad [WSS+01]. Gemäß Abbildung 2.13 entsprechen sie *strategischen*, *zielorientierten* oder *operationalen Policys* [Koc96]. Eine Schicht in der Managementpyramide stellt jeweils Anforderungen an die direkt nachfolgende Schicht [HAN99]. Die dazugehörigen Policys lassen sich somit hierarchisch anordnen [Wie95, Koc96]. Dabei kann zwischen deklarativen und imperativen Policys unterschieden werden [DKK+11b]. *Deklarative Policys* beschreiben einen Sollzustand in Form von Zielvorgaben, wohingegen *imperative Policys* Handlungsanweisungen festlegen. Es besteht eine Lücke zwischen den definierten Zielen und den daraus gefolgerten Maßnahmen zur Zielerreichung. Diese Lücke wird mit Hilfe einer *Policy-Hierarchie* [MC93, MS93, Wie95, Koc96] geschlossen. Dabei werden Policys zunächst auf einem hohen Abstraktionsniveau definiert und anschließend schrittweise so weit verfeinert, bis sie ein hinreichend niedriges Abstraktionsniveau besitzen. Dieser Vorgang wird als *Policy-Refinement* bezeichnet [BLMR04, Lüc06, DKK+11b]. In der Regel wächst die Anzahl der Policys mit zunehmendem Detaillierungsgrad.

3
Verwandte Arbeiten

Das Kapitel bietet einen Überblick über ausgewählte Forschungs- und Standardisierungsarbeiten, die einen Bezug zum Management serviceorientierter Systeme in einer Multi-Provider-Umgebung aufweisen. Den Schwerpunkt bilden Lösungsansätze aus dem Bereich des technischen Managements, die Informationsmodelle zur Serviceüberwachung und -steuerung einsetzen. Derzeit ist das Management von Multi-Cloud-Umgebungen noch Gegenstand der Forschung [AP17, HBV17, RR18]. Es stellt hinsichtlich des Managements einer Multi-Provider-Umgebung zwar nur ein, dafür aber sehr wesentliches Teilproblem dar. Seine Ursache lässt sich auf eine unzureichende Standardisierung zurückführen. Die Folge ist eine fehlende Interoperabilität zwischen den Ausführungsumgebungen und Managementsystemen [Pet11, EMPR12, Lew13]. Dies erschwert die anbieter- und umgebungsübergreifende Steuerung sowie den Austausch von Managementinformationen.

3.1 Forschungsarbeiten

Die nachfolgenden Abschnitte stellen Forschungsbeiträge vor, die sich mit anwendungsspezifischen Managementarchitekturen und dem Entwurf von Informationsmodellen beschäftigen. Es lassen sich eigenständige Informationsmodelle von Erweiterungen standardisierter Informationsmodelle unterscheiden. Eigenständige Informationsmodelle existieren unabhängig von einem Managementstandard und werden zumeist mit Hilfe von UML spezifiziert. Ihre Anwendung erfordert die Abbildung in ein bestehendes Informationsmodell. Die Transformation kann entweder manuell oder (teil-)automatisiert erfolgen. Das Ergebnis sind deklarative Schemadefinitionen, die Struktur und Aufbau der Managementobjekte beschreiben und ihre Beziehungen untereinander festlegen.

3.1.1 Anwendungsspezifische Managementarchitekturen

Anwendungsspezifische Managementarchitekturen sind auf das Management spezieller Anwendungsdomänen zugeschnitten. Die im Folgenden beschriebenen Beiträge befassen sich mit dem Customer-Service-Management, dem Management dynamischer Virtueller Organisationen in Grids und dem Service-Level-Management.

Konzeption und Anwendung einer Customer-Service-Management-Architektur

Michael Langer definiert in seiner Dissertation [Lan01] eine Managementarchitektur für das *Customer-Service-Management* (CSM). Das CSM umfasst die Bereitstellung und Weitergabe von Informationen über die Servicegüte. Dadurch kann ein Kunde die Einhaltung der vertraglich vereinbarten Anforderungen überwachen. Im Rahmen der Arbeit wird eine CSM-Schnittstelle definiert, über die sich die bisher getrennten Managementsysteme von Kunde und Anbieter koppeln lassen. Dadurch kann ein Austausch von Managementinformationen zwischen den Organisationen erfolgen. Der Schwerpunkt der Arbeit liegt auf der Integration der CSM-Schnittstelle in die Managementumgebung des Anbieters. Dabei stehen Wiederverwendbarkeit und Anpassbarkeit im Vordergrund. Die CSM-Managementarchitektur umfasst ein Informations-, Organisations-, Kommunikations- und Funktionsmodell. Das Informationsmodell definiert Managementobjekte für Vertragsbeziehungen, Nutzer, Rollen und QoS. Das Organisationsmodell umfasst die Organisationsaspekte in Form von CSM-spezifischen Rollen und Kooperationsformen. Das Kommunikationsmodell spezifiziert die dazugehörigen Nutzungs- und Interaktionsbeziehungen. Das Funktionsmodell zergliedert das CSM in die Funktionsbereiche SLA-, Problem-, Order-, Configuration-, Accounting-, und Service-Infrastructure-Management. Zudem legt die CSM-Managementarchitektur fest, zu welchen Informationsquellen innerhalb des Servicemanagements des Anbieters Nutzungsbeziehungen bestehen müssen und wie sich die Informationen dazu einsetzen lassen, um dem Kunden eine übergeordnete CSM-Schnittstelle zur Verfügung zu stellen.

Management dynamischer Virtueller Organisationen in Grids

Michael Schiffers entwickelt in seiner Dissertation [Sch07] eine Managementarchitektur für den Aufbau und die Verwaltung *Virtueller Organisationen* (VOs) in Grids. VOs bestehen aus Personen und technischen Ressourcen realer Organisationen und ermöglichen eine kooperative Problemlösung in einem verteilten Computing-Umfeld. Die Bildung von VOs erfolgt stets zweckorientiert, alle Mitglieder verfolgen ein gemeinsames Ziel. Anders als bei realen Organisationen existieren VOs nur für einen befristeten Zeitraum und verfügen über eine hohe Dynamik hinsichtlich Aufbau, Struktur und Verhalten. Ziel der Dissertation ist die Definition einer VO-Managementarchitektur (VOMA), in der VOs als Managementobjekte repräsentiert sind. Das Informationsmodell legt die Managementinformationen fest, die zwischen den beteiligten Akteuren ausgetauscht werden. Das Organisationsmodell identifiziert die am VO-Management beteiligten Rollen und ordnet ihnen entsprechende Verantwortlichkeiten zu. Das Kommunikationsmodell spezifiziert die Abhängigkeits- und Interaktionsbeziehungen. Das Funktionsmodell orientiert sich am VO-Lebenszyklus und gliedert das VO-Management in die Funktionsbereiche Configuration-Management, VO-Provisioning, Accounting-Management, Local Management, VO-Management, Member-Management und Resource-/Service-Management. Die Definition der VOMA erfolgt plattformunabhängig. Zu Validierungszwecken werden eine WSDM-spezifische Transformation und die Abbildung des Informationsmodells auf CIM durchgeführt.

IT-gestütztes Service-Level-Management – Anforderungen und Spezifikation einer Managementarchitektur

Thomas Schaaf befasst sich in seiner Dissertation [Sch08] mit der Entwicklung einer Architektur für das IT-gestützte Service-Level-Management (SLM). Es wird ein werkzeugorientiertes SLM-Managementsystem entworfen, das auf einem Informations-, Organisations-, Kommunikations-

und Funktionsmodell basiert. Dazu werden die vier Teilmodelle jeweils aus einer Prozess- und Systemsicht heraus betrachtet. Die Prozesssicht beschäftigt sich mit der Umsetzung eines SLM-unterstützenden Managementsystems ohne Berücksichtigung von Automatisierungsaspekten. Zentrale Bestandteile sind Verfahrens- und Prozessabläufe, Akteure, Kommunikationsflüsse und Informationsartefakte in Form von Dokumenten, Servicegütevereinbarungen und Berichten. Die Systemsicht baut auf den Modellen der Prozesssicht auf und erweitert diese um Komponentenarchitekturen, Datenmodelle und Managementfunktionen. Das Ergebnis ist eine Sammlung von funktionalen Anforderungen, Anwendungsfällen sowie von Prozess- und Systemmodellen zur Umsetzung und Automatisierung des SLMs.

3.1.2 Informationsmodelle

Die nachfolgenden Abschnitte stellen Forschungsbeiträge vor, die sich mit Informationsmodellen zur Strukturierung und einheitlichen Beschreibung serviceorientierter Systeme beschäftigen. Ziel ist die Identifikation der managementrelevanten Ressourcen sowie der Einsatz bestehender Managementstandards zur Verwaltung traditioneller IT-Systeme.

MNM-Dienstmodell

Bei dem MNM-Dienstmodell [GHH+01, GHK+01] handelt es sich um ein Dienstmodell zur Beschreibung serviceorientierter Systeme, das sich unabhängig vom konkreten Anwendungsszе-

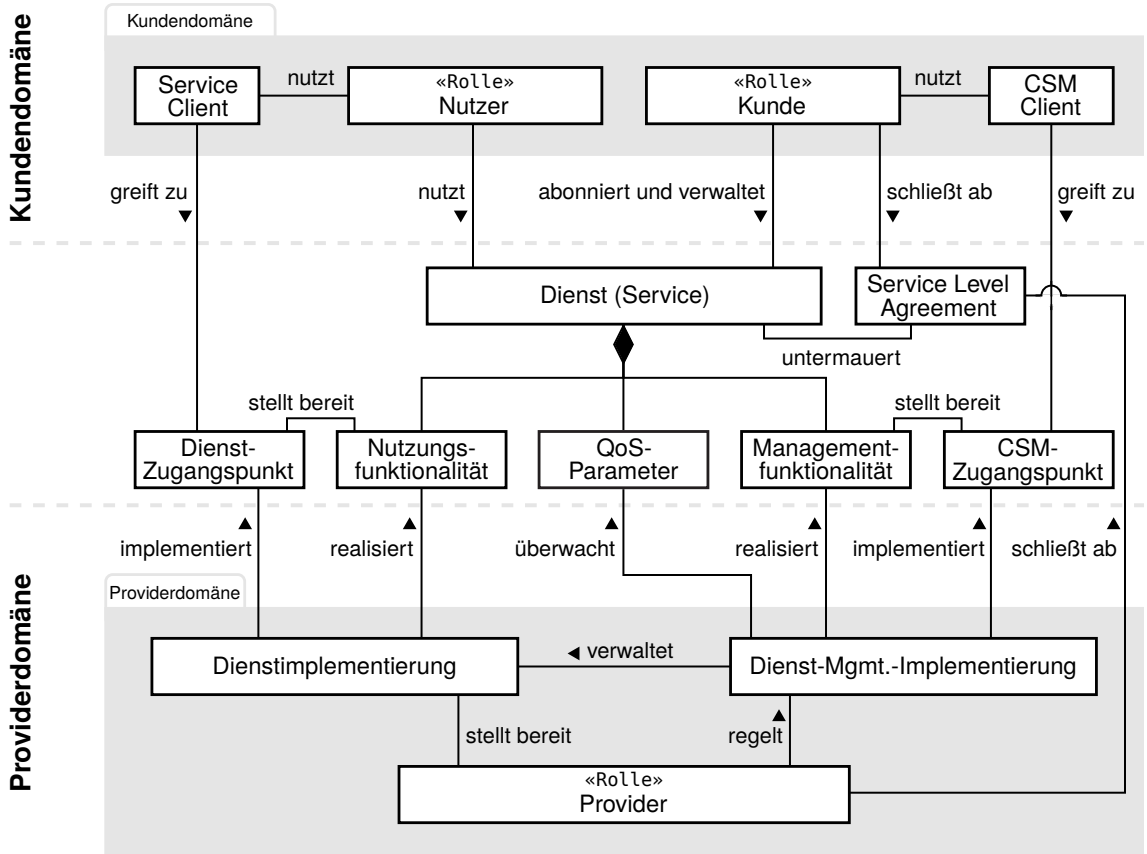


Abbildung 3.1: MNM-Dienstmodell nach [GHH+01, GHK+01]

nario verwenden lässt. Die Abbildung 3.1 zeigt seine zentralen Elemente. Es dient dem Aufbau einer einheitlichen Terminologie für das Servicemanagement. Neben der Servicemodellierung erfolgt zugleich eine Festlegung der Rollen, die an der Bereitstellung und Nutzung beteiligt sind. Teile eines Services, die Kunden und Nutzer betreffen, werden der *Kundendomäne* zugeordnet, Teile, die den Anbieter betreffen, der *Anbieterdomäne*. Services stellen Anwendungsfunktionen bereit und bieten Schnittstellen, über die sich Anwendungs- und Managementfunktionen nutzen lassen. Gemäß der SOA-Prinzipien werden Services und ihre Implementierungen durch eigenständige Beschreibungselemente repräsentiert.

Modellbasiertes Service Level Management verteilter Anwendungssysteme

Markus Debusmann entwickelt in seiner Dissertation [Deb05] einen Ansatz für ein plattformunabhängiges Service-Level-Management. Ziel ist die Vereinheitlichung der Modellierung, so dass sich unterschiedliche Managementansätze integrieren und eine Trennung zwischen Problem- und Technologiedomäne erzielen lässt. Zentrale Idee der Arbeit ist die Definition von SLA-Mustern, die konfigurationsunabhängige Abstraktionen von Service-Level-Agreements darstellen. Durch eine Modelltransformation wird aus einem SLA-Muster eine SLA-Instanz abgeleitet, die alle notwendigen Konfigurationsinformationen beinhaltet und im Format der Zielplattform vorliegt. Zur Laufzeit kann das Managementsystem die Einhaltung der SLAs überwachen. Im Rahmen der Arbeit wurde eine UML-Erweiterung erstellt, die die Modellierung von SLA-Mustern mit Hilfe eines UML-Werkzeugs ermöglicht. Die *Object Constraint Language* (OCL) dient der Spezifikation von Funktionsvorschriften zur Berechnung kundenspezifischer Metriken. Auf Grundlage einer plattformunabhängigen Modellierung lässt sich, abgestimmt auf das Managementsystem, Code zur Etablierung und Überwachung der SLAs generieren.

Konzeption einer Service-MIB – Analyse und Spezifikation dienstorientierter Managementinformation

Martin Sailer befasst sich in seiner Dissertation [Sai07] mit der Konzeption einer serviceorientierten Informationsbasis, die als *Service-MIB* bezeichnet wird. Die Grundlage dafür bildet ein vierstufiges Vorgehensmodell, das sich in eine Analyse-, Spezifikations-, Überwachungs- und Nutzungsphase unterteilt. In der Analysephase wird der eigentliche Informationsbedarf aus Sicht des Managements ermittelt. In der Spezifikationsphase erfolgt die Modellierung und Beschreibung der Managementinformationen mit Hilfe der eigens entwickelten Spezifikationssprache namens *Service Information Specification Language* (SISL). In der Überwachungsphase gewährleistet ein Werkzeug für das Servicemonitoring die Aktualität der Datenbasis. Die sogenannte *Service Monitoring Architecture* (SMONA) ermöglicht eine Überwachung von Serviceeigenschaften basierend auf den SISL-Spezifikationen. Die SMONA dient als Bindeglied zur vorhergehenden Spezifikationsphase. In der Nutzungsphase erfolgt der Zugriff auf die Managementobjekte, auf deren Grundlage die Managementfunktionen bereitgestellt werden.

Semantisches Informationsmodell für die Betriebsunterstützung dienstorientierter Systeme

Frederic Majer entwickelt in seiner Dissertation [Maj10] ein semantisches Informationsmodell zur Betriebsunterstützung serviceorientierter Systeme. Dabei werden grundlegende Rollen und Aufgaben identifiziert und definiert. Ein domänenspezifisches Informationsmodell auf Grundlage von Ontologien dient als Mittel zur Vereinheitlichung der Managementinformationen. Das

Informationsmodell bildet den Ausgangspunkt zur Berechnung von Servicegüteeigenschaften sowie zur Durchsetzung von Servicegüteanforderungen. Die im Rahmen der Arbeit entwickelte *integrated information map* (i2map) dient der Verwaltung und Überwachung serviceorientierter Systemlandschaften. Hierbei führt die i2map die Managementinformationen, die jeweils unterschiedlichen Datenquellen entstammen, in einem gemeinsamen Modell zusammen.

Towards an Infrastructure Description Language for Modeling Computing Infrastructures

Der von [GHGL12] vorgeschlagene Lösungsansatz sieht eine technologieunabhängige Beschreibung von Infrastrukturressourcen mit Hilfe der eigens entwickelten Spezifikationssprache namens *Infrastructure and Network Description Language* (INDL) vor. Diese ermöglicht eine Erfassung von physikalischen und virtuellen Host-Systemen sowie von Ressourcen der Netzwerkschicht. Der Schwerpunkt liegt auf der Repräsentation von Netzwerktopologien, von föderierten Berechnungsinfrastrukturen sowie von optischen Netzwerken.

3.1.3 Erweiterungen standardisierter Informationsmodelle

Die nachfolgenden Abschnitte stellen Forschungsbeiträge vor, die standardisierte Informationsmodelle um zusätzliche Managementobjekte erweitern. Dabei handelt es sich um MIB-Definitionen für SNMP sowie um CIM-Schemata für WBEM. Den Schwerpunkt bilden Lösungsansätze, die sich mit der Ressourcenrepräsentation im Umfeld cloudbasierter und virtueller Umgebungen beschäftigen.

Aggregating IaaS Service

Der Forschungsbeitrag von [LYMZ11] befasst sich mit der einheitlichen Beschreibung von Infrastrukturressourcen in cloudbasierten Umgebungen. Zu diesem Zweck wurde ein sogenannter *Infrastructure-as-a-Service Aggregator* (IaaSA) entwickelt. Dieser kann auf der einen Seite an unterschiedliche IaaS-Anbieter angebunden werden und bietet auf der anderen Seite den Nutzern eine vereinheitlichte Ressourcenrepräsentation. Der Ansatz erweitert CIM um zusätzliche Managementobjektklassen für die Bereiche Compute, Netzwerk und Storage, die unter anderem virtuelle Maschinen und IP-Adressen umfassen.

A SNMP-based Virtual Machines Management Interface

Der Lösungsansatz von [HB12] definiert eine *Virtual-Machines-MIB* für das SNMP-basierte Management virtueller Maschinen. Neben der einheitlichen Überwachung bietet die MIB Möglichkeiten, steuernd auf den Lebenszyklus einzuwirken. Dabei wird das Erstellen, Löschen, Neustarten, Ein- und Ausschalten sowie Pausieren von virtuellen Maschinen unterstützt. Gleichwohl ist eine vertikale Skalierung möglich, indem sich über das Managementobjekt CPUs hinzufügen und der Speicherplatz vergrößern lässt. Der Zugriff auf das Virtualisierungssystem erfolgt über *libvirt*, einer Sammlung quelloffener Werkzeuge und Schnittstellen zur hypervisorübergreifenden Verwaltung virtueller Maschinen auf einem Wirtssystem.

A CIM (Common Information Model) based Management Model for Clouds

Der in dem Forschungsbeitrag von [Red12] beschriebene Ansatz sieht für das einheitliche Management von Applikationen und Infrastrukturen in cloudbasierten Umgebungen eine

Erweiterung von CIM vor. Es werden grundlegende Managementobjekte identifiziert, die Eigenschaften und Ressourcen für das WBEM-basierte Cloud-Management nutzbar machen. Sie repräsentieren Ressourcen, die in den Funktionsbereichen Server-/Hypervisor-Management, Virtualization-Management, Storage-Management, Data-Management und Networking-Infrastructure-Management Verwendung finden.

CIM-SDN: A Common Information Model Extension for Software-defined Networking

Der Forschungsbeitrag von [PCCA13] befasst sich mit einer Erweiterung von CIM zur Repräsentation von Ressourcen aus dem Bereich des *Software-defined Networking* (SDN). Mit Hilfe der OCL werden Invarianten definiert, wodurch sich fehlerhafte Konfigurationen sowie Unstimmigkeiten im Netzwerk-Entwurf schneller auffinden und beseitigen lassen. CIM-SDN ermöglicht es, Netzwerkressourcen herstellerunabhängig unter Einsatz von WBEM einheitlich zu überwachen und zu konfigurieren.

3.2 Arbeiten von Standardisierungsgremien

Die nachfolgenden Abschnitte stellen Spezifikationen internationaler Standardisierungs- und Industriegremien vor, die sich mit dem Problem des einheitlichen Managements heterogener Ressourcen in einem verteilten IT-Umfeld befassen. Dies beinhaltet Beiträge der Distributed Management Task Force, des Open Grid Forums sowie der Organization for the Advancement of Structured Information Standards.

3.2.1 Distributed Management Task Force

Die im Jahre 1992 gegründete *Distributed Management Task Force* (DMTF) ist eine internationale Normungsorganisation, die aus über 200 führenden Herstellern von Netzwerkprodukten und Managementlösungen besteht. Hersteller und Anwender sind in gemeinsamen Arbeitsgruppen organisiert. Sie befassen sich mit der Entwicklung und Verbreitung von Standards für das technische Management. Vorrangiges Ziel ist die Vereinfachung und Vereinheitlichung des Managements vernetzter Systeme.

Open Virtualization Format

Das *Open Virtualization Format* (OVF) [DMT14a] ist ein offener Standard zur Paketierung und Verteilung von Softwareanwendungen für virtuelle Maschinen. Mit dem OVF lassen sich *virtuelle Appliances* über unterschiedliche hypervisorbasierte Virtualisierungssysteme hinweg verwenden. Bei virtuellen Appliances handelt es sich um vorinstallierte, vorkonfigurierte und sofort einsetzbare Softwareanwendungen, die bereits mit einem Betriebssystem in der virtuellen Maschine zusammengestellt sind. OVF dient der Interoperabilität und Portabilität von virtuellen Maschinen. Ein OVF-Paket enthält einen OVF-Deskriptor. Hierbei handelt es sich um ein XML-basierte Datei, die Name, Hardwarevoraussetzungen und Verweise auf andere Dateien enthält. Darüber hinaus enthält das OVF-Paket in der Regel eine Netzwerkbeschreibung, eine Liste der virtuellen Hardware, Informationen zu den virtuellen Laufwerken sowie Zertifikatsdateien und Informationen über das Betriebssystem.

Common Information Model

Das *Common Information Model* (CIM) [DMT17] ist ein objektorientiertes Informationsmodell und Bestandteil von WBEM. Es dient der einheitlichen Beschreibung von Managementinformationen. CIM repräsentiert die verwalteten Ressourcen als Managementobjekte und legt Aufbau und Struktur sowie die Beziehungen unter den Managementobjekten fest. Die erste Version erschien im Jahre 1997. Seitdem wurde das Informationsmodell kontinuierlich weiterentwickelt, verfeinert und angepasst. Es liegt derzeit in der Version 2.51.0 (Stand: Dezember 2018) vor und kann als UML-Klassendiagramm oder als textuelle Repräsentation im *Managed Object Format* (MOF) abgerufen werden. CIM ist zur Verbesserung der Erweiterbarkeit in die folgenden vier Ebenen unterteilt:

Metamodell Das *Metamodell* bestimmt die syntaktische Struktur von CIM und legt die Darstellungsform der Modellelemente durch Klassen, Attribute und Assoziationen fest. Eine Besonderheit von CIM ist der Einsatz strikter Vererbung. Dadurch lassen sich Methoden und Attribute von übergeordneten Klassen in den jeweils untergeordneten Klassen nicht überschreiben.

Kernmodell Das *Kernmodell* umfasst nur wenige Klassen, deren Aufgabe darin besteht, gemeinsame Eigenschaften aller Elemente des CIM-Modells zu repräsentieren. Sie sollen sich in allen Bereichen und Disziplinen des technischen Managements einsetzen lassen. Das Kernmodell definiert eine Grundmenge an Klassen, Eigenschaften und Assoziationen, die jeweils den Ausgangspunkt von Verfeinerungen bilden.

Gemeinsames Modell Das *gemeinsame Modell* verfeinert die Klassen des Kernmodells und ordnet sie Managementbereichen zu. Dabei müssen die Klassen technologie- und implementierungsneutral sein, aber dennoch konkret genug, um als Grundlage für Managementanwendungen zu dienen. Das Modell deckt gegenwärtig die folgenden Managementbereiche ab: Application, Database, Device, Event, Interop, IpsecPolicy, Metrics, Network, Physical, Policy, Security, Support, System und User.

Erweiterungsschemata Die unterste Ebene umfasst technologiespezifische Erweiterungen. Dabei werden die Klassen des gemeinsamen Modells um zusätzliche Eigenschaften und Assoziationen verfeinert. Die Ebene enthält unter anderem herstellerspezifische Erweiterungen zur Repräsentation von Betriebssystemen wie Unix, Linux und Windows.

CIM umfasst derzeit über 1 600 Klassen und Assoziationen und stellt damit ein ausdrucksstarkes Informationsmodell für die Beschreibung von IT-Systemen dar. Aufgrund seines Umfangs, der Verflechtung der Klassen und der Tiefe der Vererbungshierarchie gilt es aber auch als teilweise übermäßig komplex und schwer handhabbar [KKS01].

Cloud Infrastructure Management Interface

Das *Cloud Infrastructure Management Interface* (CIMI) [DMT16] ist ein offener Standard, der im Jahre 2013 erschienen ist. Er definiert eine Managementschnittstelle in Form eines RESTful Webservices zur Verwaltung von Cloud-Infrastrukturen. Die Spezifikation legt die Informationsobjekte sowie das Austauschprotokoll fest. Es regelt die Interaktion zwischen Cloud-Umgebungen sowie zwischen Providern und Nutzern. Die vom Standard festgelegten Managementobjekte repräsentieren Systeme, Maschinen, Netzwerke und Volumes. Die Verteilung von Softwareanwendungen erfolgt mittels OVF. Der Standard hat sich bisher nicht durchsetzen können. Er wird

derzeit von keinem der großen kommerziellen Cloud-Anbietern unterstützt, auch eine Umsetzung in die offene Cloud-Plattform *OpenStack* hat nicht stattgefunden. Eine Implementierung erfolgte nur im Rahmen von *Apache Deltacloud*, diese stellt eine übergeordnete Managementschnittstelle zur Verfügung, um unterschiedliche Cloud-Umgebungen einheitlich ansteuern zu können. Das Projekt wurde aufgrund mangelnder Aktivität im Jahre 2015 eingestellt.

3.2.2 Open Grid Forum

Das *Open Grid Forum* (OGF) ist eine Gemeinschaft von Anwendern, Entwicklern und Anbietern, die die Entwicklung des verteilten Rechnens vorantreibt. Es wurde im Jahre 2006 als Zusammenschluss des *Global Grid Forums* und der *Enterprise Grid Alliance* gegründet und entwickelt Standards für das Grid- und Cloud-Computing.

Open Cloud Computing Interface

Das *Open Cloud Computing Interface* (OCCI) [Ope16a] ist ein offener Standard, der erstmals im Jahre 2010 veröffentlicht wurde. OCCI spezifiziert, ähnlich wie CIMI, eine Managementschnittstelle auf Grundlage eines RESTful Webservices zur Verwaltung von Ressourcen in cloudbasierten Umgebungen. Der Standard besteht aus einem Kernmodell sowie Erweiterungen zur Beschreibung von Infrastrukturen, Plattformen und Service-Level-Agreements.

3.2.3 OASIS

Die *Organization for the Advancement of Structured Information Standards* (OASIS) ist ein international agierendes gemeinnütziges Konsortium, das die Entwicklung, Abstimmung und Übernahme von Standards vorantreibt. Das Konsortium hat über 5 000 Mitglieder sowie mehr als 600 Organisationen aus über 65 Ländern. Es wurde im Jahre 1993 unter dem Namen *SGML Open* gegründet und im Jahre 1998 zu OASIS umbenannt.

Cloud Application Management for Platforms

Das *Cloud Application Management for Platforms* (CAMP) [OAS14] ist ein Standard zur Verwaltung und Steuerung von Cloud-Anwendungen in PaaS-Umgebungen. Die Spezifikation definiert eine Managementschnittstelle und ein Paketformat, um Anwendungen auf Plattformen bereitstellen, überwachen und anpassen zu können. Der Standard soll die Interoperabilität zwischen unterschiedlichen PaaS-Plattformen sicherstellen. Zu diesem Zweck werden Anforderungen aufgestellt, die von den PaaS-Anbietern umzusetzen sind. Dies soll gewährleisten, dass sich PaaS-Services unabhängig von der zugrundeliegenden Infrastruktur nutzen lassen. Mit *OpenStack Solum* und *Apache Brooklyn* liegen zwei Implementierungen von CAMP vor.

Topology and Orchestration Specification for Cloud Applications

Die *Topology and Orchestration Specification for Cloud Applications* (TOSCA) [OAS13] ist ein Standard für die automatisierte Bereitstellung und Verwaltung von Cloud-Anwendungen. TOSCA ermöglicht es, die Struktur einer Anwendung, die Abhängigkeiten der Softwarekomponenten sowie die zur Ausführung erforderlichen Infrastrukturressourcen plattformunabhängig zu beschreiben. Die Struktur einer Cloud-Anwendung wird mit Hilfe von Topology-Templates erfasst. Ein *Topology-Template* ist ein gerichteter Graph, der aus Knoten und Kanten besteht. Die Knoten (*Node-Templates*) repräsentieren die Softwarekomponenten der Anwendung. Die Kanten

(*Relationship-Templates*) stellen die Abhängigkeiten zwischen den Softwarekomponenten dar. Die Semantik von Node-Templates und Relationship-Templates wird mit Hilfe von *Node-Types* und *Relationship-Types* definiert. Node-Types können Eigenschaften sowie parametrierbare Managementoperationen definieren, mittels derer sich Instanzen dieses Node-Types verwalten lassen. Die Definitionen werden in einem *Service-Template* zusammengefasst, das gemeinsam mit den Softwareartefakten und Ressourcen in einem *Cloud-Service-Archive* (CSAR) ausgeliefert wird. Das Ergebnis ist ein vollständiger Bauplan einer Cloud-Anwendung, der von der TOSCA-Laufzeitumgebung zur Ausführung gebracht werden kann und die Bereitstellung der Anwendung bewirkt.

4

Problemanalyse

Die Problemanalyse dient der Bestandsaufnahme und der Erfassung des gegenwärtigen Ist-Zustands im Hinblick auf das Management serviceorientierter Systeme in einer Multi-Provider-Umgebung. Eine Analyse der Ausgangssituation zeigt zunächst den Handlungsbedarf auf und identifiziert die zentralen Problemfelder. In diesem Zusammenhang werden Rahmenbedingungen, Umfeld und Entwicklungen erläutert, die das einheitliche Management heterogener Ausführungsumgebungen erschweren. Im Anschluss erfolgt die Formulierung der Zielsetzung. Die daraus abgeleiteten Anforderungen bilden den Ausgangspunkt der Lösungserstellung.

4.1 Ausgangssituation

Serviceorientierte Systeme bestehen aus einer Vielzahl lose miteinander gekoppelter Services. Bereitstellung und Betrieb erfolgen auf Grundlage unterschiedlicher Ausführungsumgebungen, Technologien und Implementierungen. Ein reibungsloser Betrieb erfordert fortwährende Überwachung der Services selbst, ihrer Beziehungen untereinander sowie der beteiligten Hardware- und Softwarekomponenten.

4.1.1 Beziehungsgeflecht zwischen Services

Die Abbildung 4.1 zeigt den schematischen Aufbau eines serviceorientierten Systems. Es besteht aus den folgenden drei Ebenen: die unterste Ebene umfasst die am Verbund beteiligten Ausführungsumgebungen, die mittlere Ebene enthält die auf Host-Systemen (virtuellen Maschinen, Containern, Bare-Metal-Servern) installierten Softwarekomponenten und die oberste Ebene beinhaltet die davon bereitgestellten Services. Beziehungen existieren sowohl zwischen als auch innerhalb der Ebenen. Services bilden ein komplexes Geflecht aus Abhängigkeits- und Nutzungsbeziehungen, das es zu erfassen und zu kontrollieren gilt. Ein Service greift zur Funktionsbereitstellung in der Regel auf Funktionen anderer Services zurück, das Binden kann statisch oder dynamisch erfolgen. Unterschiedliche Störgrößen beeinflussen die qualitativen Eigenschaften der Nutzungsbeziehung. Dazu zählen zeitliche Verzögerungen beim Nachrichtenaustausch und der Anfragebearbeitung. Des Weiteren sind dies Fehlersituationen, die durch Hardware- und Softwarefehler sowie fehlerhafte Konfigurationen ausgelöst werden. Beein-

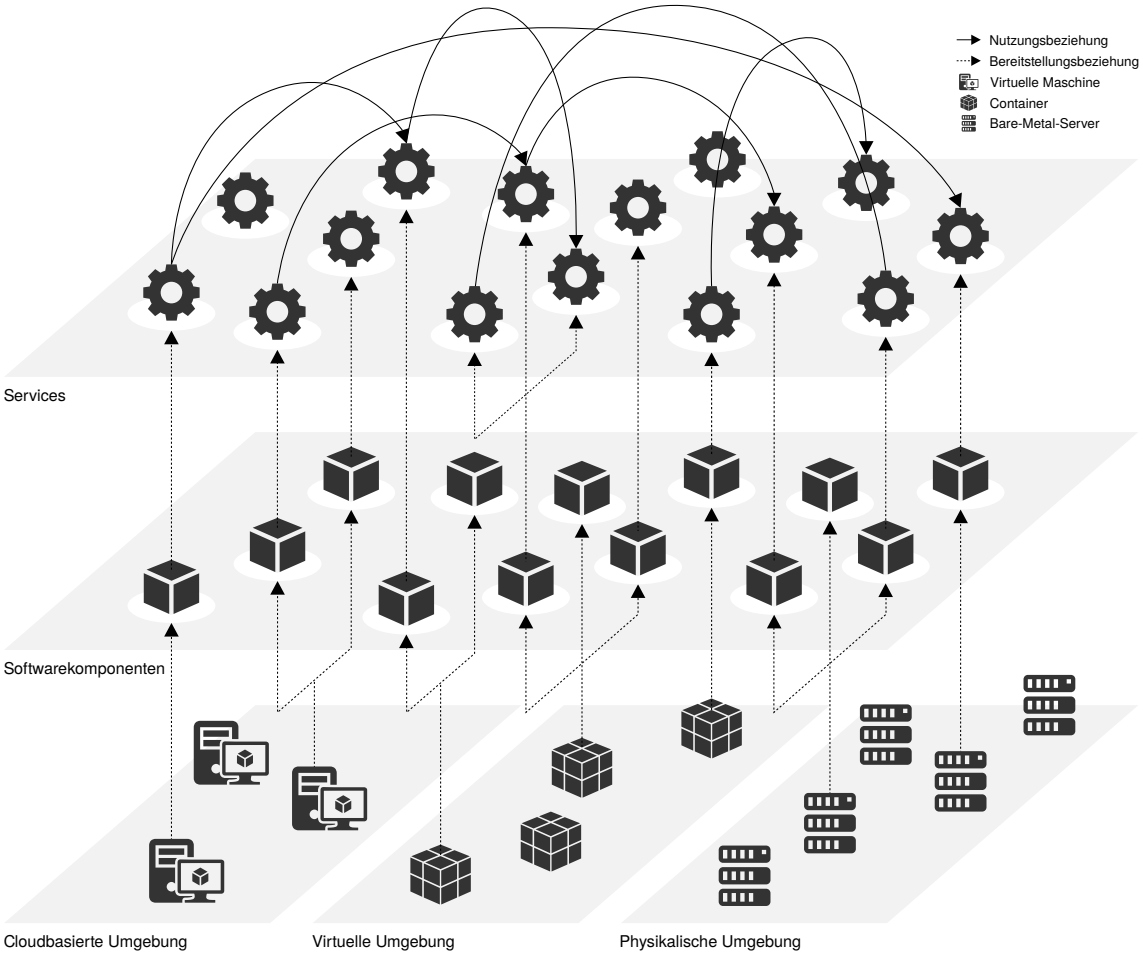


Abbildung 4.1: Aufbau eines serviceorientierten Systems in einer Multi-Provider-Umgebung

trächtigungen beeinflussen mitunter große Teile des Beziehungsgeflechts. Um Ursache und Wirkung voneinander trennen zu können, setzt die Fehlerdiagnose Managementobjekte voraus, die Zusammenhänge innerhalb der Servicebereitstellung und -nutzung erkennbar machen. Dies erfordert eine lückenlose und zugleich einheitliche Erfassung des Beziehungsgeflechts über die Grenzen der Ausführungsumgebungen hinweg.

4.1.2 Servicegüteanforderungen

Die Serviceerbringung unterliegt Qualitätsanforderungen, die zwischen Nutzer und Anbieter in Servicegütevereinbarungen vertraglich festgeschrieben sind. Services werden vom Nutzer nach Bedarf angefordert und mit nicht-funktionalen Anforderungen versehen. Aufgabe des Anbieters ist es, diese Services gütegesichert zu erbringen. Fehler und Leistungsverschlechterungen müssen zeitnah erkannt und beseitigt werden. Die Gewährleistung der Servicegüteanforderungen erfordert Maßnahmen der perfektiven und korrektiven Adaption, d. h. Anpassungen mit dem Ziel einer Leistungsoptimierung bzw. Fehlerkorrektur. Die Grundlage dafür bilden Servicegüteparameter, die innerhalb der Servicegütevereinbarungen definiert sind. Die Festlegung der Zielvorgaben erfolgt mit Hilfe von Service-Level-Objectives, deren Nichteinhaltung Vertragsstrafen nach sich ziehen können.

4.1.3 Umgebungsspezifische Managementanforderungen

Neben technologischen Unterschieden weisen Ausführungsumgebungen zudem funktionale Unterschiede auf. Physikalische Umgebungen besitzen eine geringe Dynamik, Veränderungen finden nur vereinzelt statt und erfordern zumeist manuelle Eingriffe. Systeme nehmen über einen längeren Zeitraum hinweg eine dedizierte Aufgabe wahr, größere hardware- und softwareseitige Anpassungen sind insbesondere beim Produktivbetrieb eher Ausnahme als Regel. Das technische Management setzt auf einer bestehenden Infrastruktur auf und übernimmt die Überwachung und Steuerung. Im Gegensatz dazu verfügen cloudbasierte und virtuelle Umgebungen über eine hohe Dynamik. Zu den Aufgaben des technischen Managements zählen auch die Anpassung und Optimierung der Infrastruktur. Dies umfasst die Erstellung von Host-Systemen zur Skalierung und Migration von Services.

Eigenverantwortlich betriebene und fremdbezogene Services

Bei eigenverantwortlich betriebenen Services kontrolliert das technische Management die gesamte Servicebereitstellung, angefangen bei den Host-Systemen über die Softwarekomponenten bis hin zu den Services. Bei fremdbezogenen Services sind hingegen nur die Services sichtbar. Der Anbieter stellt zur Überwachung und Steuerung Managementinformationen, insbesondere Servicegüteparameter, zur Verfügung. Der Servicebetrieb erfolgt gewöhnlich in einer cloudbasierten oder virtuellen Umgebung und untersteht dem Verantwortungsbereich des Anbieters. Die an der Serviceerbringung beteiligten Host-Systeme und Softwarekomponenten sind von außen nicht sichtbar. Zu den Aufgaben des technischen Managements zählen somit neben der Verwaltung eigenverantwortlich betriebener Services auch die fremdbezogener Services.

4.1.4 Verbund heterogener Ausführungsumgebungen

Die Schwierigkeiten im Umgang mit einer Multi-Provider-Umgebung lassen sich darauf zurückführen, dass Ausführungsumgebungen nicht vorrangig für den Einsatz innerhalb eines Verbunds ausgelegt sind. Unterschiedliche Managementschnittstellen, Datenformate und Informationsmodelle erschweren das einheitliche Management. Zudem gilt es, den Verbund flexibel an veränderte Rahmenbedingungen und Anforderungen anpassen zu können. Dafür müssen sich diesem jederzeit Ausführungsumgebungen hinzufügen und auch wieder entfernen lassen. Während noch vor wenigen Jahren Produktivsysteme zumeist in physikalischen Umgebungen betrieben wurden, kommen in letzter Zeit vermehrt cloudbasierte und virtuelle Umgebungen zum Einsatz. Lösungen folgen dem „as-a-Service“-Gedanken, also der bedarfsgesteuerten Bereitstellung und Nutzung von Infrastruktur-, Plattform- und Anwendungsservices. Um Alt- und Neusysteme bestmöglich aufeinander abzustimmen, müssen sich diese in einen gemeinsamen Managementprozess integrieren lassen.

Heterogenität

Die Gewährleistung von Interoperabilität und Portabilität im Umfeld serviceorientierter Systeme, im Speziellen innerhalb eines Verbunds heterogener Ausführungsumgebungen, ist noch Gegenstand der Forschung [VB17, NCR+18, APDM18]. Die Ursache der Heterogenität liegt in der Verteilung und Autonomie der Ausführungsumgebungen. Fehlende Standards und eine bisher nur unzureichend erfolgte Festlegung von Managementobjekten [LKBT11, Lew13] sind die Folgen. In diesem Zusammenhang kann zwischen *technischer Heterogenität* und *semantischer Heterogenität* unterschieden werden. Die technische Heterogenität lässt sich weiter in

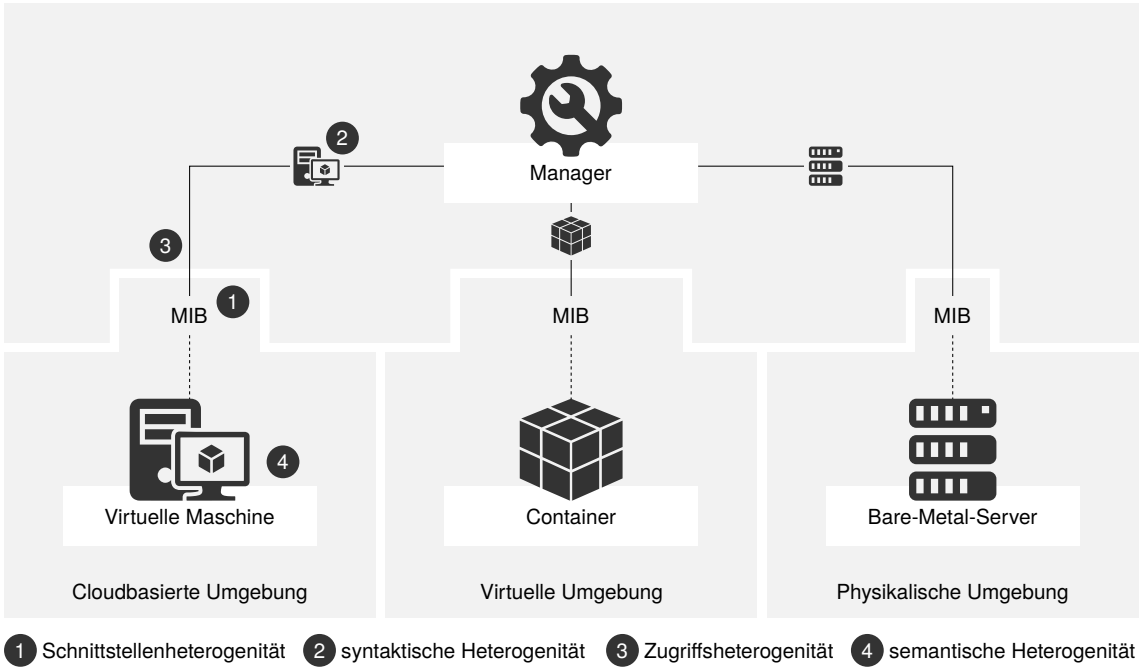


Abbildung 4.2: Heterogenität von Ausführungsumgebungen

Schnittstellenheterogenität, syntaktische Heterogenität und Zugriffsheterogenität unterteilen. Die Abbildung 4.2 veranschaulicht die unterschiedlichen Formen von Heterogenität. Sie zeigt eine Multi-Provider-Umgebung, die aus einer cloudbasierten, virtuellen und physikalischen Umgebung besteht. Jede dieser Ausführungsumgebungen verfügt über eine MIB, die Managementobjekte bereithält. Die nachfolgenden Abschnitte zeigen auf, inwiefern die technische und die semantische Heterogenität das Management einer Multi-Provider-Umgebung erschweren.

4.1.5 Managementschnittstellen

Interaktion und Ansteuerung erfolgen bei cloudbasierten und virtuellen Umgebungen über eine zentrale Managementschnittstelle, mittels derer sich Services bereitstellen, abfragen und beenden lassen. In der Regel kommt dafür eine umgebungsspezifische, nicht-standardisierte Managementschnittstelle zum Einsatz (❶ *Schnittstellenheterogenität*). Zudem besitzen Managementobjekte in Abhängigkeit zur Schnittstelle unterschiedliche Repräsentationen (❷ *syntaktische Heterogenität*). Es kommen zwar vermehrt strukturierte Beschreibungssprachen wie XML, JSON oder YAML zum Einsatz, diese lassen sich aber nicht immer verlustfrei ineinander überführen. Zudem führt die Verwendung nicht-standardisierter Objektschemata zu Unterschieden innerhalb der syntaktischen Repräsentation der Managementobjekte. So unterscheiden sich beispielsweise die Repräsentationen zweier Managementobjekte zur Beschreibung virtueller Maschinen zumeist deutlich, wenn diese in unterschiedlichen cloudbasierten Umgebungen betrieben werden.

Zugriff auf Managementobjekte

Ein Managementprotokoll regelt den Austausch von Managementinformationen, definiert Bedeutung und Aufbau von Nachrichten und legt die Kommunikationspartner fest. Es beschreibt,

wie sich die Operationen der Managementschnittstelle über ein Kommunikationsnetzwerk aufrufen lassen. Ein Managementstandard sieht typischerweise für den Zugriff auf die Managementobjekte ein eigenes Protokoll vor: CIM-XML (WBEM), SNMP oder SyncML (OMA DM). Die anbieterspezifischen Managementschnittstellen cloudbasierter und virtueller Umgebungen sind zumeist als Webservices realisiert und nutzen entweder SOAP oder REST als Nachrichtenaustauschprotokoll. Hauptproblem ist, dass innerhalb einer Multi-Provider-Umgebung neben unterschiedlichen Managementschnittstellen auch unterschiedliche Protokolle für den MIB-Zugriff zum Einsatz kommen (❸ *Zugriffsheterogenität*).

Auffinden von Managementagenten

Für den Aufbau eines internen Objektmodells müssen dem Manager zur Abfrage der Managementobjekte die Endpunkte der Managementagenten bekannt sein. In diesem Zusammenhang weisen bestehende Managementstandards, die ursprünglich für die Verwaltung physikalischer Umgebungen entwickelt wurden, Unterschiede auf. Sie verfügen teilweise über keinen Mechanismus, der das dynamische Auffinden von Managementagenten ermöglicht und überlassen die Problemlösung der Implementierung des Managementsystems. Während WBEM einen Auffindungsmechanismus auf Grundlage des *Service Location Protocol* (SLP) spezifiziert, machen weder SNMP noch OMA DM seitens des Standards Umsetzungsvorgaben. Zur Lösung des Problems kommen in der praktischen Anwendung Verzeichnisdienste sowie Netzwerkscanner auf Basis des *Internet Control Message Protocol* (ICMP) zum Einsatz.

4.1.6 Ressourcenrepräsentation durch Managementobjekte

Den MIBs liegen spezielle Informationsmodelle oder mittels nicht-standardisierter Objektschemata erweiterte Informationsmodelle zugrunde. Zudem sind sie isoliert, existieren unabhängig voneinander und besitzen keine MIB-übergreifenden Objektrelationen. Es liegt im Verantwortungsbereich der Manager, diese Beziehungen selbstständig herzustellen und innerhalb eines zusammenhängenden Objektmodells abzubilden. Für das Management serviceorientierter Systeme in einer Multi-Provider-Umgebung existiert derzeit weder ein standardisiertes noch ein allgemein akzeptiertes Informationsmodell, das die Umgebung zusammenhängend beschreibt. Folglich werden aus Managementsicht mitunter gleiche Ressourcen von unterschiedlichen Managementobjekten erfasst (❹ *semantische Heterogenität*).

Uneinheitliche Managementobjekte

Die Repräsentation von Ressourcen ist zudem von der Technologie der Ausführungsumgebung abhängig. Wie die Abbildung 4.2 zeigt, werden Host-Systeme durch uneinheitliche Managementobjekte repräsentiert. Während in der cloudbasierten Umgebung virtuelle Maschinen zum Einsatz kommen, verwendet die virtuelle Umgebung Container und die physikalische Umgebung Bare-Metal-Server. Dementsprechend liegen auch in den MIBs drei unterschiedliche Managementobjekte vor. Grundsätzlich handelt es sich aber um Ausprägungen desselben Konzepts, d. h. um Host-Systeme, die gemeinsam mit einem Betriebssystem die Laufzeitumgebung zur Ausführung von Softwarekomponenten bereitstellen. Für das einheitliche Management ist es wichtig, dass diese funktionale Übereinstimmung auch durch ein gemeinsames Managementobjekt zum Ausdruck kommt. Findet eine solche Vereinheitlichung nicht statt, operiert die Überwachungs- und Steuerungslogik direkt auf den Managementobjekten der Ausführungsum-

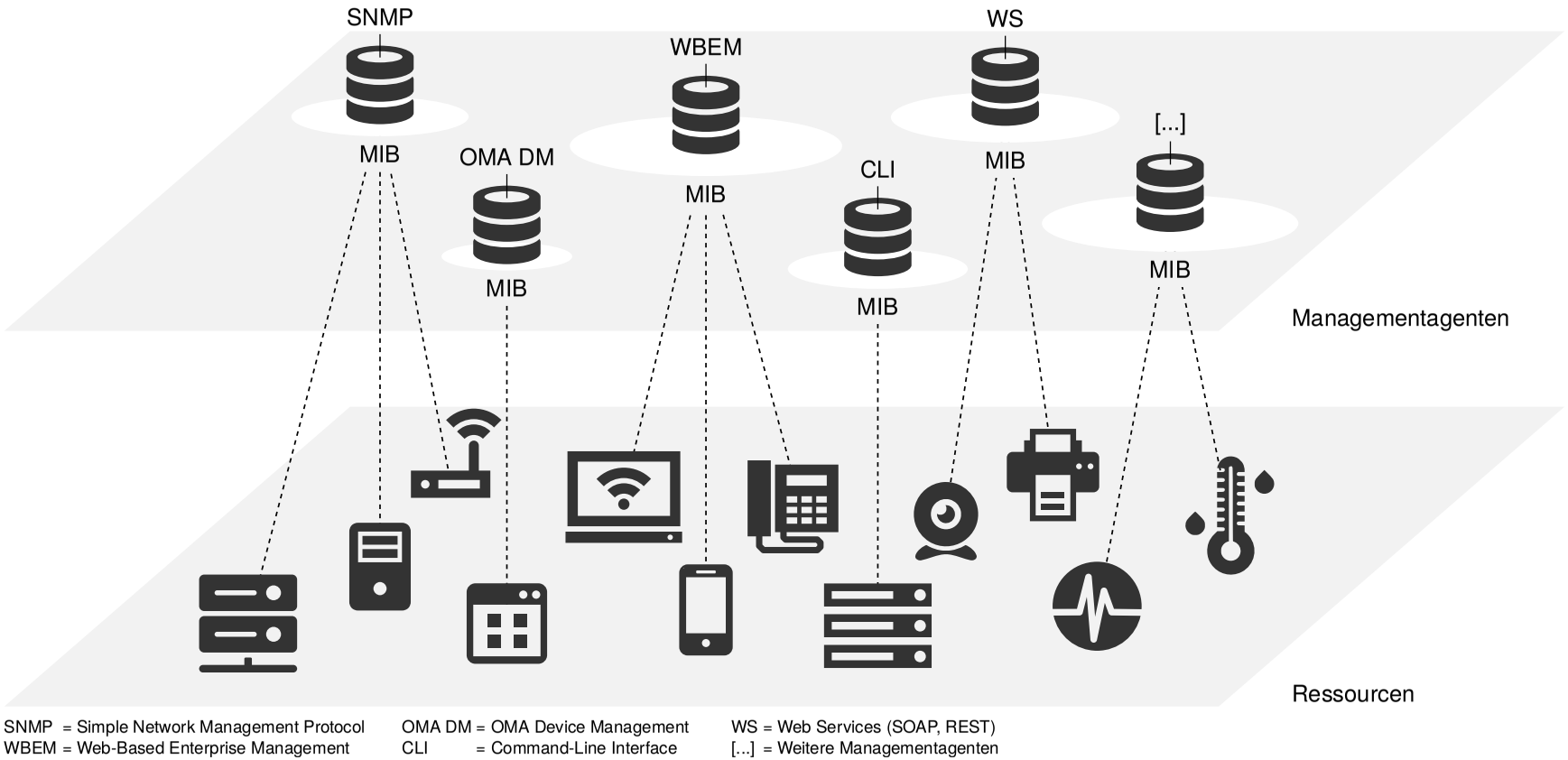


Abbildung 4.3: Unzusammenhängende Managementinformationen einer Ausführungsumgebung

gebung. Aufgrund der fehlenden Entkoppelung entstehen Abhängigkeiten, die verhindern, dass sich die Managementlogik zur Steuerung anderer Ausführungsumgebungen einsetzen lässt.

Unzusammenhängende Managementinformationen

Des Weiteren verfügt eine Ausführungsumgebung zumeist nicht über eine einzelne MIB, über die sich die Managementobjekte zentral abfragen lassen, sondern über mehrere MIBs, die jeweils Teilmengen bereitstellen. Dadurch liegen die Managementinformationen unzusammenhängend als Fragmente vor. Das Problem veranschaulicht die Abbildung 4.3. Sie zeigt zwei hierarchisch angeordnete Ebenen: die untere Ebene enthält die Ressourcen der Ausführungsumgebung, die obere Ebene die innerhalb der MIBs hinterlegten Managementobjekte. Der Zugriff erfolgt über Managementagenten, die sich wiederum in zwei Gruppen unterteilen lassen. Die eine Gruppe bietet eine zu SNMP, OMA DM oder WBEM konforme Managementschnittstelle, die andere Gruppe eine nicht-standardisierte Schnittstelle beispielsweise per Webservice oder Kommandozeile. Zur Nutzung der Managementobjekte muss der Manager die MIBs einzeln abfragen. Dies setzt Kenntnisse darüber voraus, in welchen MIBs sich die Managementinformationen befinden und über welche Managementprotokolle der Zugriff erfolgt. Gleichzeitig muss der Manager über entsprechende Implementierungen in Form von Softwarekomponenten verfügen. Dies erhöht seine Komplexität und erschwert Anpassungen und Erweiterungen.

4.2 Zielsetzung

Das Ziel der Arbeit ist eine Verbesserung und Vereinfachung des Managements serviceorientierter Systeme. Im Zentrum der Betrachtung stehen das einheitliche Management einer Multi-Provider-Umgebung und die Gewährleistung von Servicegüteanforderungen.

Einheitliches Management einer Multi-Provider-Umgebung

Hauptaufgabe ist die Identifikation der managementrelevanten Ressourcen und ihre Repräsentation durch einheitliche Managementobjekte in einem Informationsmodell. Ziel ist die Überwindung der semantischen und der technischen Heterogenität der Managementinformationen und Datenquellen. Die Ressourcen gilt es, unabhängig von der Ausführungsumgebung, dem Anbieter und dem Betriebsmodell einheitlich zu überwachen und zu steuern.

Gewährleistung von Servicegüteanforderungen

Die Managementobjekte sollen die Grundlage einer autonomen Kontrollschleife zur Selbstadaption bilden. Der schematische Aufbau ist in der Abbildung 4.4 dargestellt. Der Managementprozess durchläuft demnach wiederholt die folgenden drei Phasen:

Überwachung In der Überwachungsphase werden Managementobjekte und ihre Zustandswerte abgefragt.

Analyse und Diagnose In der Analyse- und Diagnosephase werden die Daten analysiert und mit den Vorgaben verglichen. Liegen Abweichungen vor, müssen Maßnahmen zur Korrektur ergriffen werden.

Anpassung Der ermittelte Änderungsplan kommt in der Anpassungsphase zur Ausführung.

Abbildung 4.4: Kontrollschleife zur Gewährleistung der Servicegüte

Ziel ist die Gewährleistung von Servicegüteanforderungen, die an eigenverantwortlich betriebene Services gestellt werden. Dies setzt voraus, dass sich die am Verbund beteiligten Ausführungsumgebungen zur Serviceerbringung flexibel einsetzen lassen. Es ist Aufgabe des Managementprozesses, trotz des Eintritts von Fehlern und Leistungsverschlechterungen, den gütegesicherten Servicebetrieb weitestgehend aufrechtzuerhalten. Mitunter lassen sich durch den Einsatz adaptiver Maßnahmen Servicegüteverletzungen noch rechtzeitig abwenden.

4.3 Anforderungen

Die Analyse der Ausgangssituation hat die grundsätzlichen Schwierigkeiten aufgezeigt, die es bei der Umsetzung eines einheitlichen Managements in einer Multi-Provider-Umgebung zu lösen gilt. Darauf aufbauend wird zunächst eine Anforderungsliste erstellt. Diese umfasst Anforderungen an ein Informationsmodell, das die Repräsentation serviceorientierter Systeme in einem von Heterogenität geprägten Umfeld ermöglicht.

4.3.1 Modellierung

Anforderungen an die Modellierung beziehen sich auf die Vorgehensweise und die Prinzipien, die es bei der Erstellung des Informationsmodells einzuhalten gilt. Sie legen Leitlinien fest, an denen sich der Modellierungsprozess orientieren muss. Er unterliegt folgenden Vorgaben:

Vereinheitlichung In Übereinstimmung mit der Zielsetzung sind die managementrelevanten Ressourcen durch Managementobjekte zu repräsentieren und innerhalb eines Informationsmodells festzulegen. Gefordert wird ein ausdrucksstarkes Basismodell, das bereits einen Großteil der Ressourcen erfasst.

Objektorientierung Die Definition der Managementobjekte soll mit Hilfe der objektorientierten Modellierung erfolgen. Zur Verbesserung von Kompatibilität und Anwendbarkeit

dürfen Managementobjekte über keine Operationen verfügen. Es sind ausschließlich öffentliche Eigenschaften gestattet, die sich von außen lesen oder schreiben lassen.

Serviceorientierung Services stehen im Zentrum der Modellierung und sind Gegenstand von Serviceverträgen. Ziel ist die Festlegung von Managementobjekten, die den Managementprozess dazu befähigen, die Multi-Provider-Umgebung hinsichtlich eines gütegesicherten Servicebetriebs eigenständig anzupassen und zu optimieren.

4.3.2 Modelleigenschaften

Modelleigenschaften legen allgemeine Merkmale des Informationsmodells fest. Sie gehen über die Modellierung hinaus und betreffen Anwendbarkeit und Integration in Anwendungsszenarien und Managementsysteme. Folgende Eigenschaften gilt es zu erfüllen:

Erweiterbarkeit Das Informationsmodell muss um zusätzliche Managementobjekte erweiterbar sein, die technologie- und implementierungsspezifische Elemente repräsentieren. Die Erweiterungen müssen sich in das Basismodell integrieren lassen.

Kompatibilität Grundsätzlich sollte das Informationsmodell innerhalb standardkonformer Managementsysteme (SNMP, OMA DM, WBEM) anwendbar sein. Dazu müssen sich Managementobjekte durch die von den Managementstandards festgelegten Beschreibungssprachen (SMIv2, DDF, MOF) spezifizieren lassen.

4.3.3 Managementobjekte

Das Informationsmodell bildet die Basis zur Gewährleistung von Interoperabilität, indem es die Managementobjekte vorgibt, auf denen der Überwachungs- und Steuerungsprozess operiert. Dadurch erfolgt zugleich auch eine Festlegung der Analyse-, Diagnose- und Anpassungsmöglichkeiten. Aus diesem Blickwinkel heraus gilt es, folgende Ressourcen zu modellieren:

Multi-Provider-Umgebung Die Multi-Provider-Umgebung ist durch ein Managementobjekt zu beschreiben, das die am Verbund beteiligten Ausführungsumgebungen sowie die darin enthaltenen Projekte umfasst.

Projekte Projekte stellen Strukturelemente dar und stehen in direkter Abhängigkeit zu ihrer Ausführungsumgebung. Sie umfassen eine Infrastruktur-, Plattform- und Anwendungsschicht. Diese Schichten sind zu erfassen und mit Hilfe von Managementobjekten zu beschreiben.

Servicegüte Die innerhalb der Servicegütevereinbarungen festgelegten Anforderungen in Form messbarer Zielvorgaben müssen sich durch den Managementprozess auf Einhaltung bzw. Nichteinhaltung überprüfen lassen. Dies setzt Servicegüteparameter voraus, die den Ist-Zustand erfassen und über Managementobjekte zugänglich machen.

Funktionen In der Funktionsmodellierung sind die seitens des Managements erforderlichen Funktionseinheiten zu definieren und durch Managementobjekte zu beschreiben. In diesem Zusammenhang gilt es, Zuständigkeiten und Verantwortlichkeiten sowie Interaktions- und Kommunikationsbeziehungen festzulegen.

Policys Um den Managementprozess überwachen zu können, sind die eingesetzten Policys durch Managementobjekte zu repräsentieren. Dadurch verfügt das Managementsystem über die grundlegende Fähigkeit zur Selbstüberwachung.

Domänen Mit Hilfe von Managementdomänen müssen sich Managementobjekte nach administrativen und funktionalen Gesichtspunkten gruppieren lassen. Die Domänenbildung dient dazu, unterschiedliche Managementobjekte mit Hilfe von Policys organisatorisch gleich zu behandeln.

Benutzer Im Rahmen der Benutzermodellierung sind Personen, Organisationen und Organisationseinheiten zu erfassen. Sie sind insbesondere an Serviceverträgen beteiligt, die voraussetzen, dass beide Vertragspartner eindeutig identifizierbar sind.

Berechtigungen Der Zugriff auf Managementobjekte darf ausschließlich autorisierten Benutzern möglich sein. Deshalb ist es erforderlich, ein Rechte- und Rollenkonzept festzulegen, das Zugriffsrechte auf Managementobjekte bezieht und in Verbindung mit einer Operation das Lesen, Hinzufügen, Löschen und Verändern gestattet.

5 Lösungsansatz

Das Kapitel erläutert den Lösungsansatz und die Vorgehensweise zur Vereinheitlichung des Managements einer Multi-Provider-Umgebung. Es begründet Entwurfsentscheidungen und ordnet das Vorgehen in einen größeren Zusammenhang ein. Wie bereits die Problemanalyse aufgezeigt hat, müssen mehrere Anforderungen bei der Lösungserstellung Berücksichtigung finden. Im Zentrum der Betrachtung steht ein Informationsmodell, dessen Spezifikation die Festlegung eines Modellierungsansatzes notwendig macht. Abschließend folgt die Darstellung der Herausforderungen und Neuerungen, in deren Zusammenhang zugleich der Forschungsbeitrag und die Ergebnisse der Arbeit beschrieben werden.

5.1 Entwurf

Das Management einer Multi-Provider-Umgebung erfordert den Einsatz einer MIB, die die managementrelevanten Ressourcen mit Hilfe einheitlicher Managementobjekte erfasst. Ein zusammenhängendes Geflecht von Objektbeziehungen ermöglicht die Traversierung der Managementobjekte. Die Grundlage dafür ist ein Informationsmodell, das den strukturellen Aufbau der MIB festlegt. Die nachfolgenden Abschnitte erläutern den Modellierungsansatz, die Grobstruktur des Informationsmodells und die Funktionsweise der MIB.

5.1.1 Modellierung

Die Modellierung folgt dem objektorientierten Ansatz. Hierbei werden Ressourcen durch Managementobjektklassen repräsentiert. Die Vererbung ist auf die Einfachvererbung beschränkt, so dass jede Managementobjektklasse höchstens eine Basisklasse besitzt. Eine (binäre) Assoziation beschreibt eine Beziehung zwischen zwei Managementobjektklassen. Assoziationen stellen selbst keine Managementobjektklassen dar und werden daher nicht von eigenständigen Managementobjekten erfasst. Ihre Verwaltung erfolgt über eines der beiden beteiligten Managementobjekte. Assoziationen, die über einen eigenen Zustand verfügen, sind als Managementobjektklassen zu modellieren. Die Spezifikation des Informationsmodells erfordert ein strukturiertes Vorgehen, das in drei aufeinander aufbauenden Phasen unterteilt ist (vgl. [Sai07]). Das Ergebnis einer Phase stellt die Eingabe der nachfolgenden Phase dar. Die Abbildung 5.1 ver-

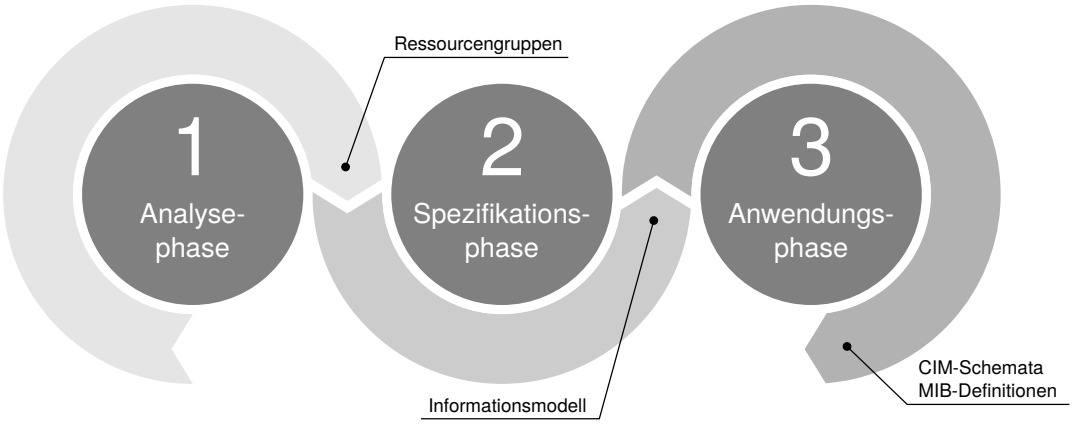


Abbildung 5.1: Dreiphasiges Vorgehen zur Gewinnung der Managementobjekte

anschaulicht dieses Vorgehen. Zur Gewinnung der Managementobjekte werden die folgenden Phasen durchlaufen:

Analysephase Die Phase dient der Ermittlung des Informationsbedarfs. Den Ausgangspunkt bilden die anforderungsgerechte Bereitstellung und der gütegesicherte Betrieb von Services. Die daran beteiligten Ressourcen müssen identifiziert, analysiert und geordnet werden. Ziel ist die Bildung von Ressourcengruppen, die sich aus Managementsicht gleichartig überwachen und steuern lassen.

Spezifikationsphase Im Rahmen der Phase erfolgt die Modellierung der Managementobjekte. Die zuvor identifizierten Ressourcengruppen werden als Managementobjektklassen repräsentiert und über Assoziationen miteinander in Beziehung gesetzt. Von besonderer Bedeutung ist die Bildung von Vererbungshierarchien, auf denen Erweiterbarkeit und Polymorphie basieren. Das Ergebnis ist ein Informationsmodell für das einheitliche Management einer Multi-Provider-Umgebung.

Anwendungsphase Die Anwendung des Informationsmodells erfordert die Spezifikation der Managementobjekte mit Hilfe einer Beschreibungssprache. Das Ergebnis variiert je nach verwendetem Managementstandard und entspricht unter anderem MIB-Definitionen oder CIM-Schemata. Diese ermöglichen die Integration in ein Managementsystem und bilden die Grundlage zur Erstellung und Erweiterung der Managementagenten.

Das dreiphasige Vorgehen kommt auch bei der Modellierung von Erweiterungen zum Einsatz. Dabei liegt grundsätzlich ein Informationsdefizit vor, d. h. managementrelevante Ressourcen, Eigenschaften oder Beziehungen werden von bestehenden Managementobjektklassen entweder gar nicht oder nur unzureichend erfasst. Diese Lücke wird in der Analysephase identifiziert und in der Spezifikationsphase durch Festlegung von Managementobjektklassen und Assoziationen geschlossen. Überführung und Integration in den Betrieb erfolgen in der abschließenden Anwendungsphase.

5.1.2 Informationsmodell

Das Informationsmodell besteht aus einem Basismodell und dessen Erweiterungen. Hauptaufgabe ist die Überwindung der semantischen Heterogenität der Managementinformationen mit Hilfe von vereinheitlichten Managementobjekten. Die nachfolgende Tabelle 5.1 veranschaulicht

den Lösungsansatz. Sie listet Begriffe aus dem vereinheitlichten Informationsmodell auf und stellt ihnen einige synonym verwendete Bezeichner gegenüber, die in den Informationsmodellen der Ausführungsumgebungen Verwendung finden:

Begriff	Synonym	Beschreibung
Node	Host Computer Server Instanz Virtuelle Maschine (VM) Container	Gerät, das mit Hilfe programmierbarer Rechenvorschriften Daten verarbeitet
Prozessor	Central-Processing-Unit (CPU) Core	Zentrale Recheneinheit in einem Computer
Arbeitsspeicher	Memory Random-Access-Memory (RAM)	Direktzugriffsdatenspeicher zur temporären Ablage von Programmen und Daten
Netzwerk-Interface	Port Network-Interface-Card (NIC) Ethernet-Adapter	Schnittstelle, die den Zugang zu einem Netzwerk ermöglicht
Block-Device	Volume Hard-Disk-Drive (HDD) Solid-State-Drive (SSD)	Gerät zur Speicherung von Datenblöcken
Netzwerk	Network Subnet Local-Area-Network (LAN) Wide-Area-Network (WAN)	Kommunikationsnetzwerk für den Austausch von Daten
…	…	…

Tabelle 5.1: Begriffe und Synonyme in den Informationsmodellen

Basismodell

Das Basismodell enthält Managementobjektklassen, die sich in unterschiedlichen Nutzungsszenarien einsetzen lassen. Dabei handelt es sich um Klassen, die zwar auf der einen Seite technologie- und implementierungsunabhängig sind, auf der anderen Seite aber ausdrucksstark genug, um managementrelevante Ressourcen zu erfassen und mit Hilfe von Eigenschaften und Beziehungen zu beschreiben. Die zentralen Elemente des Basismodells sind:

Projekte Projekte stellen logische Unterstrukturen von Ausführungsumgebungen dar. Ihnen sind Ressourcen zugeordnet, die sich der Infrastruktur-, Plattform- oder Anwendungsschicht zuordnen lassen. Die Abbildung 5.2 zeigt eine vereinfachte Projektstruktur, die den Ausgangspunkt der Modellierung bildet und jeder Ausführungsumgebung zugrunde gelegt wird. Bei eigenverantwortlich betriebenen Services sind neben den Host-Systemen auch die Applikationsstapel zu erfassen. Ein Applikationsstapel besteht aus:

- Applikationen
- Services
- Middleware

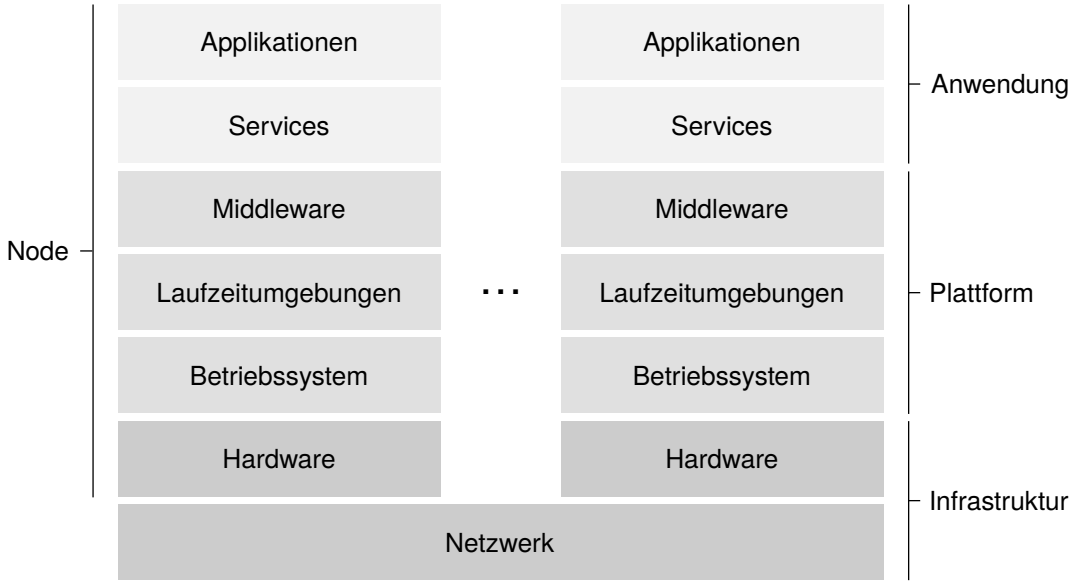


Abbildung 5.2: Vereinfachte Projektstruktur

- Laufzeitumgebungen
- Betriebssystem

Services Die Bereitstellung von Services erfolgt stets im Umfeld eines Projektes und das unabhängig davon, ob es sich um eigenverantwortlich betriebenene oder fremdbezogene Services handelt. Der Unterschied zeigt sich im Managementobjekt dahingehend, dass fremdbezogene Services keine Assoziationen zu Hardware- oder Softwarekomponenten besitzen, da die technische Bereitstellung dem Verantwortungsbereich eines anderen Managementsystems untersteht.

Erweiterungen

Erweiterungen ergänzen das Basismodell um technologie- und implementierungsspezifische Managementobjekte. Bestehende Managementobjektklassen fungieren hierbei als Basisklassen. Strikte Vererbung gewährleistet, dass sich Objekte entlang der Vererbungshierarchie polymorph behandeln lassen. Dies stellt ein Schlüsselkonzept dar, das den Einsatz der Überwachungs- und Steuerungslogik auf verfeinerten und damit auch zum Entwurfszeitpunkt unbekannten Objektklassen ermöglicht. Zu den Erweiterungen zählen insbesondere die Managementobjekte, die das technische Management, die Managementfunktionen und Policys repräsentieren.

5.1.3 Managementinformationsbasis

Die MIB einer Multi-Provider-Umgebung wird mit Hilfe einer Abstraktionsschicht realisiert, die die vereinheitlichten Managementobjekte bereitstellt. Die Abstraktionsschicht entkoppelt die Datennutzung von den technischen Details des Datenzugriffs. Die Abbildung 5.3 veranschaulicht das Prinzip und zeigt anhand eines Beispiels, wie sich die unterschiedlichen MIBs der Ausführungsumgebungen integrieren und vereinheitlichen lassen. Cloudbasierte, virtuelle und physikalische Umgebungen umfassen Managementobjekte zur Beschreibung von virtuellen Maschinen, Containern und Bare-Metal-Servern. Diese Ressourcen werden innerhalb der

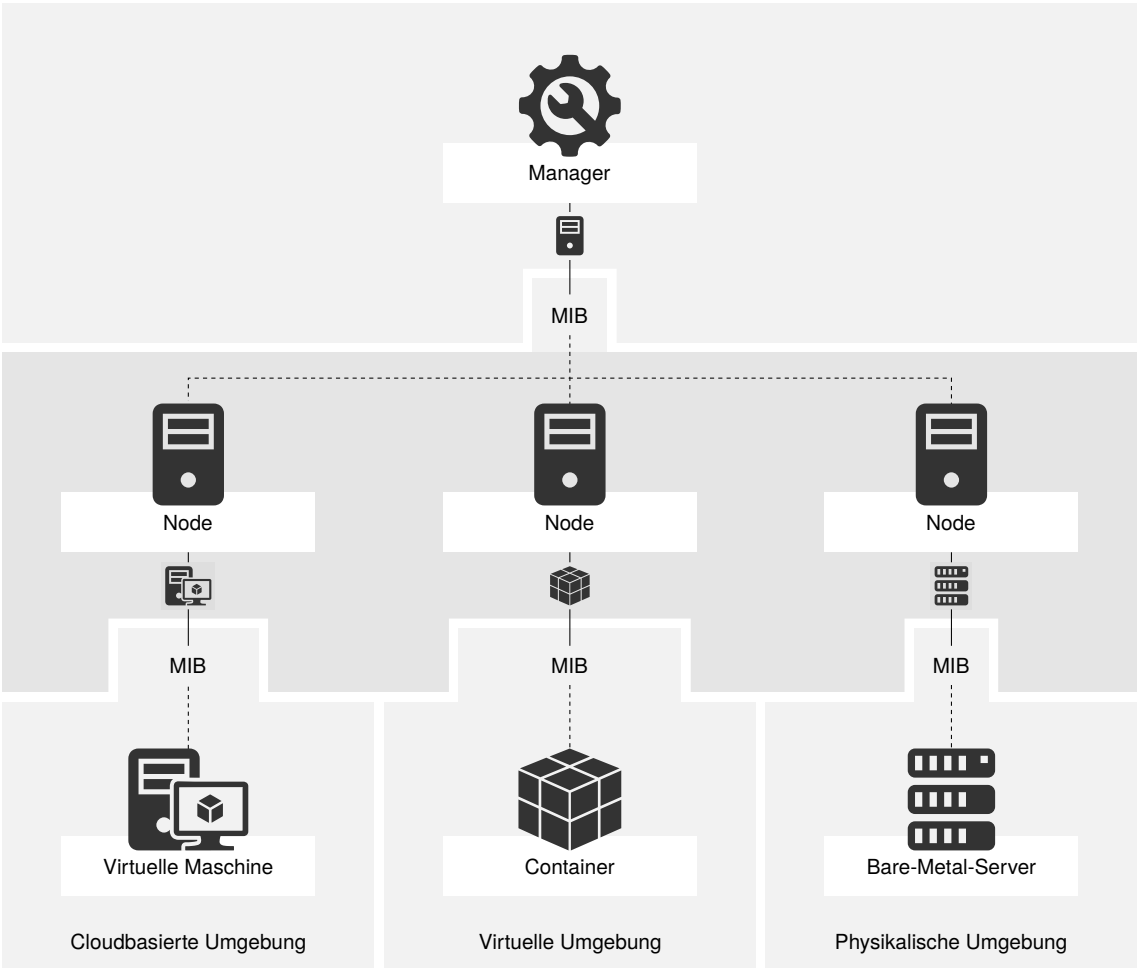


Abbildung 5.3: Abstraktionsschicht zur Vereinheitlichung der Managementobjekte

Abstraktionsschicht einheitlich als Nodes repräsentiert. Der technologische Unterschied schlägt sich lediglich in einem Eigenschaftswert des übergeordneten Managementobjekts nieder.

Informationsintegration

Grundsätzlich handelt es sich hierbei um das Problem der Informationsintegration, d. h. der Zusammenführung von Informationen heterogener Datenquellen mit dem Ziel einer einheitlichen Repräsentation mittels integrierter Sicht. Dafür existieren zwei Lösungsansätze:

Physische Integration Die Daten werden innerhalb eines Vorverarbeitungsschritts ins Zielformat transformiert und anschließend in einer separaten Datenbasis gespeichert. Diese Datenbasis stellt die integrierte Sicht dar. Der Ansatz wird auch als *materialisierte Integration* [GM99] bezeichnet.

Logische Integration Die Daten verbleiben in den Datenquellen, Anfragen werden an die jeweiligen Datenquellen durchgereicht. Die integrierte Sicht ist physisch nicht vorhanden und wird dynamisch bereitgestellt. Eine andere Bezeichnung des Ansatzes ist *virtuelle Integration* [SL90].

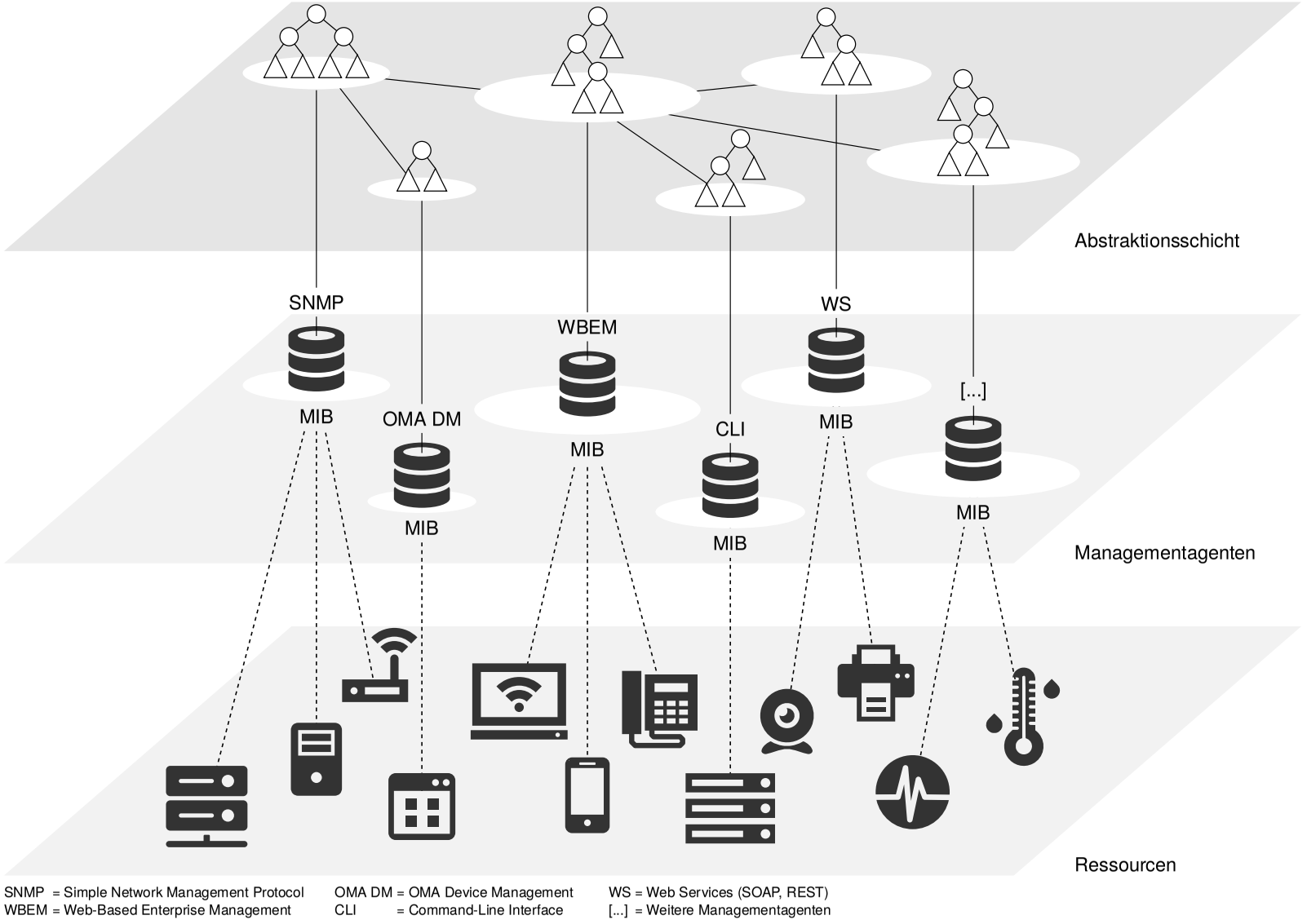


Abbildung 5.4: Zusammenführung der Managementinformationen einer Ausführungsumgebung

Zur Vermeidung einer redundanten Datenhaltung, die mit Hilfe von Pull-/Push-Mechanismen konsistent gehalten werden muss, kommt für die Abstraktionsschicht der logische Integrationsansatz zur Anwendung. Die Daten verbleiben auf den Quellsystemen, Anfragen nutzen die bereitgestellten Zugriffsschnittstellen. Die Hauptaufgabe der Abstraktionsschicht liegt in der Zugriffskoordination und der Verwaltung der *Managementobjektprovider*. Diese stellen die übergeordneten Managementobjekte zur Verfügung und verdecken Unterschiede beim Zugriff sowie der Repräsentation der untergeordneten Managementobjekte. Die Managementobjektprovider übernehmen die Rolle von Adaptern und lösen das Problem der technischen Heterogenität.

Integrierte Sicht

Die Herstellung einer integrierten Sicht auf Grundlage standardisierter und nicht-standardisierter Datenquellen ist Grundvoraussetzung für das einheitliche Management heterogener Ausführungsumgebungen. Die Abbildung 5.4 verdeutlicht diesen Zusammenhang und zeigt die Anwendung der Abstraktionsschicht zur Zusammenführung der Managementinformationen innerhalb einer Ausführungsumgebung. Durch die Bereitstellung der übergeordneten Managementobjekte lässt sich die Datenbasis um fehlende Objektbeziehungen ergänzen, die ein MIB-übergreifendes Management bisher verhindert haben. Dadurch kann die Ausführungsumgebung durch ein zusammenhängendes Managementobjekt beschrieben und mit dem Verbundobjekt assoziiert werden. Der Zugriff auf die Datenbasen erfolgt für den Manager transparent, denn er greift ausschließlich auf das übergeordnete Managementobjekt zu. Die Weiterleitung der Anfragen an die Datenquelle bzw. die Ermittlung der zur Bearbeitung verantwortlichen Managementobjektprovider obliegt dem Aufgabenbereich der Abstraktionsschicht.

5.2 Herausforderungen

Während sich die technische Heterogenität durch den Einsatz einer Abstraktionsschicht und demnach gänzlich implementierungsseitig lösen lässt, ist die Überwindung der semantischen Heterogenität ein größeres Hindernis. Zwar stellt die Vereinheitlichung der Managementobjekte eine vielversprechende Lösungsstrategie dar, ihre Umsetzung gestaltet sich aber ungleich schwieriger. Im Speziellen gelten folgende Problembereiche als Herausforderung:

Modellgranularität

Eine Herausforderung stellt die Festlegung der Modellgranularität dar. Es gilt, einen Kompromiss zwischen Ausdrucksstärke und Vereinfachung zu finden. Umfassen Managementobjekte eine zu große Gruppe von Ressourcen, sind Überwachung und Steuerung nur auf einem abstrakten Niveau durchführbar. Zustandsinformationen sind wenig aussagekräftig, und Beeinflussungsmöglichkeiten unterliegen starken Einschränkungen. Ist der Detailgrad der Modellierung hingegen zu fein gewählt, stehen dem Management zwar eine Vielzahl von Zustandsinformationen und Beeinflussungsmöglichkeiten zur Verfügung, die Komplexität des Informationsmodells macht es dann aber schwer verständlich und erweiterbar. Strukturen und Abhängigkeiten lassen sich nicht mehr intuitiv erfassen, wodurch Akzeptanz und Einsatzbereitschaft deutlich herabgesetzt sind. Zudem wird der Entwurf der Überwachungs- und Steuerungslogik erschwert. Planungs- und Umsetzungsfehler sind die Folge, die sich negativ auf die Robustheit und Stabilität des Managementsystems auswirken.

Abstraktionsniveau

Des Weiteren gilt es, Managementobjekte festzulegen, die allgemeine Konzepte repräsentieren und die sich sowohl in cloudbasierten wie auch virtuellen und physikalischen Umgebungen wiederfinden lassen. Abstraktion zielt darauf ab, unter Detailverzicht diese allgemeinen Konzepte zu identifizieren. Das Abstraktionsniveau legt fest, welche Details als allgemeingültig, spezifisch oder vernachlässigbar einzustufen sind. Dabei besteht ein direkter Zusammenhang zwischen der Modellgranularität und dem Abstraktionsniveau. Während die Modellgranularität festlegt, welche Ressourcen Teil des Informationsmodells werden, bestimmt das Abstraktionsniveau den Feinheitsgrad der Repräsentation.

Adaptierbarkeit

Eine besondere Herausforderung stellt die Adaptierbarkeit einer Multi-Provider-Umgebung dar. Diese Eigenschaft bildet die Grundlage für ein adaptives Systemverhalten. Die Überwachungs- und Steuerungslogik setzt seitens des Informationsmodells entsprechende Managementobjekte voraus, deren Rekonfiguration eine Anpassung der Multi-Provider-Umgebung bewirken. Es gilt, die Infrastruktur-, Plattform- und Anwendungsschicht gleichermaßen beeinflussen zu können. Dabei ist zu berücksichtigten, dass Managementobjekte ausschließlich erzeugt, entfernt, ausgelesen und verändert werden können, weitere Operationen stehen nicht zur Verfügung. Auf Grundlage dieser Basisoperationen muss sich die Adaptierbarkeit umsetzen lassen.

5.3 Neuerungen

Vor dieser Arbeit existierte kein Informationsmodell einer Multi-Provider-Umgebung, das die managementrelevanten Ressourcen der Ausführungsumgebungen durch einheitliche Managementobjekte repräsentiert und das im Hinblick auf die Gewährleistung von Servicegüteanforderungen sowie die Definition einer MIB. Die Vereinheitlichung heterogener Ausführungsumgebungen in Form cloudbasierter, virtueller und physikalischer Umgebungen mit dem Ziel einer vereinfachten Überwachung und Steuerung wurde bisher nur unzureichend behandelt. Insbesondere fehlte es an einer Vorgehensweise, die einen gütegesicherten Betrieb von eigenverantwortlich erbrachten Services in einer Multi-Provider-Umgebung anbieter- und umgebungsübergreifend ermöglicht.

Abgrenzung

Zwar existiert mit [Maj10] ein erster Beitrag zur Spezifikation eines semantischen Informationsmodells für serviceorientierte Systeme. Die Arbeit verfolgt aber einen ontologiegetriebenen Ansatz und befasst sich nicht mit der Vereinheitlichung von Managementobjekten zur Überwachung und Steuerung serviceorientierter Systeme in einer Multi-Provider-Umgebung. Die in [Sai07] entwickelte Service-MIB konzentriert sich auf die Berechnung von Qualitätsmaßen aus Komponentenparametern sowie der Repräsentation von Servicegütevereinbarungen. Die in [Deb05, Sch08, Liu11] entwickelten Ansätze definieren kein Informationsmodell, sondern legen den Schwerpunkt auf das Servicegütemanagement. Standardisierungsvorhaben wie OVF [DMT14a], OCCI [Ope16a] und CIMI [DMT16] konzentrieren sich auf das Problem der technischen Heterogenität innerhalb der Infrastrukturschicht, während Plattform- und Anwendungsschicht größtenteils unberücksichtigt bleiben. Mit TOSCA [OAS13] und CAMP [OAS14]

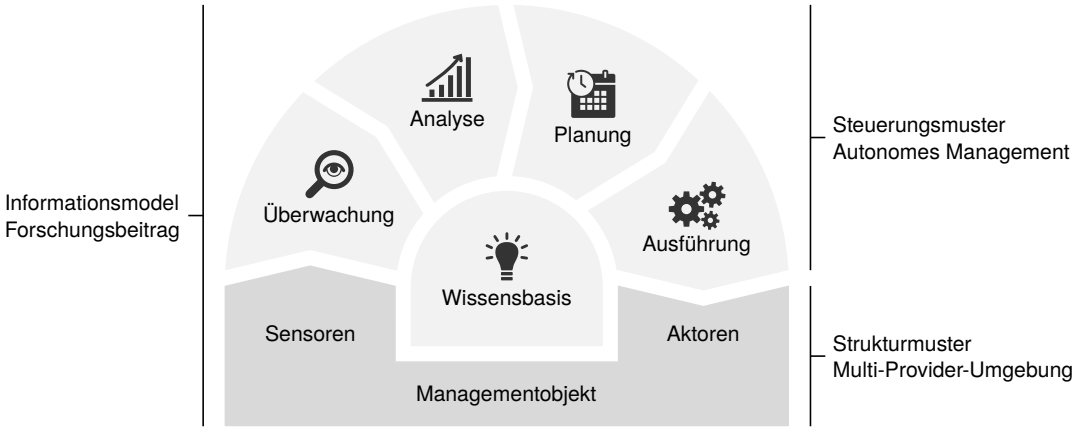


Abbildung 5.5: Einordnung des Forschungsbeitrags

existieren zwei Standards zur Vereinheitlichung der Anwendungsbereitstellung innerhalb cloudbasierter und virtueller Umgebungen, sie befassen sich aber lediglich mit den technischen Aspekten der initialen Provisionierung.

Forschungsbeitrag und Ergebnisse

Die von IBM initiierte Autonomic-Computing-Initiative liefert hinsichtlich der Entwicklung adaptiver Systeme wichtige Erkenntnisse und Beiträge, darunter das Muster des autonomen Elements. Sein Einsatz zur automatisierten Überwachung und Steuerung einer Multi-Provider-Umgebung erscheint vielversprechend. Voraussetzung dafür ist, dass die Ressource durch ein einzelnes Managementobjekt repräsentiert ist und über eine entsprechende Sensorik und Aktorik verfügt. Diese Forderung erweist sich als problematisch, da eine Multi-Provider-Umgebung bisher nicht als Managementobjekt aufgefasst wurde. Dementsprechend existieren keine Beiträge zu dessen Definition und den darauf aufsetzenden Steuerungsmustern. Es ist Aufgabe der vorliegenden Arbeit, diese Lücke zu schließen. Die Abbildung 5.5 ordnet den Forschungsbeitrag ein und zeigt, dass das Informationsmodell, im Speziellen die Repräsentation eines Umgebungsverbunds durch ein Managementobjekt, den ersten Schritt in Richtung eines adaptiven Managements darstellt. Im Rahmen der Arbeit wurden folgende Ergebnisse erzielt:

- Spezifikation eines Informationsmodells für das einheitliche Management einer Multi-Provider-Umgebung
- Repräsentation einer Multi-Provider-Umgebung als zusammenhängendes Managementobjekt
- Überwindung der semantischen Heterogenität der Managementinformationen mittels Informationsmodell
- Überwindung der technischen Heterogenität der Datenquellen mittels Abstraktionsschicht
- Definition von Struktur- und Steuerungsmustern zur Überwachung und Steuerung
- Festlegung eines funktionalen Rollenmodells für das Projektmanagement
- Vorgehensweise für die anforderungsgerechte Bereitstellung und den gütegesicherten Betrieb nicht-interaktiver Services in einer Multi-Provider-Umgebung

6

Informationsmodell

Die Vereinheitlichung der Managementobjekte bildet die Lösungsstrategie zur Überwindung der semantischen Heterogenität beim Management einer Multi-Provider-Umgebung. Ziel ist die Spezifikation eines Informationsmodells, das die managementrelevanten Ressourcen durch Managementobjekte repräsentiert und Beziehungen durch Assoziationen beschreibt. Im Vordergrund steht die Serviceerbringung unter Qualitätsanforderungen. Hierbei hängen die qualitativen Eigenschaften eines Services vom reibungslosen Zusammenwirken aller beteiligten Funktionen und Komponenten ab. Zur Anpassung an das jeweilige Anwendungsszenario müssen implementierungs- und technologiespezifische Erweiterungen möglich sein. Zunächst wird die Grundstruktur des Informationsmodells entworfen. Darauf aufbauend erfolgt die Umgebungs-, Projekt-, Servicegüte-, Funktions-, Policy-, Domänen-, Benutzer- und Berechtigungsmodellierung. Abschließend werden Struktur- und Steuerungsmuster für das Management eines Umgebungsverbunds festgelegt.

6.1 Grundstruktur

Die Grundstruktur des Informationsmodells legt Anordnung und Aufbau der Managementobjektklassen fest und gibt die Rahmenbedingungen der Modellierung vor. Die Spezifikation erfolgt mit Hilfe der grafischen Beschreibungssprache UML. Die Abbildung 6.1 zeigt ein UML-Profil, das die Klassen *Classifier* und *Property* des UML-Kernels [RJB04] verwendet und die beiden folgenden Stereotypen als zentrale Bausteine des Informationsmodells vorsieht:

Managementobjektklasse Managementobjektklassen dienen als Baupläne für Managementobjekte und beschreiben Gruppen von Ressourcen, die aus Managementsicht über eine ähnliche Struktur und ein ähnliches Verhalten verfügen. Dabei kann es sich sowohl um physikalische wie auch virtuelle und logische Ressourcen handeln. Managementobjektklassen können Assoziationen zu anderen Managementobjektklassen besitzen. In diesem Zusammenhang begrenzt die Multiplizität die minimale und maximale Anzahl der Managementobjekte, mit denen ein Objekt der Managementobjektklasse in Beziehung stehen kann.

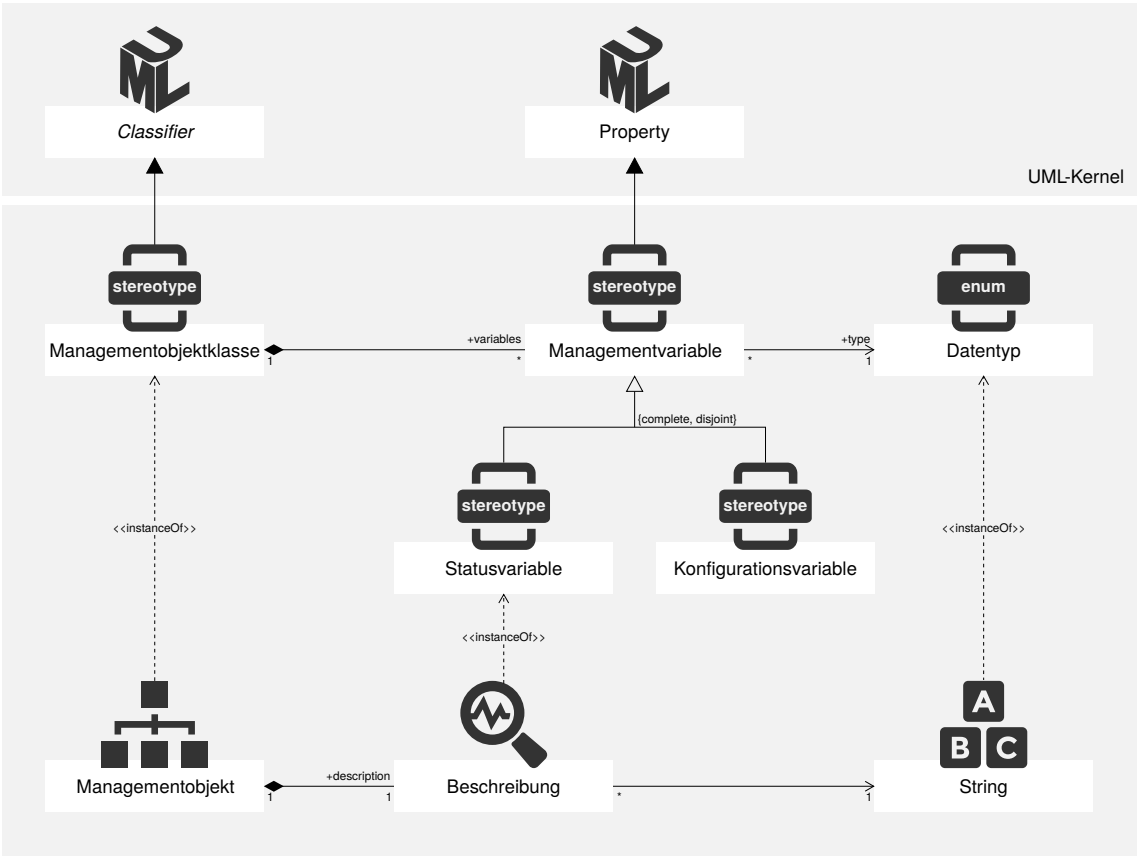


Abbildung 6.1: Grundstruktur des Informationsmodells

Managementvariable Managementobjektklassen verfügen über öffentliche Eigenschaften, die in Form von Managementvariablen modelliert werden. Sie repräsentieren Zustands- und Konfigurationswerte. Managementvariablen gewähren Zugriff auf Einzelwerte eines elementaren Datentyps und sind streng typisiert. Sie stellen die zu Überwachungs- und Steuerungszwecken festgelegten Managementinformationen bereit. Managementvariablen dienen der Zugriffsabstraktion und ermöglichen, dass sich Zustands- und Konfigurationswerte von Ressourcen einheitlich verwenden lassen, ohne dass dem Manager die technischen Details des Datenzugriffs bekannt sein müssen. Abhängig von ihrem Einsatzzweck lassen sich Managementvariablen in die beiden folgenden Gruppen unterteilen:

Statusvariablen Mit Hilfe von Statusvariablen kann sich der Manager über den Ressourcenzustand informieren. Es besteht ausschließlich lesender Zugriff. Das Anlegen, Löschen oder Verändern ist seitens des Managers nicht möglich. Es unterliegt der Verantwortung des Managementagenten, Statusvariablen anzulegen und ihre Wertebelegungen an den Ressourcenzustand anzupassen. Statusvariablen bilden die Grundlage der Überwachung und stellen die Datenbasis für das Treffen von Steuerungsentscheidungen bereit.

Konfigurationsvariablen Mittels Konfigurationsvariablen kann der Manager auf den Ressourcenzustand einwirken. Das Anlegen, Löschen, Auslesen und Verändern ist seitens des Managers möglich. Ziel ist es, eine Ressource dazu zu veranlassen, einen

Zustandsübergang durchzuführen. Betrifft eine Konfiguration mehrere Konfigurationsvariablen gleichzeitig, müssen diese in Einzelschritten mit Werten belegt werden.

Die Generalisierungsgruppe ist vollständig und überlappungsfrei, d. h. Erweiterungen sind unzulässig und Managementvariablen entsprechen entweder Status- oder Konfigurationsvariablen. Zwischen Managementvariablen und Managementobjektklasse besteht eine Kompositionsbeziehung, so dass Managementvariablen über keinen modellweit eindeutigen Bezeichner verfügen müssen. Es gilt lediglich zu gewährleisten, dass keine zwei Managementvariablen mit gleicher Bezeichnung existieren, die derselben Managementobjektklasse zugeordnet sind. Die Multiplizitätsangabe definiert eine Managementvariable als optional oder obligatorisch. Optionale Managementvariablen lassen sich dynamisch hinzufügen und entfernen, obligatorische Managementvariablen existieren hingegen über die gesamte Lebenszeit des Managementobjekts hinweg.

6.1.1 Wurzelentität

Managementobjekte sind Ausprägungen von Managementobjektklassen, die mittels des gleichnamigen Stereotyps modelliert werden. Das Informationsmodell ist hierarchisch strukturiert, so dass die Spezifikation einer Managementobjektklasse zugleich auch ihre Einordnung in eine Vererbungshierarchie erfordert. Die Wurzel bildet dabei stets die folgende Klasse:

Managamentobjekt Die Klasse stellt die Basisklasse aller Managementobjekte dar und fungiert als zentrale Wurzelentität des Informationsmodells. Sie bildet grundlegende Eigenschaften in Form von Statusvariablen ab, die an abgeleitete Managementobjektklassen weiter vererbt werden. Zu den Zustandsgrößen zählen:

- Identifikator
- Name
- Beschreibung
- Version

Die Festlegung einer zentralen Wurzelentität sorgt dafür, dass Erweiterungen vom Basismodell abhängig sind und die Gesamtheit der Managementobjekte stets eine zusammenhängende Struktur bildet. Aus Gründen der Übersichtlichkeit wird in den nachfolgenden Abbildungen auf die Repräsentation dieser Vererbungsbeziehung verzichtet. Es gilt per Konvention, dass jede Managementobjektklasse, die in keine Vererbungshierarchie eingeordnet ist, die Klasse *Managementobjekt* als direkte Oberklasse besitzt.

6.1.2 Referenzpunkte

Erweiterungen sind sogenannte *Referenzpunkte* zugeordnet. Dabei handelt es sich um Managementobjektklassen des Basismodells oder anderer Erweiterungen, die als direkte Oberklassen der modellierten Managementobjektklassen fungieren und ihren Ursprung außerhalb der eigenen Erweiterung haben. Referenzpunkte unterstehen vielfach einer anderen Zuständigkeit und können Änderungen unterworfen sein, die sich auf die eigenen Managementobjektklassen auswirken. Grundsätzlich kann jede Managementobjektklasse als Referenzpunkt fungieren, es sei denn eine Generalisierungsgruppe wird als vollständig definiert. Diese Beschränkung unterbindet jedwede Erweiterung. In einem solchen Fall sollte eine über- bzw. untergeordnete

Managementobjektklasse als Oberklasse verwendet werden. Dabei muss die Vererbungshierarchie so lange von oben nach unten durchlaufen werden, bis eine Klasse gefunden wird, die nicht Teil einer abgeschlossenen Generalisierungsgruppe ist.

6.2 Multi-Provider-Umgebung

Mit Hilfe der Grundstruktur gilt es, eine Multi-Provider-Umgebung durch ein zusammenhängendes Managementobjekt zu beschreiben. Dies setzt die Spezifikation einer Managementobjektklasse voraus. Auf Grundlage der in Abschnitt 2.4 erfolgten Klassifikation von Ausführungsumgebungen entspricht eine Multi-Provider-Umgebung einem Umgebungsverbund, der aus unterschiedlichen Ausführungsumgebungen verschiedener Anbieter und Technologien besteht. Die sich daraus ergebende hierarchische Struktur lässt sich mit Hilfe des Kompositionsmusters [GHJV94] erfassen. Die nachfolgende Abbildung 6.2 zeigt die dazugehörigen Managementobjektklassen. Es handelt sich um:

Ausführungsumgebung Eine Ausführungsumgebung stellt eine physikalische, virtuelle oder cloudbasierte Umgebung dar. Indem diese drei Umgebungen jeweils durch dieselbe Managementobjektklasse repräsentiert werden, lässt sich die semantische Heterogenität der Managementinformationen überwinden.

Umgebungsverbund Ein Umgebungsverbund entspricht einem losen Verbund von Ausführungsumgebungen.

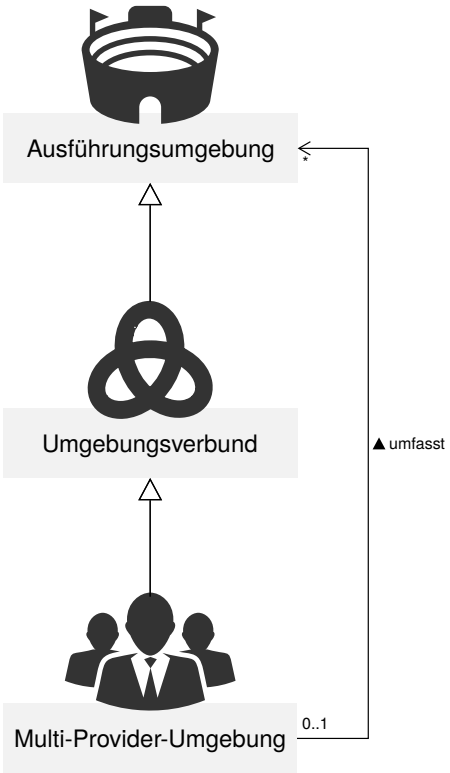


Abbildung 6.2: Managementobjekte zur Repräsentation einer Multi-Provider-Umgebung

Multi-Provider-Umgebung Die Multi-Provider-Umgebung ist eine zusammengesetzte Ausführungsumgebung, die beliebig viele andere Ausführungsumgebungen umfassen kann. Eine Ausführungsumgebung verfügt über einen eigenen Lebenszyklus und existiert unabhängig von einem Umgebungsverbund.

6.3 Projekte

Aus Sicht des technischen Managements ist eine cloudbasierte, virtuelle und physikalische Umgebung in Projekte unterteilt. Diese können jeweils über eine vollständig ausgeprägte Infrastruktur-, Plattform- und Anwendungsschicht verfügen. Ein *Projekt*, auch *Tenant* genannt, entspricht einer Organisationseinheit zur anbieterseitigen Verwaltung und Bereitstellung von Services. Projekte sind vollständig gegeneinander gekapselt und verfügen über Absicherungen, die eine unautorisierte Informationsgewinnung verhindern sollen. Dafür kommen Mechanismen zur Identitäts- und Zugriffskontrolle zum Einsatz, die dafür sorgen, dass ausschließlich authentifizierte und autorisierte Anwender Kontrolle über Projekte ausüben können. Damit Anwender nicht unberechtigt an Informationen anderer gelangen oder Zugriff auf Ressourcen und Services außerhalb der eigenen Zuständigkeit erhalten, ist eine durchgängige Trennung entlang der Infrastruktur-, Plattform- und Anwendungsschicht erforderlich.

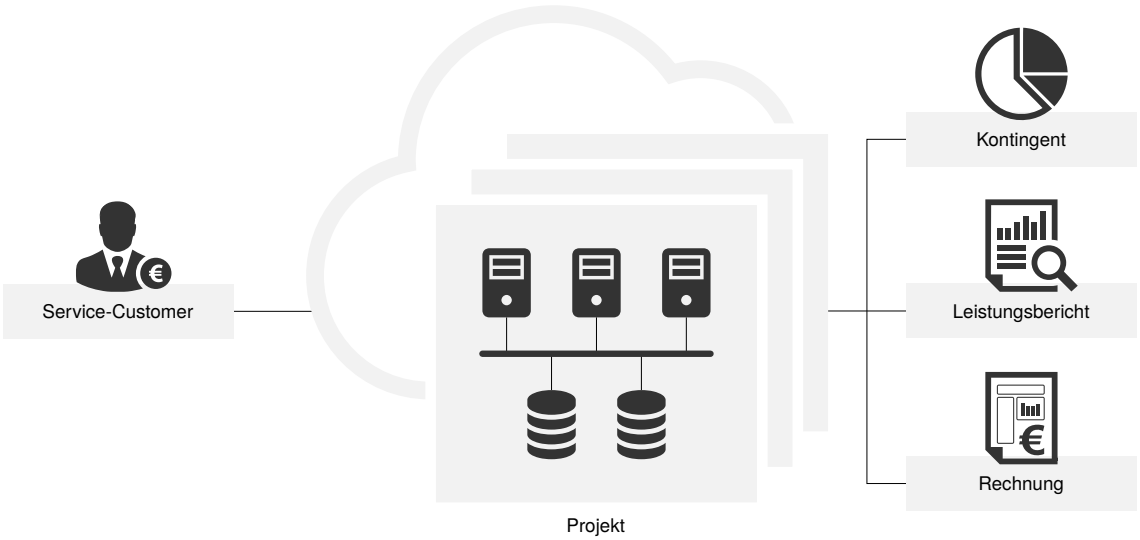
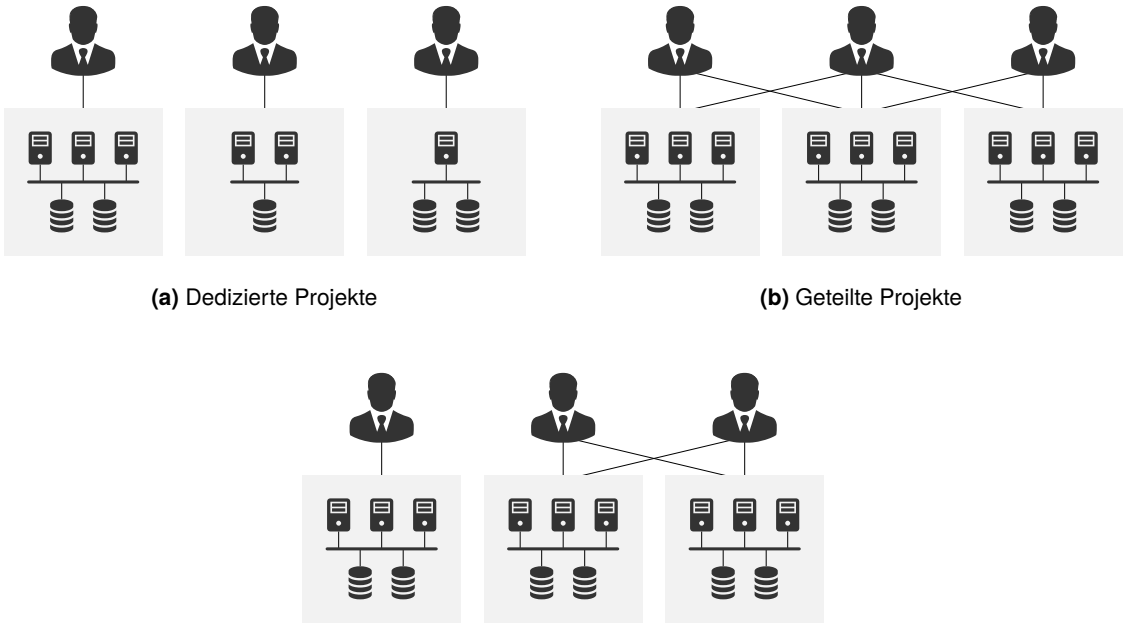


Abbildung 6.3: Organisatorische Struktur einer Ausführungsumgebung

Die Abbildung 6.3 zeigt die organisatorische Struktur einer Ausführungsumgebung. Services werden stets im Rahmen von Projekten erbracht, wobei Umfang und Menge der nutzbaren Ressourcen durch ein *Kontingent* (engl. *Quota*) begrenzt sein können. Sind vertragliche Zusicherungen bzgl. der Leistungserbringung gemacht worden, ist der Anbieter dazu verpflichtet, einen *Leistungsbericht* (engl. *Report*) zu erstellen. Dieser dient als Nachweis dafür, dass alle qualitativen Anforderungen eingehalten wurden. Nach Ablauf der Abrechnungsperiode wird eine *Rechnung* (engl. *Invoice*) erstellt, die die in Anspruch genommenen Ressourcen und Services umfasst. Die Kostenberechnung erfolgt in Übereinstimmung mit dem vereinbarten Kostenmodell. Die Zuordnung zwischen Service-Customer und Projekt ist nicht festgelegt. Es lassen sich drei unterschiedliche Organisationsformen unterscheiden. Diese sind in der Abbildung 6.4 dargestellt. Es handelt sich um:



(a) Dedizierte Projekte

(b) Geteilte Projekte

(c) Gemischte Projekte

Abbildung 6.4: Organisationsformen von Projekten

Dedizierte Projekte Ein *dediziertes Projekt* steht zur Nutzung exklusiv einem einzigen Service-Customer zur Verfügung. Alle Ressourcen und Services, die innerhalb eines dedizierten Projekts bereitgestellt werden, dürfen ausschließlich von ihm verwendet werden. Eine wirksame Trennung zwischen den Anwendern ist somit bereits auf organisatorischer Ebene gewährleistet und bedarf keiner zusätzlichen Maßnahmen. Der Einsatz dedizierter Projekte vereinfacht die Abrechnung, da die Bereitstellung von Ressourcen und Services stets innerhalb desselben Projekts erfolgt. Dadurch lassen sich die entstandenen Kosten direkt einem Verursacher zuordnen.

Geteilte Projekte Innerhalb eines *geteilten Projekts* werden Ressourcen und Services für unterschiedliche Service-Customer erbracht. Die Trennung zwischen den Anwendern ist deutlich komplexer und erfordert zusätzliche Maßnahmen auf der Infrastruktur-, Plattform- und Anwendungsschicht. Des Weiteren sind genaue Kenntnisse darüber erforderlich, ob und in welcher Form sich Services untereinander negativ beeinflussen, die denselben Ressourcen zugeordnet sind. Es hängt vom Anwendungsfall bzw. den daran beteiligten Softwarekomponenten ab, ob sich dieselben Ressourcen und Services von mehreren Anwendern gleichzeitig nutzen lassen oder ab welcher Schicht dedizierte Services zur wirksamen Isolierung erforderlich sind.

Gemischte Projekte Bei *gemischten Projekten* kommen dedizierte und geteilte Projekte gleichzeitig zum Einsatz. Diese Organisationsform stellt einen Kompromiss zwischen ressourceneffizienter Auslastung, geringeren Kosten und vermindertem Verwaltungsaufwand dar. Statt eine Applikation wiederholt in jedem dedizierten Projekt aufsetzen zu müssen, mag es sinnvoll erscheinen, ein geteiltes Projekt zu erstellen, das eine Anbindung unter Einsatz von Hochverfügbarkeitstechniken mehreren Service-Customern zur Verfügung stellt. Insbesondere dann, wenn Lizenzgebühren anfallen oder die Applikation nicht

direkt zugängig sein soll, bieten sich Bereitstellung und Betrieb in einem geteilten Projekt an.

Die Repräsentation eines Projekts durch ein Managementobjekt ist von zentraler Bedeutung für das einheitliche Management heterogener Ausführungsumgebungen. Dazu müssen managementrelevante Ressourcen identifiziert und der Infrastruktur-, Plattform- oder Anwendungsschicht zugeordnet werden. Hierbei gilt es besonders, Zusammenhänge zwischen den Schichten herzustellen. Die nachfolgende Abbildung 6.5 zeigt die dazugehörigen Managementobjekte und veranschaulicht den Zusammenhang zwischen Projekt und Ausführungsumgebung:

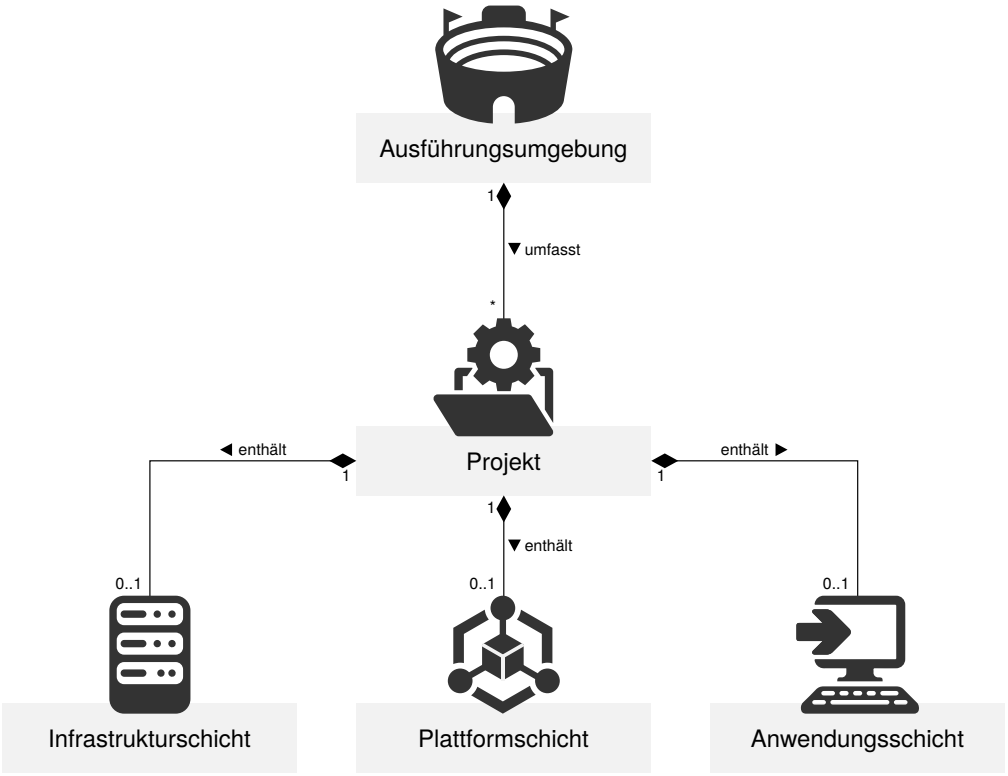
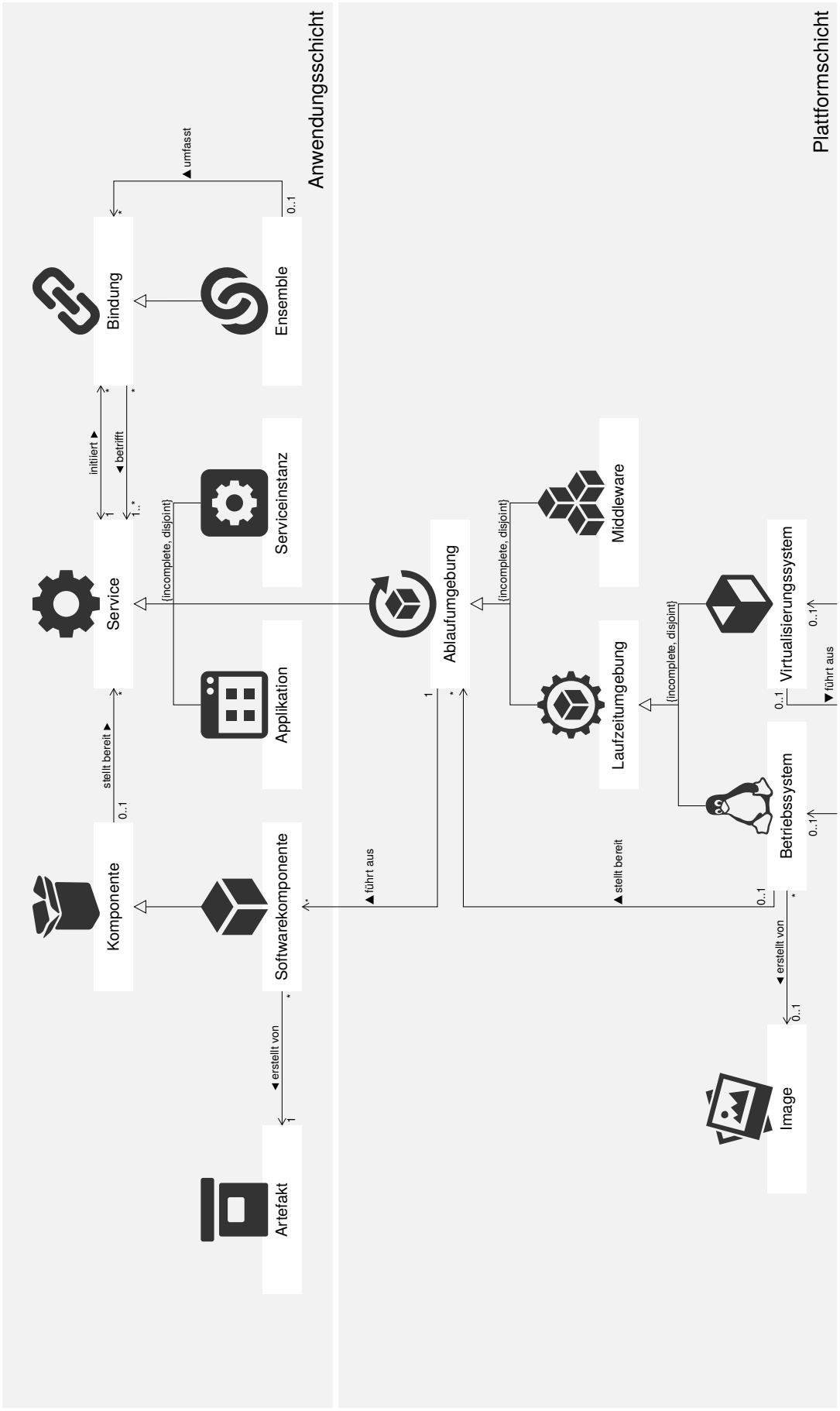


Abbildung 6.5: Managementobjekte zur Repräsentation von Projekten

Demnach umfasst eine Ausführungsumgebung beliebig viele Projekte, die ihrerseits genau einer Ausführungsumgebung zugeordnet sind. Projekte verfügen über keinen eigenen Lebenszyklus und existieren ausschließlich im Kontext einer Ausführungsumgebung. Ein Projekt besteht aus einer Infrastruktur-, Plattform- und Anwendungsschicht, die wiederum vom Projekt abhängig sind. Je nach Gegebenheiten und Erfordernissen ist es möglich, auf die Repräsentation einer oder mehrerer Schichten zu verzichten. Dadurch lassen sich Anwendungsszenarien erfassen, in denen ausschließlich Applikationen und Services zum Einsatz kommen, die fremdbezogen werden und für die keinerlei Informationen über Bereitstellung und Betrieb vorliegen. Dieses noch sehr neue IT-Betriebskonzept wird als *serverloses Computing* oder *serverlose Architektur* [BCC+17, MB17] bezeichnet. Dabei treten für den Entwickler die Ressourcen der Infrastruktur- und Plattformschicht in den Hintergrund. Der Schwerpunkt liegt auf der Erstellung und Einbettung der fachlichen Funktionen in die Anwendungsschicht. Die Abbildung 6.6 zeigt die Managementobjekte der drei Schichten, die in den nachfolgenden Abschnitten erläutert werden. Die Grundlage bildet das in Abschnitt 2.4.3.3 vorgestellte Servicemodell des


Anwendungsschicht
Plattformschicht
Bindung
Ensemble
umfasst
0..1
*
initiiert
betrifft
1
1..*
Service
Serviceinstanz
{incomplete, disjoint}
stellt bereit
0..1
Komponente
Softwarekomponente
Applikation
Artefakt
erstellt von
führt aus
Ablaufumgebung
Middleware
Laufzeitumgebung
stellt bereit
Virtualisierungssystem
führt aus
Betriebssystem
erstellt von
Image

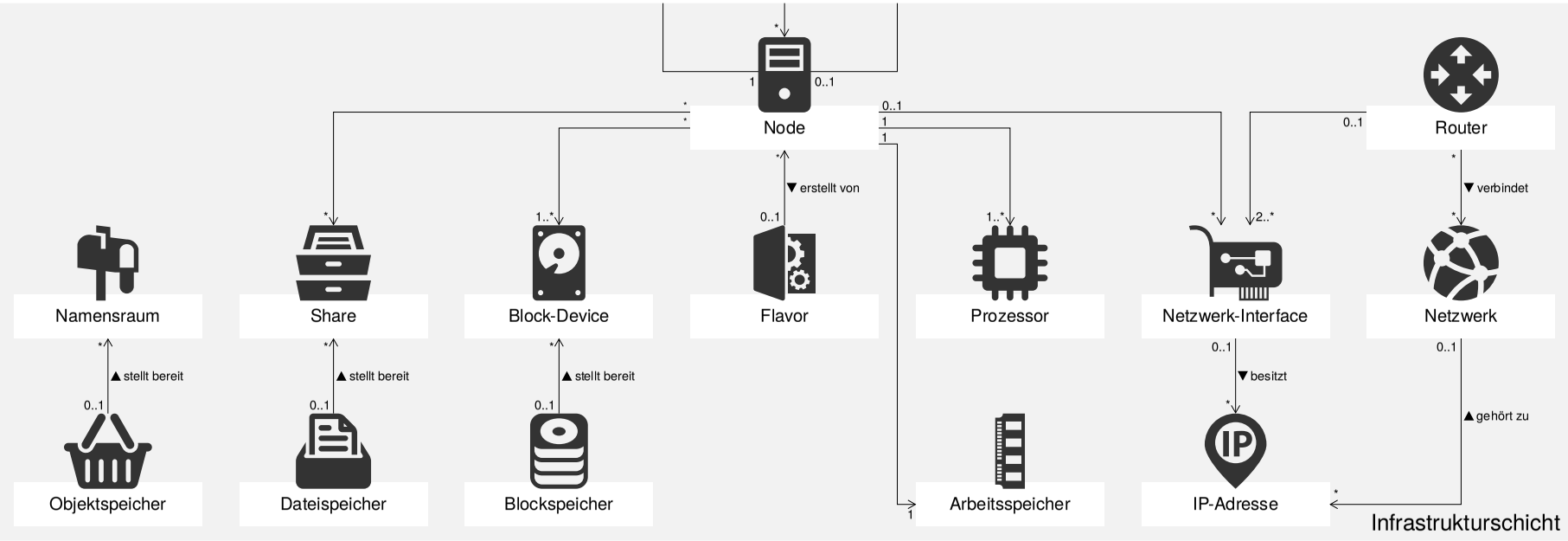


Abbildung 6.6: Managementobjekte der Anwendungs-, Plattform- und Infrastrukturschicht

Cloud-Computings. Es lässt sich neben cloudbasierten auch auf virtuelle und physikalische Umgebungen anwenden. Problematisch ist, dass das Servicemodell zu grobgranular spezifiziert ist und ein zu hohes Abstraktionsniveau besitzt. Insbesondere bleibt die Servicebereitstellung in Eigenverantwortung gänzlich unberücksichtigt. Als Strukturmuster zur Kategorisierung, Aufteilung und Zuordnung von Managementobjekten leistet es aber einen wichtigen Beitrag.

6.3.1 Infrastrukturschicht

Auf der Infrastrukturschicht werden Services für Storage, Netzwerk und Compute bereitgestellt. Diese drei Bereiche sind wie folgt modelliert:

6.3.1.1 Storage

Die Abbildung 6.7 zeigt die Managementobjekte der Storage-Schicht, die jeweils unterschiedliche Speichertechnologien und -verfahren repräsentieren. Es handelt es sich um:

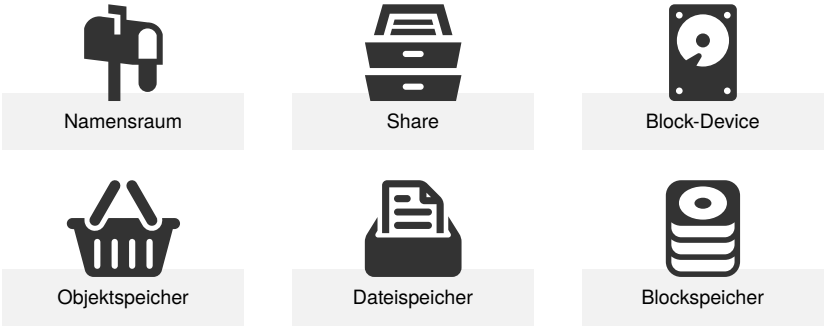


Abbildung 6.7: Managementobjekte der Storage-Schicht

Objektspeicher Der *Objektspeicher* (engl. *Object Storage*) steht stellvertretend für eine Speichertechnologie, in der große unstrukturierte Datenmengen für die Langzeitspeicherung redundant in einem skalierbaren Datenspeicher abgelegt sind. Daten werden als *Objekte* bezeichnet, die zusätzlich zu den reinen Nutzdaten, Metadaten und einen eindeutigen Identifikator besitzen. Objekte werden als *Binary Large Object* (BLOB) gespeichert und lassen sich entweder über den Identifikator direkt referenzieren oder über Metadaten auffinden. Objektspeicher kommen aufgrund ihrer guten Skalierbarkeit, des API-basierten Zugriffs und der Möglichkeit zur Nutzung von Namensräumen bevorzugt in cloudbasierten Applikationen zum Einsatz. Objektspeicher verfügen über große Speicherkapazitäten und sind besonders geeignet für die Ablage von Big Data. Der Verzicht auf Indizierung und der Einsatz von Replikation beschleunigen den Zugriff. Objektspeicher werden unter anderem bereitgestellt von: *OpenStack Swift*, *Amazon Simple Storage Service* (Amazon S3), *Azure Blob Storage* und *Ceph Reliable Autonomic Distributed Object Storage* (Ceph RADOS).

Namensraum Objektspeicher stellen Anwendern dedizierte *Namensräume* zur Verfügung. Diese besitzen keinen hierarchischen Aufbau, sondern bieten lediglich eine flache Struktur von Objekten gleicher Ordnung. Objekte lassen sich somit nicht innerhalb anderer Objekte platzieren. Dies vereinfacht die Speicherung und das Wiederauffinden von Objekten.

Dateispeicher In einem *Dateispeicher* (engl. *File Storage*) werden Daten hierarchisch in Dateien und Ordnern abgelegt und über die Dateisystemschnittstelle des Betriebssystems

zugänglich gemacht. Neben den reinen Nutzdaten hält das Dateisystem zu jeder Datei auch eine Reihe zusätzlicher Metadaten bereit wie Beschreibung, Besitzer, Zugriffsrechte und Größe. Verteilte Dateisysteme, die Dateien und Ordner über ein Kommunikationsnetzwerk zugänglich machen, sind: *Network File System* (NFS) [HN15], *Common Internet File System* (CIFS) [Mic17], *Ceph File System* (CephFS) und *Gluster File System* (GlusterFS).

Share Ein *Share* entspricht einer entfernten Datei- und Ordnerablage, die sich in ein bestehendes Dateisystem einhängen bzw. mounten lässt. Dafür sind auf dem Zielsystem technologiespezifische Treiber erforderlich. Auf die entfernten Dateien kann anschließend genauso zugegriffen werden, als ob sie sich im lokalen Dateisystem befinden würden.

Blockspeicher Der *Blockspeicher* ist die typische native Speicherschnittstelle der meisten Speichermedien. Daten werden in Sequenzen von Bits oder Bytes überführt und auf Blöcke fester Länge aufgeteilt. Jeder Block besitzt eine eindeutige Adresse, aber keine zusätzlichen Metadaten. Der Blockspeicher eignet sich nicht als Lösung für den nebenläufigen Zugriff innerhalb verteilter Systeme. Er kann zu einem Zeitpunkt ausschließlich von höchstens einem Computersystem verwendet werden. Der entfernte Zugriff erfolgt über die Protokolle: *Internet Small Computer Systems Interface* (iSCSI), *ATA over Ethernet* (AoE) oder *Fibre Channel* (FC).

Block-Device Blockspeicher eignen sich besonders für geschwindigkeitssensitive Daten und werden in Form von *Block-Devices* bereitgestellt. Virtualisierte Block-Devices werden auch als *Volumes* bezeichnet. Zustandsgrößen sind:

- Gesamtkapazität
- Lese-/Schreibgeschwindigkeit
- Lese-/Schreiblatenz
- Lese-/Schreibfehler

Neben der rein technologischen Speicherklassifikation lassen sich zudem unterscheiden:

Flüchtiger Speicher Der *flüchtige Speicher* (engl. *Ephemeral Storage*) ist an die Lebensdauer des Systems gebunden und wird bei dessen Beendigung automatisch mit gelöscht. Alle darauf gespeicherten Daten gehen verloren. Er dient als Speicherort des Betriebssystems sowie als Ablage temporärer Arbeitsdaten und Cache-Informationen. Sollen Daten über die Lebensdauer des Systems hinaus verfügbar sein, muss nicht-flüchtiger Speicher zum Einsatz kommen.

Persistenter Speicher Zur Speicherung permanenter Daten ist *persistenter Speicher* (engl. *Persistent Storage*, *Volume Storage*) erforderlich. Daten existieren unabhängig vom System und werden bei dessen Terminierung nicht automatisch mit gelöscht. Allerdings erlaubt auch dieser Speicher keinen nebenläufigen Zugriff und kann dementsprechend zu einem Zeitpunkt von höchstens einem System verwendet werden. Das Abhängen bzw. Dismounten sowie das Einhängen in ein anderes System ist jederzeit verlustfrei möglich.

Gemäß Abbildung 6.6 werden Namensräume von Objektspeicher, Shares von Dateispeicher sowie Block-Devices von Blockspeicher bereitgestellt. Diese Beziehung gilt jedoch vorrangig für virtualisierte Speicherlösungen. Datei- und Blockspeicher lassen sich auch mit Hilfe physikalischer Hardwarekomponenten realisieren. Für Dateispeicher existiert *netzgebundener Speicher* (engl. *Network Attached Storage*, NAS), für Blockspeicher Permanent- und Massenspeicher

als *Hard Disk Drive* (HDD) oder *Solid State Drive* (SSD). Bei Permanent- und Massenspeicher handelt es sich in der Regel um dedizierten Speicher, der physikalisch an einen Bare-Metal-Server gebunden und diesem exklusiv zur Nutzung überlassen ist. Das Managementobjekt eines Projekts erlaubt es, Namensräume, Shares und (native) Block-Devices unabhängig von ihrer Bereitstellung zu erfassen. Dies wird durch eine optionale Beziehung zu den Speicherkomponenten abgebildet.

6.3.1.2 Netzwerk

Je nach Fähigkeiten der Ausführungsumgebung bietet die Managementschnittstelle die Möglichkeit, Kommunikationsnetzwerke dynamisch zu erstellen und über Router miteinander zu verbinden. Dies gestattet den Datenaustausch zwischen Computersystemen, die sich in unterschiedlichen Kommunikationsnetzwerken befinden. Die Abbildung 6.8 zeigt die Managementobjekte der Netzwerkschicht. Es handelt es sich um:

Abbildung 6.8: Managementobjekte der Netzwerkschicht

Netzwerk-Interface Ein *Netzwerk-Interface* ermöglicht einem Computersystem den Zugang zu einem Kommunikationsnetzwerk. Es stellt den technischen Anschlusspunkt dar. Zustandsgrößen sind nach [BSI13]:

- Status
- Auslastung
- Empfangs- und Übertragungsrate
- Anzahl fehlerhafter Datenpakete

IP-Adresse Eine *IP-Adresse* wird innerhalb eines Kommunikationsnetzwerks verwendet, um ein Computersystem identifizieren und zwecks Datenaustausch eindeutig adressieren zu können. Zur Anbindung ist ein Netzwerk-Interface erforderlich, dem ein oder mehrere IP-Adressen zugewiesen werden. IP-Adressen existieren unabhängig von ihrer Zuweisung und sind an den Lebenszyklus des Projekts gebunden. Es lassen sich unterscheiden:

Private IP-Adresse Eine *private IP-Adresse* (engl. *Private IP*) ist eine IP-Adresse, die nicht im Internet vergeben ist. Datenpakete dürfen demnach nicht in das öffentliche Netzwerk geroutet werden.

Floating IP-Adresse Eine *Floating IP-Adresse* (engl. *Floating IP*) oder *elastische IP-Adresse* (engl. *Elastic IP*) ist eine öffentliche IP-Adresse, die sich dynamisch an ein Netzwerk-Interface binden bzw. vom selbigen wieder entfernen lässt. Sie macht das Computersystem über ein öffentliches Netzwerk erreichbar.

Netzwerk Als *Netzwerk* bezeichnet man einen Verbund aus mehreren eigenständigen Computersystemen, der den Datenaustausch untereinander ermöglicht. Jedes Computersystem wird weiterhin als Einzelsystem aufgefasst, das bei Bedarf anderen Teilnehmern Informationen zukommen lässt oder von diesen entgegennimmt. Die Netzwerkadresse repräsentiert einen Adressbereich, aus dem jedes verbundene Computersystem eine Adresse bezieht. Je nach Adressbereich handelt es sich entweder um ein *privates Netzwerk* oder ein *öffentliches Netzwerk*. Zustandsgrößen sind nach [BSI13]:

- Paketverzögerung (Latenzzeiten, Einweg- und Umlaufverzögerung)
- Abweichungen der Laufzeit (engl. Jitter)
- Durchsatz
- Paketverluste
- Vertauschung der Ankunftsreihenfolge von Datenpaketen (engl. Packet Reordering)

Router Ein *Router* ist ein Netzwerkgerät der Vermittlungsschicht, das Datenpakete zwischen Kommunikationsnetzwerken weiterleiten kann. Er verfügt über mindestens zwei Netzwerk-Interfaces und dient der Koppelung von Teilnetzen. Zu seinen Aufgaben zählen Vermittlung und Wegewahl, d. h. die logische Pfadschaltung zwischen zwei Computersystemen. Zustands- und Stellgrößen sind:

- CPU-Auslastung
- Speicherauslastung
- Routing-Tabelle

Firewallregel Eine *Firewallregel* hat die Funktion eines Netzwerkportfilters und gibt einen Port oder Portbereich für eingehenden bzw. ausgehenden öffentlichen Datenverkehr frei.

Sicherheitsgruppe Firewallregeln werden zu *Sicherheitsgruppen* (engl. *Security Groups*) zusammengefasst, die alle öffentlich zugänglichen Ports umfassen und dem Prinzip folgen: Alles, was nicht explizit erlaubt wurde, ist verboten. Mittels Sicherheitsgruppen lässt sich der öffentliche Datenverkehr filtern und eine erste Sicherheitsbarriere aufbauen. Sie verhindert, dass unerwünschte Datenpakete das Projektnetzwerk erreichen. Sicherheitsgruppen sind im Allgemeinen zustandsbehaftet, d. h. Antworten auf erlaubte eingehende Datenpakete sind gestattet, auch wenn keine explizite Freigabe vorliegt. Die Abbildung 6.9

Abbildung 6.9: Managementobjekt zur Repräsentation von Sicherheitsgruppen

zeigt das Managementobjekt zur Repräsentation von Sicherheitsgruppen, deren Zuordnung zu einem Computersystem sich dynamisch anpassen lässt.

Das Managementobjekt zur Repräsentation der Netzwerkschicht deckt zunächst nur die Vermittlungsschicht des Kommunikationsstapels ab. Die Sicherungsschicht wird seitens des Basismodells nicht erfasst. Der Einsatz entsprechender Erweiterungen gestattet es aber auch, Netzwerkgeräte wie Switches und Bridges sowie die physikalische Netzwerktopologie abzubilden. Die Netzwerkschicht vereinheitlicht die netzwerktechnischen Fähigkeiten der unterschiedlichen Ausführungsumgebungen. Insbesondere in cloudbasierten und virtuellen Umgebungen ist die Festlegung der Netzwerktopologie auf der Sicherungsschicht seitens des Managers zumeist nicht möglich. In aller Regel lässt sich aber die logische Netzwerktopologie der Vermittlungsschicht dynamisch anpassen. Eine Besonderheit ist, dass IP-Adressen aus Sicht des Managements über einen eigenständigen Lebenszyklus verfügen und daher projektseitig zu verwalten sind. In cloudbasierten und virtuellen Umgebungen verursachen private IP-Adressen in der Regel keine Kosten. Hingegen werden Floating IPs gegen Bezahlung eines Entgelts aus einem gemeinsamen Adresspool bezogen und sind dem Anwender nach der Reservierung zur exklusiven Nutzung überlassen.

6.3.1.3 Compute

Innerhalb cloudbasierter und virtueller Umgebungen lassen sich virtuelle Cluster von Computersystemen erstellen. Die Kombination aus Storage-, Netzwerk- und Compute-Services ermöglicht den Aufbau komplexer Infrastrukturen und ganzer virtueller Rechenzentren. Die Abbildung 6.10 zeigt die Managementobjekte der Compute-Schicht. Es handelt sich um:

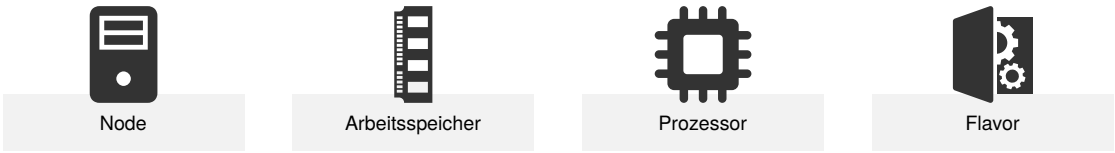


Abbildung 6.10: Managementobjekte der Compute-Schicht

Node Als *Node* wird ein Computersystem bezeichnet, auf dem sich ein Betriebssystem und Anwendungen ausführen lassen. Es repräsentiert eine in sich abgeschlossene Ausführungsumgebung zur Durchführung mathematischer Berechnungen und logischer Entscheidungen. Ein Node kann über ein Netzwerk-Interface mit einem Netzwerk verbunden werden. Nach Zuweisung einer freien IP-Adresse aus dem Adressbereich ist der Empfang und Versand von Datenpaketen möglich. Einem Node lassen sich beliebig viele Block-Devices und Shares zuordnen. Darüber hinaus repräsentiert er ein physikalisches oder virtuelles Computersystem:

Bare-Metal-Server Ein *Bare-Metal-Server* bezeichnet einen physikalischen Server, der exklusiv seinem Besitzer zur Verfügung steht. Er besteht aus physikalischen Hardwarekomponenten und wird in der Regel in einem Rechenzentrum betrieben. Auf Bare-Metal-Servern wird das Betriebssystem direkt auf der Hardware installiert. Durch Verzicht auf zusätzliche Abstraktionsschichten lässt sich gegenüber virtuellen Computersystemen eine bessere Performance erzielen.

Virtuelle Maschine Eine *virtuelle Maschine* emuliert eine physikalische Hardwareumgebung. Anfragen an CPU, Speicher, Festplatte sowie andere Hardwarekomponenten

werden von der Virtualisierungsschicht verwaltet, koordiniert und an die darunterliegende Hardware weiterleitet. Virtuelle Maschinen werden mittels eines Typ-1- oder Typ-2-Hypervisors erzeugt und stellen eine vollvirtualisierte Ausführungsumgebung bereit. Ein Typ-1-Hypervisor (Bare-Metal-Hypervisor) setzt direkt auf der Hardware auf, wohingegen ein Typ-2-Hypervisor (Hosted Hypervisor) ein auf dem Computersystem installiertes Betriebssystem erfordert.

Container Ein *Container* besitzt keinen eigenen Systemkern (engl. Kernel), sondern ist an den des Host-Betriebssystems gebunden. Es handelt sich um eine Betriebssystemvirtualisierung, wobei die Hardwareaufrufe nur auf einem einzigen Betriebssystem koordiniert werden. Dies macht Container im Vergleich zu ihren vollvirtualisierten Vertretern sehr viel leichtgewichtiger. Dadurch lassen sich die Ressourcen des Host-Systems effizienter einsetzen, so dass ein einzelner Host ohne spürbare Performanceeinbußen mehr Container gleichzeitig ausführen kann als dies bei virtuellen Maschinen der Fall wäre.

Arbeitsspeicher Beim *Arbeitsspeicher* handelt es sich um einen Direktzugriffsdatenspeicher zur temporären Ablage von Programmen und Daten. Zustandsgrößen sind:

- Verfügbarer Arbeitsspeicher
- Belegter Arbeitsspeicher
- Freier Arbeitsspeicher

Prozessor Der *Prozessor* stellt die zentrale Recheneinheit in einem Computer dar. Zustandsgrößen sind nach [BSI13]:

- Taktung
- Auslastung
- Temperatur
- Spannung
- Stromverbrauch

Flavor Als *Flavor* wird ein Maschinentyp oder -modell bezeichnet, das festlegt, welche Ressourcen einem Node in Form von CPU-Anzahl, Arbeitsspeichergröße und Festplattenplatz zur Verfügung stehen. Flavors sind von der Ausführungsumgebung abhängig und definieren Maschinentypen, die mitunter für spezielle Anwendungsszenarien optimiert sind. Dazu zählt das *High-Performance-Computing* (HPC), das Rechenarbeiten umfasst, deren Bearbeitung eine hohe Rechenleistung oder Speicherkapazität erfordert.

Die Compute-Schicht verbindet die Storage- und Netzwerkschicht und macht Nodes zu den zentralen Managementobjekten der Infrastrukturschicht. Nodes verfügen über die grundlegenden Funktionen, um Services der übergeordneten Schichten in Eigenverantwortung betreiben zu können. Die Lücke zwischen der Anwendungs- und der Infrastrukturschicht wird durch die Plattformschicht geschlossen, in der ein Großteil des Applikationsstapels (Betriebssystem, Laufzeitumgebungen und Middleware) repräsentiert wird. Nodes bilden die gemeinsame Abstraktion von Computersystemen, die jeweils in den Ausführungsumgebungen zum Einsatz kommen. Reaktionen auf Lastspitzen erfolgen durch horizontale oder vertikale Skalierung. Bei der *horizontalen Skalierung* werden zusätzliche Nodes erzeugt, die zur Entlastung der Anfragebearbeitung eingesetzt werden. Bei der *vertikalen Skalierung* erfolgt ein Wechsel auf

ein leistungsfähigeres Flavor, mit dem auch ein Wechsel der Ausführungsumgebung und damit des Projekts einhergehen kann. Die horizontale Skalierung bewirkt die Erstellung zusätzlicher Nodes, die vertikale Skalierung das Löschen des bestehenden Nodes und seine Neuerstellung auf Basis eines leistungsfähigeren Flavors im Zielprojekt.

6.3.2 Plattformschicht

Die Plattformschicht bietet Applikationen und Services ein höheres Abstraktionsniveau und stellt grundlegende Verteilungs- und Kommunikationsdienste zur Verfügung, die Entwicklung und Betrieb vereinfachen. Die Plattform selbst entspricht im engeren Sinne keiner eigenständigen Applikation. Sie bietet ausschließlich einen Ordnungs- und Entwicklungsrahmen, der die Anwendungsarchitektur bestimmt. Innerhalb der Plattformschicht werden Services zur Applikationsausführung in Form von Laufzeitumgebungen und Middleware erfasst. Die Abbildung 6.11 zeigt die Managementobjekte der Plattformschicht. Es handelt sich um:

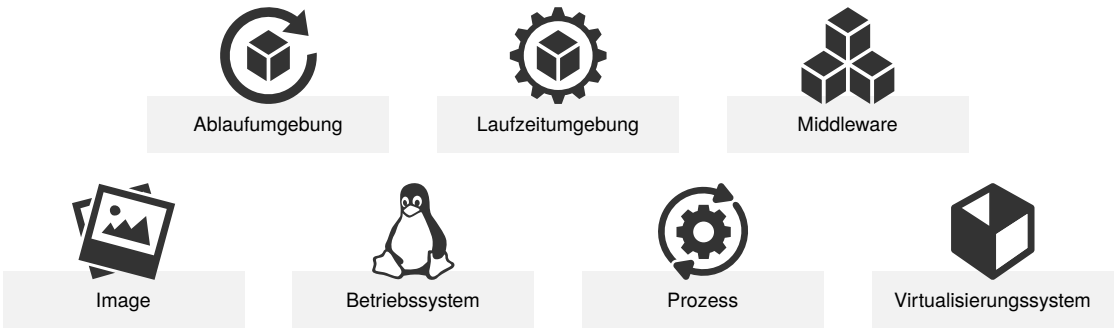


Abbildung 6.11: Managementobjekte der Plattformschicht

Ablaufumgebung Eine *Ablaufumgebung* entspricht einer Softwareschicht, die die Ausführung von Softwarekomponenten ermöglicht. Diese können wiederum Ablaufumgebungen zur Verfügung stellen, wodurch sich ein Technologiestapel abbilden lässt.

Laufzeitumgebung Eine *Laufzeitumgebung* beschreibt die zur Ausführungszeit von Anwendungen verfügbaren und vorausgesetzten Funktionen eines Laufzeitsystems. Dieses ist durch die elementaren Bestandteile einer Programmiersprache festgelegt und beschreibt ein spezifisches Ablauf- und Ausführungsmodell. Beispiele sind die Java-, Python- und Ruby-Laufzeitumgebung.

Middleware Eine *Middleware* oder *anwendungsorientierte Middleware* ist Vermittler zwischen Betriebssystem und Anwendung. Sie definiert ein eigenes Komponentenmodell und bietet vielfach eine logische Infrastruktur zur Kommunikation. Sie ergänzt die Ablaufumgebung um eine Laufzeitfunktionalität sowie um unterstützende Services für die Anwendungsschicht. Die *OSGi-Service-Plattform* ist eine unter Java weit verbreitete Middleware mit eigenem Komponentenmodell und einem Lebenszyklusmanagement von Softwarekomponenten.

Image Ein *Image* oder *Template* enthält das Abbild eines Betriebssystems, von dem sich direkt Nodes erzeugen lassen. Dies erspart die mehrmalige Installation ein und desselben Betriebssystems.

Betriebssystem Das *Betriebssystem* (engl. *Operating System*) entstammt einem Image und bildet gemeinsam mit dem Flavor die Grundlage zum Betrieb eines Nodes. Es steuert und überwacht die Ausführung von Programmen. Weitere Aufgaben sind die Verwaltung der Software- und Hardwarekomponenten sowie die Koordination der Zugriffe. Linux- und Windows-Distributionen sind die derzeit gängigsten Betriebssysteme.

Prozess Die Ausführung eines Services erfolgt innerhalb eines *Prozesses*, der vom Betriebssystem verwaltet wird. Das Managementobjekt ist in der Abbildung 6.12 dargestellt. Einem Prozess sind beliebig viele Services zugeordnet, über den die Zustands- und Ablaufsteuerung erfolgt. Damit sich auch Prozesse erfassen lassen, deren Services nicht durch Managementobjekte repräsentiert sind, ist die Assoziation als optional gekennzeichnet. Des Weiteren sind Services nicht notwendigerweise an einen Prozess gebunden. Dies gilt vorrangig für fremdbezogene Services, die nicht der eigenen Kontrolle unterstehen und für die keine technischen Informationen über Bereitstellung und Betrieb vorliegen. Zustandsgrößen eines Prozesses sind:

- Startzeitpunkt
- CPU-Nutzung
- Verwendeter Hauptspeicher
- Lese-/Schreibrate (Disk-I/O)
- Empfangs-/Übertragungsrate (Netzwerk-I/O)

Virtualisierungssystem Ein *Virtualisierungssystem* bietet eine virtuelle Laufzeitumgebung zur Erstellung und Ausführung virtueller Computersysteme auf einem Wirtssystem. Handelt es sich dabei selbst um ein virtuelles Computersystem, spricht man auch von *verschachtelter Virtualisierung*. Das Virtualisierungssystem repräsentiert sowohl die hypervisorbasierte als auch containerbasierte Virtualisierungstechnologie. Zudem ist die Abbildung eines Typ-1- und Typ-2-Hypervisors möglich, abhängig davon, ob auf dem Wirtssystem ein Betriebssystem erforderlich ist oder nicht. Die Gastsysteme werden wiederum durch Nodes repräsentiert und sind entweder virtuelle Maschinen oder Container. Typischerweise wird ein Virtualisierungssystem auf einem Bare-Metal-Server installiert. Verschachtelte Virtualisierung findet im Produktivbetrieb selten Anwendung, da mit jeder zusätzlichen Virtualisierungsschicht zugleich auch eine Verschlechterung der Performance einhergeht.

Die in der Abbildung 6.6 dargestellte Vererbungshierarchie innerhalb der Plattformschicht bildet den Ausgangspunkt zur Repräsentation des Applikationsstapels. Laufzeitumgebungen und Middleware sind Erweiterungen von Ablaufumgebungen. Des Weiteren entsprechen Betriebssysteme und Virtualisierungssysteme Laufzeitumgebungen. Betriebssysteme stellen beliebig viele Ablaufumgebungen bereit, die ihrerseits Softwarekomponenten ausführen. Beide Generalisierungsgruppen sind erweiterbar und gestatten die Modellierung spezieller Laufzeitumgebungen und Middleware. Das Paketmanagement seitens des Betriebssystems wie *Advanced Packaging*

Abbildung 6.12: Managementobjekt zur Repräsentation von Systemprozessen

Tool (APT) unter Debian-Distributionen oder *Yellowdog Updater, Modified* (YUM) unter Red-Hat-Distributionen lässt sich als Middleware auffassen, die direkt in das Betriebssystem integriert ist und Softwarekomponenten in Form von Softwarepaketen verwaltet. Diese können Ablaufumgebungen bereitstellen, wodurch sich Laufzeitumgebungen für Programmiersprachen zur Verfügung stellen lassen. Diese können ihrerseits Softwarekomponenten ausführen, die wiederum Services anbieten. Durch den Einsatz von Bindungen lassen sich die Nutzungsbeziehungen zwischen den Ablaufumgebungen repräsentieren, wodurch sich ein Applikationsstapel abbilden lässt.

6.3.3 Anwendungsschicht

Applikationen und Services sind in der Anwendungsschicht angesiedelt. Sie stellt die oberste Schicht innerhalb der Bereitstellung dar und baut logisch auf den beiden vorhergehenden Schichten auf. Die Abbildung 6.13 zeigt die Managementobjekte der Anwendungsschicht. Es handelt sich um:

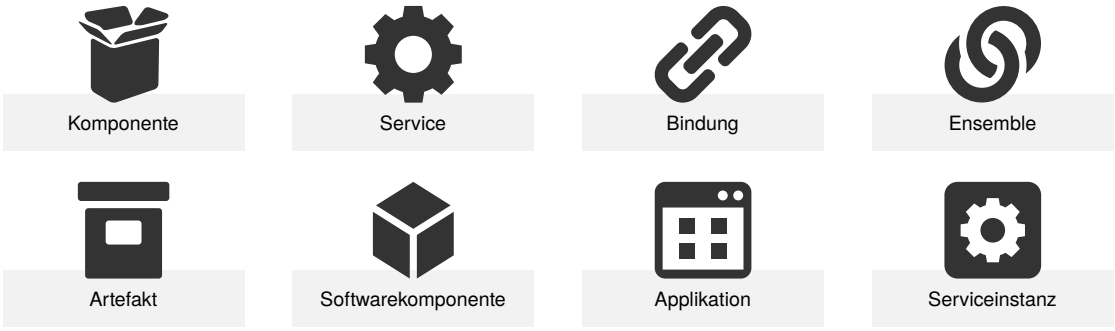


Abbildung 6.13: Managementobjekte der Anwendungsschicht

Artefakt Ein *Artefakt*, auch *Softwareartefakt* genannt, repräsentiert eine Deployment-Unit, die innerhalb einer Ablaufumgebung eingesetzt und ausgeführt werden kann. Dabei handelt es sich um ein Softwarepaket, vielfach in Form einer komprimierten Archivdatei, das neben dem ausführbaren Code ergänzende Metadaten enthält. Die Verwaltung erfolgt in Artefakt-Repositorys, in denen die Softwarepakete strukturiert abgelegt sind und die sich bei Bedarf über ein Kommunikationsnetzwerk herunterladen lassen.

Komponente Eine *Komponente* repräsentiert eine Hardware- oder Softwarekomponente, die Services zur Verfügung stellt. Die Abbildung 6.14 zeigt die dazugehörige Vererbungshierarchie, die unter anderem eine Erweiterung durch zusätzliche Hardwarekomponenten gestattet. Die Generalisierungsgruppe klassifiziert Nodes, Router, Prozessoren, Arbeitsspeicher und Netzwerk-Interfaces als Hardwarekomponenten, Objekt-, Datei- und Blockspeicher sowie Namensräume, Shares und Block-Devices als Speicherkomponenten. Während der Service die Abstraktion einer Funktion darstellt, entspricht die Komponente ihrer Implementierung in Form von Hardware (elektronischer Schaltung) oder Software (ausführbarer Programmcode).

Softwarekomponente Eine *Softwarekomponente* ist die Laufzeitrepräsentation eines Artefakts. Softwarekomponenten interagieren untereinander und mit ihrer Ablaufumgebung gemäß den Vorgaben des Laufzeitsystems. Sie besitzen wohldefinierte Schnittstellen

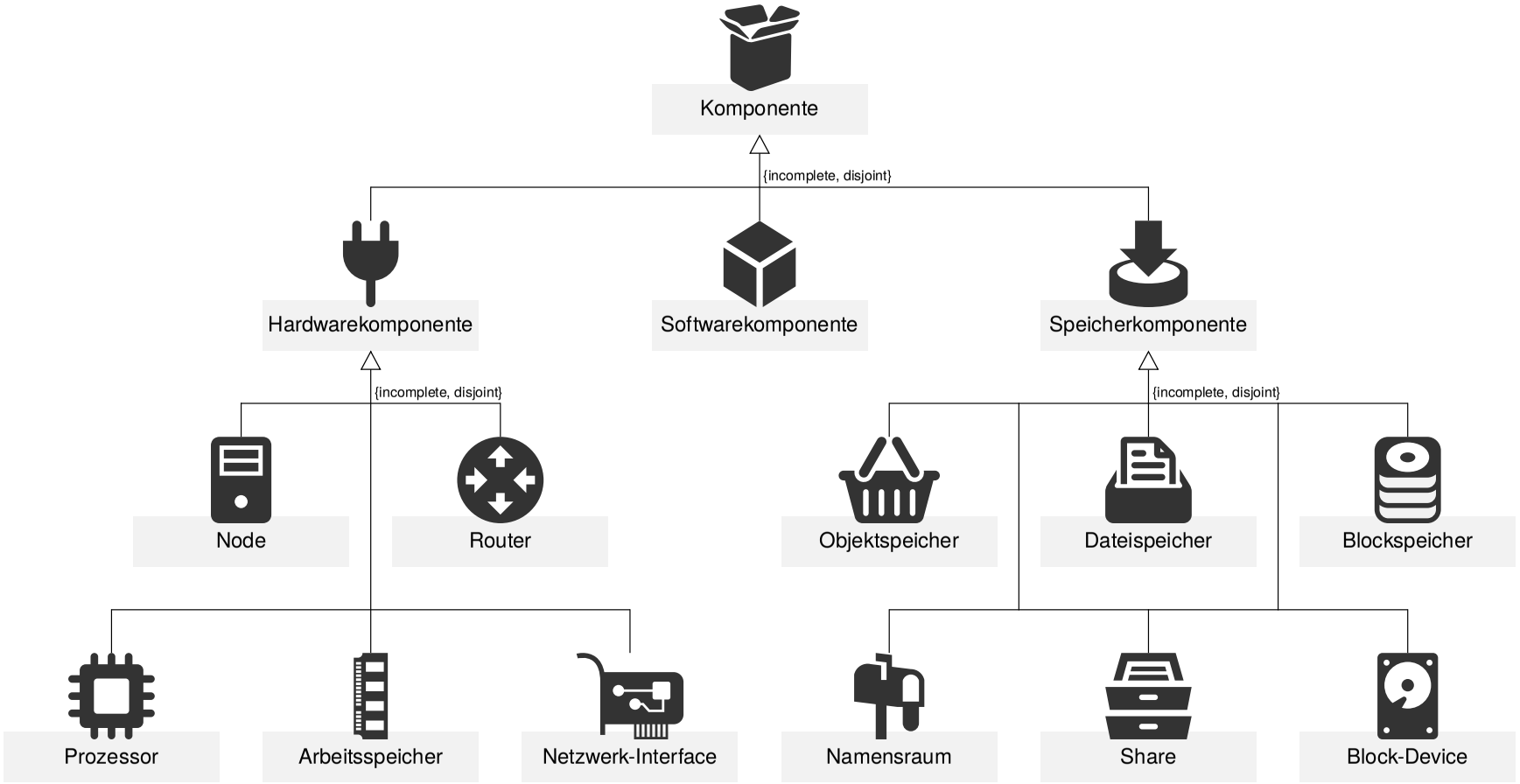


Abbildung 6.14: Vererbungshierarchie zur Repräsentation von Hardware-, Software- und Speicherkomponenten

sowie explizite und implizite Abhängigkeiten zu anderen Softwarekomponenten und der Ablaufumgebung.

Service Ein *Service* entspricht einer Funktion bzw. einer Menge von Funktionen, die von Komponenten bereitgestellt wird. Ein Service stellt eine definierte Leistung zur Verfügung und lässt sich als Baustein in Prozessabläufen einsetzen. Dabei muss es sich nicht notwendigerweise um einen Service gemäß dem SOA-Modell handeln. Ein Service gilt vielmehr als technische Dienstleitung, die einen Beitrag zu einem Anwendungsfall leistet. Demnach umfasst der Service-Begriff auch Applikationen und Ablaufumgebungen in Form von Laufzeitumgebungen, Betriebssystemen und Middleware. Ein Service wird als Managementobjektklasse repräsentiert und erfordert seitens des Informationsmodells entsprechende Verfeinerungen. Die Vererbungshierarchie ist so entworfen, dass Services entweder eine fachliche oder eine technische Funktion erfüllen. Somit ist jedem Service ein klar abgegrenzter Aufgaben- und Verantwortungsbereich zugeordnet.

Bindung Eine *Bindung* repräsentiert eine Beziehung zwischen einem initiierenden Service und seinen gebundenen Services. Dabei handelt es sich zumeist um Nutzungs-, Bereitstellungs- und Abhängigkeitsbeziehungen. Dadurch lassen sich neben der Überwachung der Services selbst auch die Beziehungen untereinander überwachen. Bindungen erfassen das Beziehungsgeflecht zwischen den Services und bilden dieses durch Managementobjekte ab. Des Weiteren machen Bindungen Angaben darüber, ob sie sich durch das Management rekonfigurieren lassen oder ob sie gänzlich unveränderlich sind.

Ensemble Häufig benötigt ein Service mehrere andere Services in einem engen Zusammenhang. Diese spezielle Art von Bindung wird als *Ensemble* [PKH+08, FLL+09] bezeichnet. In diesem Fall ist der Service darauf angewiesen, dass ihm alle Elemente des Ensembles für eine gewisse Zeitspanne exklusiv zur Verfügung stehen. Im Rahmen eines zeitlich befristeten *Leases* garantieren die Mitglieder die Einhaltung der seitens des Initiators gestellten funktionalen und nicht-funktionalen Anforderungen. Das gesamte Ensemble wird ungültig, sobald eine Bindung ungültig wird.

Applikation Eine *Applikation* entspricht einer Anwendungssoftware, die Endanwendern zur Problemlösung dient. Applikationen können dafür auf die von Services bereitgestellten Funktionen zurückgreifen. Zu diesem Zweck gehen Applikationen mit anderen Services Nutzungsbeziehungen ein, die durch Bindungen repräsentiert werden.

Serviceinstanz Eine *Serviceinstanz* stellt eine in sich abgeschlossene Funktion zur Verfügung. Serviceinstanzen entsprechen autarken, lose koppelbaren und austauschbaren Services gemäß dem SOA-Modell. Sie werden über öffentliche Schnittstellen aufgerufen, die zumeist über das Internet zugänglich sind. Dafür kommt ein einheitlicher Mechanismus zum Einsatz, der Services plattformunabhängig miteinander verbindet und die technischen Details der Kommunikation verbirgt.

Applikationen und Serviceinstanzen stellen, wie ein Großteil der Managementobjekte der Plattformschicht, Services dar. Dadurch verschwimmt die Grenze zwischen Anwendung und Plattform. Zudem lassen sich über Bindungen die Nutzungsbeziehungen zwischen den Services beider Schichten durch Managementobjekte einheitlich abbilden. Im Rahmen der Anwendungsschicht gilt es insbesondere auch, fremdbezogene Services zu erfassen. Aus diesem Grund ist die Beziehung zwischen Komponente und Service als optional modelliert. Durch

die serviceorientierte Sicht innerhalb der Plattformschicht ist es zudem möglich, Laufzeitumgebungen und insbesondere auch Middleware als fremdbezogene Services, d. h. unabhängig von Betriebssystem und Node zu erfassen. Eine Eigenschaft, die sich dadurch ergibt, dass die Vererbungshierarchie zur Beschreibung von Services schichtübergreifend definiert ist und nicht auf die Anwendungsschicht beschränkt bleibt.

6.4 Servicegüte

Neben der Repräsentation der qualitativen Eigenschaften von Services müssen zudem die Servicegütevereinbarungen erfasst werden. Die in diesem Zusammenhang erforderlichen Managementobjekte sind in der Abbildung 6.15 dargestellt. Das organisatorische Rollenmodell klassifiziert die an der Servicebereitstellung bzw. -nutzung beteiligten Rechtssubjekte. Es dient als Ausgangsbasis zur Ausgestaltung rechtsverbindlicher Vertragsbeziehungen. Die Vererbungshierarchie zur Beschreibung der organisatorischen Rollen ist in der Abbildung 6.16 dargestellt. Es handelt sich um:

Service-Provider Der *Service-Provider* ist eine juristische Person, die mit einem Service-Customer einen Vertrag über einen zu erbringenden Service schließt. Mit der Unterzeichnung verpflichtet sich der Service-Provider gegenüber dem Service-Customer, den Service über den gesamten Nutzungszeitraum hinweg gemäß den vertraglich vereinbarten Bedingungen bereitzustellen und zu erbringen. Gleichzeitig muss er den Nachweis erbringen, dass die Anforderungen und Vorgaben auch tatsächlich eingehalten werden. Ihm obliegt die Pflicht, die technischen Voraussetzungen dafür zu schaffen, dass sich die qualitativen

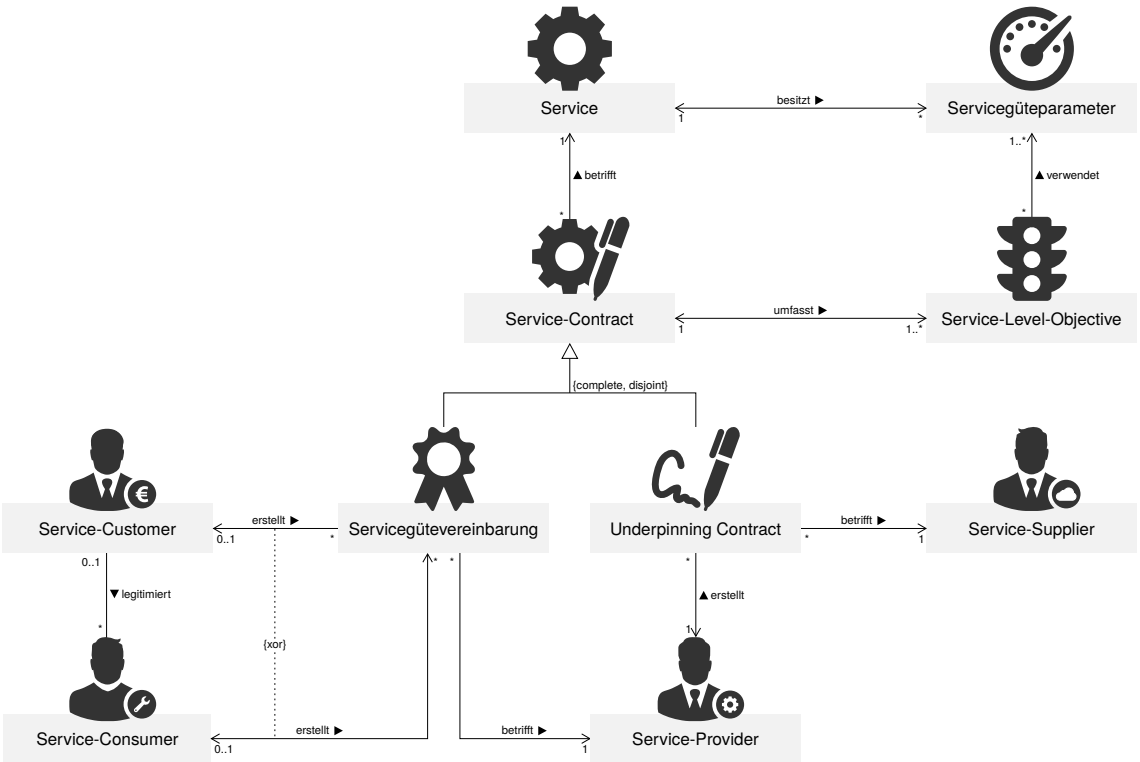


Abbildung 6.15: Managementobjekte zur Repräsentation der Servicegüte

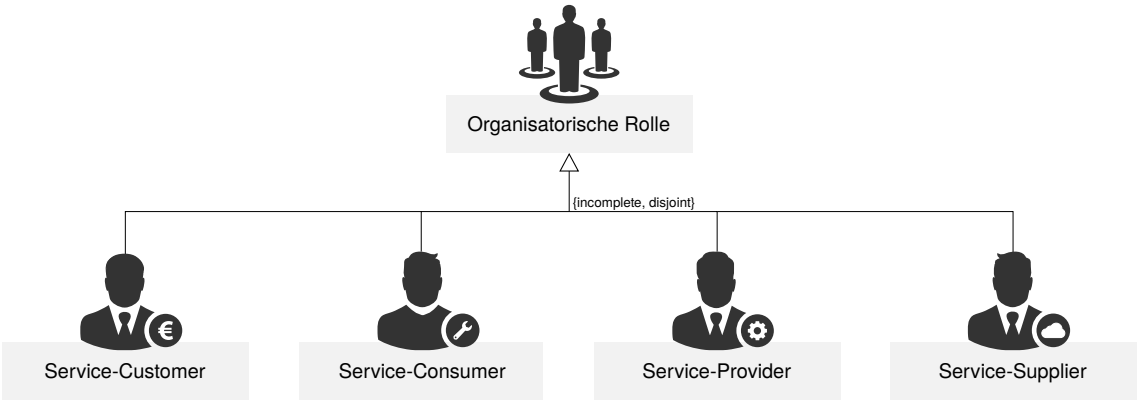


Abbildung 6.16: Managementobjekte zur Repräsentation der organisatorischen Rollen

Eigenschaften durch den Service-Customer jederzeit einsehen und überwachen lassen. Etwaig auftretende Verletzungen sind vom Service-Provider zu erfassen, zu protokollieren und zeitnah zu melden.

Service-Supplier Der *Service-Supplier* ist eine juristische Person, die einem Service-Provider unterstützende Services bereitstellt, auf die er im Rahmen der eigenen Serviceerbringung zurückgreift. Der Service-Supplier entspricht einem Zulieferer, dessen Einsatz dem Service-Customer in der Regel verborgen bleibt. Dabei handelt es sich um einen fremdbezogenenen Service, der dem Verantwortungsbereich des Service-Supplier untersteht. Er verpflichtet sich gegenüber dem Service-Provider, den Service über den Nutzungszeitraum hinweg gemäß den vertraglichen Vereinbarungen zu erbringen.

Service-Customer Der *Service-Customer* ist eine juristische Person, die zur Unterstützung der eigenen Aktivitäten einen Service in Anspruch nimmt. Zu diesem Zweck schließen der Service-Customer und der Service-Provider eine vertragliche Vereinbarung über die Servicebereitstellung. Durch Unterzeichnung verpflichtet sich der Service-Customer zur Einhaltung der Nutzungsbedingungen und akzeptiert die Höhe der Vergütung sowie die Zahlungsmodalitäten. Es ist Aufgabe des Service-Customer, vor Beginn der Vertragsverhandlungen die relevanten funktionalen und nicht-funktionalen Anforderungen zu erfassen. Dazu zählen insbesondere rechtliche Datenschutzbestimmungen beim Umgang mit personenbezogenen Daten. Greift der Service während der Verarbeitung auf schützenswerte Daten zu, muss der Service-Provider durch den Einsatz entsprechender Schutzmaßnahmen die Einhaltung aller rechtlichen Bestimmungen gewährleisten. Kommt der Service-Provider seinen vertraglichen Verpflichtungen nicht oder nur eingeschränkt nach, kann der Service-Customer Vertragsstrafen geltend machen. Dabei handelt sich in der Regel um Konventionalstrafen, die den Service-Provider zur Zahlung eines Geldbetrags verpflichten.

Service-Consumer Der *Service-Consumer* ist eine unbeschränkt geschäftsfähige natürliche Person, die mit dem Service-Customer in einem Vertragsverhältnis steht, das ihn zur Nutzung des Services legitimiert. Jede Verwendung muss sich unabstreitbar ihrem Verursacher zuordnen lassen. Dem Service-Customer wird im Anschluss gemäß dem vereinbarten Kostenmodell die vom Service-Consumer in Anspruch genommene Leistung in Rechnung gestellt.

Neben der Repräsentation der Rechtssubjekte müssen zudem die qualitativen Eigenschaften der Services sowie die vertraglich vereinbarten Zielvorgaben abgebildet werden. Zu diesem Zwecke kommen die folgenden Managementobjekte zum Einsatz:

Service-Contract Der *Service-Contract* repräsentiert einen rechtsverbindlichen Vertrag zwischen zwei Vertragspartnern. Er besteht aus den vereinbarten Zielvorgaben, die durch Service-Level-Objectives beschrieben werden.

Servicegütevereinbarung Die *Servicegütevereinbarung* (engl. *Service-Level-Agreement*, SLA) ist ein Service-Contract und wird zwischen einem Service-Provider und einem Service-Customer bzw. Service-Consumer geschlossen. Neben statisch geschlossenen Servicegütevereinbarungen zwischen Service-Provider und Service-Customer existieren zudem dynamisch geschlossene Servicegütevereinbarungen zwischen Service-Provider und Service-Consumer. Diese Servicegütevereinbarungen setzen voraus, dass vorab eine statisch geschlossene Servicegütevereinbarung vorliegt, in der die Vertragsbedingungen, die Servicegüteparameter sowie die anpassbaren Service-Level-Objektives festgelegt sind. Im Rahmen dieser Vereinbarung kann dann der Service-Consumer vor der eigentlichen Serviceinanspruchnahme Anforderungen formulieren, die Bereitstellung und Betrieb seitens des Service-Providers beeinflussen. Dabei kann es sich um Vorgaben handeln, die die Leistungsfähigkeit oder Verfügbarkeit eines Services betreffen. Grundlegendes Prinzip ist, dass dynamisch geschlossene Servicegütevereinbarungen ausschließlich Verfeinerungen statisch geschlossener Servicegütevereinbarungen darstellen. Es lassen sich daher weder die zuvor formulierten Zielvorgaben als ungültig erklären noch sind Angaben möglich, die über die definierten Freiheitsgrade hinausgehen. Es existiert somit ein rechtsverbindlicher Rahmen, der es dem Service-Provider ermöglicht, die zur Einhaltung der Zielvorgaben erforderlichen Überwachungs- und Steuerungsstrategien zu entwerfen und zu implementieren.

Underpinning Contract Ähnlich wie eine Servicegütevereinbarung ist auch ein *Underpinning Contract* ein Service-Contract, der aber zwischen Service-Provider und Service-Supplier geschlossen wird. Er entspricht einem Zulieferungsvertrag und betrifft die Bereitstellung eines unterstützenden Services, auf den der Service-Provider zur Bereitstellung seiner eigenen Services zurückgreift. Underpinning Contracts beziehen sich auf fremdbezogene Services und müssen auf die Servicegütevereinbarungen abgestimmt sein.

Servicegüteparameter Ein *Servicegüteparameter* repräsentiert eine qualitative Eigenschaft eines Services. Ein Service kann über beliebig viele Servicegüteparameter verfügen, wobei ein Servicegüteparameter wiederum genau einem Service zugeordnet ist. Die Definition von Servicegüteparametern erfolgt innerhalb des Service-Contract und beinhaltet die Festlegung von Bedeutung und Wertebereich. Hierbei muss die Forderung der Messbarkeit erfüllt sein. Ein Servicegüteparameter verfügt über Bezeichner, Beschreibung, Einheit und Wert. Servicegüteparameter stehen vielfach in direkter Abhängigkeit zum Service und müssen in Übereinkunft mit dem Service-Customer vereinbart werden. Beispiele sind:

- Durchsatz
- Antwortzeit
- Verfügbarkeit
- Kosten

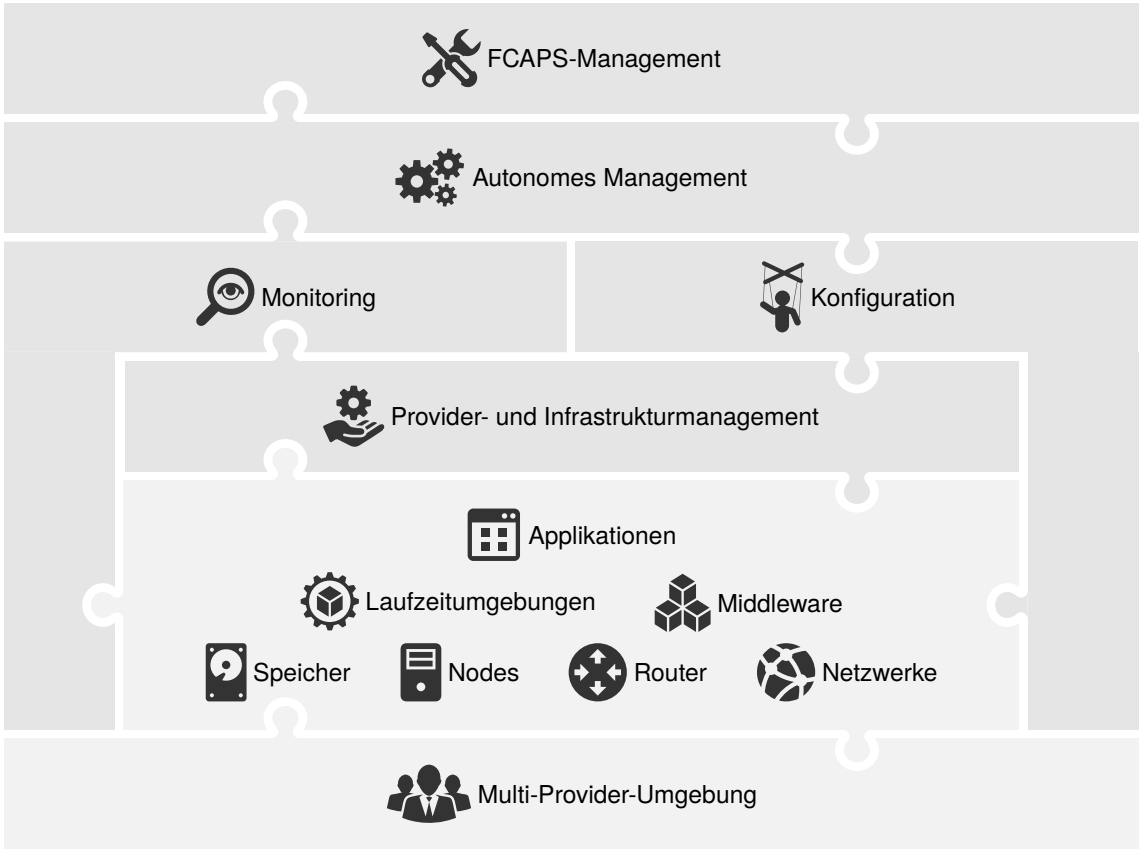


Abbildung 6.17: Funktionen für das Management einer Multi-Provider-Umgebung

Service-Level-Objective Ein *Service-Level-Objective* beschreibt eine Zielvorgabe, die über einen Servicegüteparameter formuliert ist. Ihre Einhaltung muss vom Service-Provider überwacht werden. Verletzungen oder Beeinträchtigungen gilt es, durch den Einsatz korrektiver bzw. perfektiver Maßnahmen zu beseitigen.

Die Repräsentation der Servicegüte durch Managementobjekte bildet die Grundvoraussetzung für einen gütegesicherten Servicebetrieb. Sie ermöglichen es, sich abzeichnende Zielverfehlungen frühzeitig zu erkennen, um angemessen darauf reagieren zu können. Aus Managementsicht besteht kein grundsätzlicher Unterschied zwischen statisch und dynamisch geschlossenen Servicegütevereinbarungen. Beide Arten lassen sich durch dieselbe Managementobjektklasse beschreiben. Der Unterschied zeigt sich lediglich darin, dass der Service-Provider einen Service-Consumer anstelle eines Service-Customer als Vertragspartner hat.

6.5 Funktionen

Das Management einer Multi-Provider-Umgebung bedarf einer Vielzahl eng miteinander verzahnter Funktionen und Abläufe. Von zentraler Bedeutung ist der Aufbau einer autonomen Kontrollschleife zur Gewährleistung der Servicegüteanforderungen. Dies erfordert den Einsatz automatisierter Funktionen zur Bereitstellung, Überwachung und Steuerung. Dadurch lassen sich Services gütegesichert sowie kosten- und leistungsoptimiert betreiben. Angebote, Anbieter und Ausführungsumgebungen gilt es aufeinander abzustimmen und bedarfsgerecht einzusetzen.

Die Abbildung 6.17 gibt einen Überblick über die Funktionen und ihr Zusammenwirken. Es handelt sich um:

Provider- und Infrastrukturmanagement Das Provider- und Infrastrukturmanagement umfasst die Verwaltung der Anbieter, der Serviceangebote und der Infrastruktur in Form von Storage-, Netzwerk- und Compute-Services. Des Weiteren zählen die Bereitstellung und Beendigung fremdbezogener Services sowie die von virtuellen Maschinen und Containern zu seinen Aufgaben.

Monitoring Die kontinuierliche Überwachung des Umgebungszustands ist eine Grundfunktion des Managements. Messwerte und Kennzahlen gehen unmittelbar in das Reporting und die Überwachung der Servicegütevereinbarungen ein. Das Monitoring übernimmt innerhalb der Kontrollschleife die Sensor-Funktionen und leitet die Leistungs- und Statusinformationen in den Regler, der bei Abweichung die Umgebung wieder zurück in einen erwünschten Zustand überführt.

Konfiguration Zwecks Servicebereitstellung und -betrieb muss der Service-Provider den Applikationsstapel eines Node aus der Ferne kontrollieren und beeinflussen können. Das Infrastrukturmanagement erstellt Nodes, das Konfigurationsmanagement installiert und konfiguriert Softwarekomponenten. Grundsätzlich besteht die Möglichkeit, das Infrastrukturmanagement wiederum über das Konfigurationsmanagement zu steuern. Dafür erhält es als Eingabe eine deklarative Beschreibung des Sollzustands der Umgebung. Anschließend werden auf Grundlage des Ist-Zustands die zur Zustandsüberführung notwendigen Aktionen abgeleitet und angewendet. Die Operationen des Konfigurationsmanagements müssen idempotent sein, d. h. die mehrmalige Anwendung derselben Konfiguration darf keine Auswirkung auf den Umgebungszustand haben, insofern der Zielzustand bereits vorliegt. Für die Kontrolle einer Multi-Provider-Umgebung nimmt das Konfigurationsmanagement einen zentralen Stellenwert ein, ermöglicht es doch steuernd auf die Infrastruktur-, Plattform- und Anwendungsschicht einzuwirken. Das Konfigurationsmanagement erbringt innerhalb der Kontrollschleife die Aktor-Funktionen.

Autonomes Management Kontrolliert wird die Multi-Provider-Umgebung von autonomen Managern. Sie übernehmen die Steuerungsaufgaben und stoßen Anpassungen an. Grundlage dafür sind die vom Monitoring erfassten Leistungs- und Statusinformationen. Die Steuerungslogik wird nicht direkt von den Managern implementiert, sondern ist in ausführbare Policys ausgelagert, die die übergeordneten Managementfunktionen erbringen. Ein Manager ist generisch und stellt einen Ausführungsrahmen zur Verfügung, der unter anderem eine Schnittstelle zum Abfragen und Anpassen der Managementobjekte anbietet. Die Bereitstellung der Policys erfolgt in Form versionierter Softwareartefakte. Dadurch lassen sich Fehlerkorrekturen und Erweiterungen bündeln und gezielt zur Anwendung bringen. Typischerweise ist eine Vielzahl von Policys für die Aufrechterhaltung systemweiter Eigenschaften verantwortlich. Dieser Grundgedanke folgt dem Prinzip der Trennung von Zuständigkeiten (engl. *Separation of Concerns* , SoC) zur Verbesserung der Verständlichkeit, Wartbarkeit und Austauschbarkeit. Dementsprechend sollten anstelle von monolithischen, bevorzugt kleinere, flexiblere Policys zum Einsatz kommen, die eng miteinander kooperieren und die über klar voneinander abgegrenzte Verantwortungsbereiche verfügen.

FCAPS-Management Gemäß dem FCAPS-Modell lassen die Managementaufgaben den fünf Funktionsbereichen Fehler-, Konfigurations,- Abrechnungs-, Leistungs- und Sicherheits-

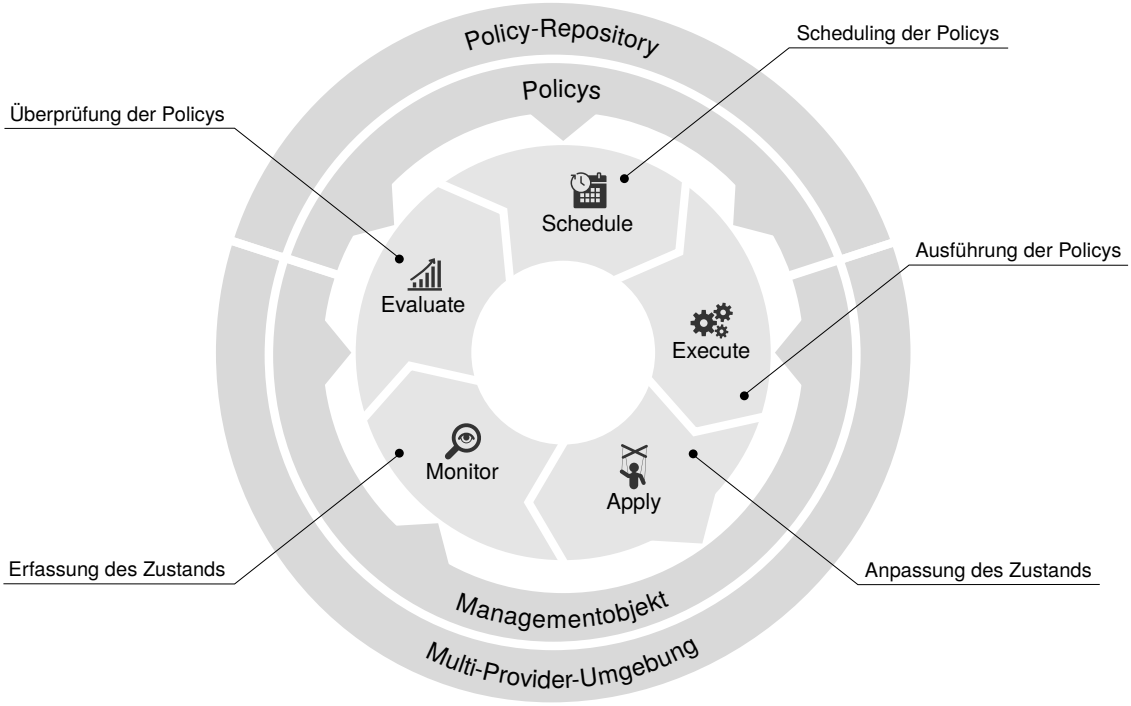


Abbildung 6.18: MESEA-Kontrollschleife

management zuordnen. Sie setzen auf den Grundfunktionen zur Überwachung und Konfiguration auf. Neben Policys lassen sich zudem Managementanwendungen einsetzen, die gemeinsam einen unterbrechungsfreien und leistungsgerechten Betrieb gewährleisten sollen. Dafür gilt es, die Servicegüte kontinuierlich zu überwachen und gegebenenfalls durch korrigierende Anpassungen wiederherzustellen. Ansonsten können gemäß der Servicegütevereinbarungen Vertragsstrafen geltend gemacht werden. Um diese bereits im Vorfeld abzuwenden, lassen sich adaptive, korrektive und perfektive Maßnahmen einsetzen. Adaptive Maßnahmen umfassen Anpassungen an veränderte Anforderungen und Rahmenbedingungen, korrektive Maßnahmen überführen die Umgebung zurück in einen fehlerfreien Zustand und perfektive Maßnahmen dienen der Optimierung des Betriebs. Dabei kann es sich um den Austausch eines aus Sicht des Service-Providers besser geeigneten Service handeln, der kostengünstiger, leistungsfähiger oder zuverlässiger ist. Entscheidungsgrundlage bilden die von den Anbietern garantierten qualitativen Eigenschaften der Services.

6.5.1 MESEA-Kontrollschleife

Wie sich bereits der Abbildung 6.17 entnehmen lässt, bilden Monitoring, Konfiguration und autonomes Management eine Kontrollschleife mit Rückkopplung. Die im Rahmen der IBM-Autonomic-Computing-Initiative vorgestellte MAPE-K-Kontrollschleife dient als Ausgangspunkt für die Entwicklung einer spezialisierten Kontrollschleife für das Management einer Multi-Provider-Umgebung. Der als *MESEA-Kontrollschleife* bezeichnete Regelkreis ist in der Abbildung 6.18 dargestellt und besteht aus den folgenden fünf Phasen:

Monitor Die Grundlage der Monitor-Phase bilden die seitens des Informationsmodells festgelegten Managementobjekte, insbesondere die durch Status- und Konfigurationsvariablen

abgebildeten Zustands- und Stellgrößen. Innerhalb der Phase werden die Zustände der Managementobjekte aus der Datenbasis abgerufen. Sie umfasst neben statistischen auch dynamische Managementobjekte mit geringer sowie hoher Änderungsfrequenz. Von zentraler Bedeutung ist die Fähigkeit einer ereignisbasierten Benachrichtigung bei Struktur- und Wertänderungen. Dies erlaubt es, zeitnah auf Zustandsänderungen zu reagieren, um je nach Erfordernis Anpassungen einzuleiten. Aufgabe der Phase ist die Feststellung einer Differenz zweier aufeinanderfolgender Zustände. Ob dafür eine Pull- oder Push-basierte Strategie zum Einsatz kommt, hängt von den technischen Fähigkeiten der Datenbasis ab. Die so ermittelte Differenz wird durch ein Änderungsereignis beschrieben, das den Ausgangspunkt der nachfolgenden Phase bildet.

Evaluate Das Policy-Repository enthält die für das Management erforderlichen Policys. Innerhalb der Evaluate-Phase wird daraus eine Teilmenge ermittelt, die diejenigen Policys enthält, die eine Ausführung erfordern. Ist die Menge leer, terminiert die Kontrollschleife vorzeitig.

Schedule In der Schedule-Phase erfolgt die Festlegung der Ausführungsreihenfolge. Dazu werden die Policys gemäß ihrer Prioritätsangabe sortiert. Für Policys gleicher Priorität existiert keine eindeutige Ordnung, und eine nebenläufige Ausführung ist möglich. Der auf diese Weise erstellte Ausführungsplan dient der nächsten Phase als Eingabe.

Execute Die Ausführung der Policys erfolgt in der Execute-Phase, in der alle erforderlichen Änderungen an den Managementobjekten berechnet werden. Dazu können Policys zusätzliche Zustandsinformationen aus der Datenbasis abrufen. Sind alle Policys terminiert, wird das Ergebnis zwecks Anwendung an die nächste Phase übergeben.

Apply Abschließend überführt das Konfigurationsmanagement in der Apply-Phase das Gesamtsystem in den gewünschten Zielzustand. Dazu erfolgt eine Anpassung der Managementobjekte, wodurch eine Anpassung der Multi-Provider-Umgebung angestoßen wird. Der veränderte Zustand spiegelt sich erneut in der Datenbasis wider und führt zu Struktur- und Wertänderungen. Diese werden vom Monitoring erfasst und können einen weiteren Schleifendurchlauf zur Folge haben.

Die in der MAPE-K-Kontrollschleife enthaltene Wissensbasis (Knowledge) lässt sich im Rahmen der MESEA-Kontrollschleife verfeinern. Zunächst sind zur Entwicklungszeit folgende Kenntnisse erforderlich:

- Managementobjekte des Informationsmodells
- Korrektive und perfektive Adaptionsstrategien und -maßnahmen

Zur Laufzeit bilden die Managementobjekte mit Hilfe von Status- und Konfigurationsvariablen das Wissen über den Ist-Zustand der Multi-Provider-Umgebung ab. Die im Policy-Repository hinterlegten Policys basieren auf dem Wissen, welche Adaptionsmaßnahmen sich zur Beseitigung von Fehlern und Beeinträchtigungen einsetzen lassen. Die Managementlogik erfordert spezielle Kenntnisse über die Ursache-/Wirkungszusammenhänge. Innerhalb der Evaluate-Phase analysieren und bewerten Policys den aktuellen Systemzustand. Wird dabei eine als unerwünscht eingestufte Situation festgestellt, überführen sie das System als Reaktion darauf wieder in einen erwünschten bzw. als besser geltenden Zustand.

Unterschied zur MAPE-K-Kontrollschleife

Der zentrale Unterschied zwischen der MAPE-K- und der MESEA-Kontrollschleife ist der Einsatz der Apply-Phase. Diese dient dazu, die Multi-Provider-Umgebung effizient in einen Nachfolgezustand zu überführen. Innerhalb der Execute-Phase bestimmen die Policys, welche Änderungen an den Managementobjekten erforderlich sind. Erst nach ihrer Terminierung erfolgt die Anpassung der Managementobjekte. Würde unmittelbar nach Terminierung einer einzelnen Policy eine umgebungsübergreifende Rekonfiguration angestoßen und geschähe dies mehrfach innerhalb eines Schleifendurchlaufs, wäre ein hoher Berechnungs- und Kommunikationsaufwand die Folge. Für jeden Node müsste seine Konfiguration berechnet, übermittelt und angewendet werden. Durch Verschiebung der Rekonfiguration in die Apply-Phase lässt sich dieser Aufwand reduzieren, indem die Multi-Provider-Umgebung innerhalb eines Schleifendurchlaufs höchstens eine einmalige Anpassung erfährt.

6.5.2 Funktionales Rollenmodell

Die Managementfunktionen gilt es technischen Funktionseinheiten zuzuteilen. Ihre Definition erfolgt mit Hilfe funktionaler Rollen. Eine *funktionale Rolle* spezifiziert die Ausprägung eines Applikationsstapels. Die Zuweisung an einen Node erfolgt bei seiner Klassifikation. Dies veranlasst das Konfigurationsmanagement, den Applikationsstapel in den durch die funktionale Rolle vorgegebenen Zielzustand zu überführen. Dadurch werden Laufzeitumgebungen und Middleware aufgesetzt, Softwarekomponenten zur Ausführung gebracht und Services bereitgestellt. Die funktionale Rolle beschreibt die Funktion eines Node innerhalb des Anwendungsszenarios. Dementsprechend wird eine Bezeichnung verwendet, die zwar Rückschlüsse auf den Verwendungszweck zulässt, aber keinen direkten Bezug zur technischen Realisierung aufweist. Es bestehen somit Freiheitsgrade bei der Umsetzung. Mit Hilfe funktionaler Rollen lassen sich sowohl Management- wie auch Anwendungsfunktionen bereitstellen. Das Management einer Multi-Provider-Umgebung erfordert den Einsatz mehrerer funktionaler Rollen. Ihre Überwachung und Konfiguration setzt wiederum die Repräsentation durch Managementobjekte voraus. Die Abbildung 6.19 zeigt die dazugehörige Vererbungshierarchie. Das funktionale Rollenmodell umfasst die folgenden Managementobjekte:

Controller Der *Controller* erlaubt es, auf Nodes Konfigurationsläufe anzustoßen, die eine Überprüfung und Anpassung des Applikationsstapels bewirken. Die Kommunikation erfolgt über die vom Messenger bereitgestellte nachrichtenorientierte Middleware. Der Controller ermöglicht die entfernte Steuerung von Nodes und kommt vorrangig in Umgebungen zum Einsatz, die ein Pull-basiertes Konfigurationsmanagement einsetzen. In diesem Fall ist auf jedem Node ein Agent installiert, der aktiv von einem Server seine Konfiguration anfordert und diese nach Empfang zur Anwendung bringt.

Manager Bei einem *Manager* handelt es sich um eine abstrakte Rolle, die von Nodes wahrgenommen wird, denen die Verwaltung der Managementobjekte oder die Gewährleistung der Servicegüteanforderungen unterliegt.

Autonomer Manager Der *autonome Manager* ist eine Rolle, die die Funktionen des Objektmanagers und die des Servicemanagers gemeinsam auf demselben Node erbringt.

Objektmanager Der *Objektmanager* ist verantwortlich für Bereitstellung und Anpassung der Managementobjekte. Die Anbindung der Ressourcen stellen Managementobjektprovider

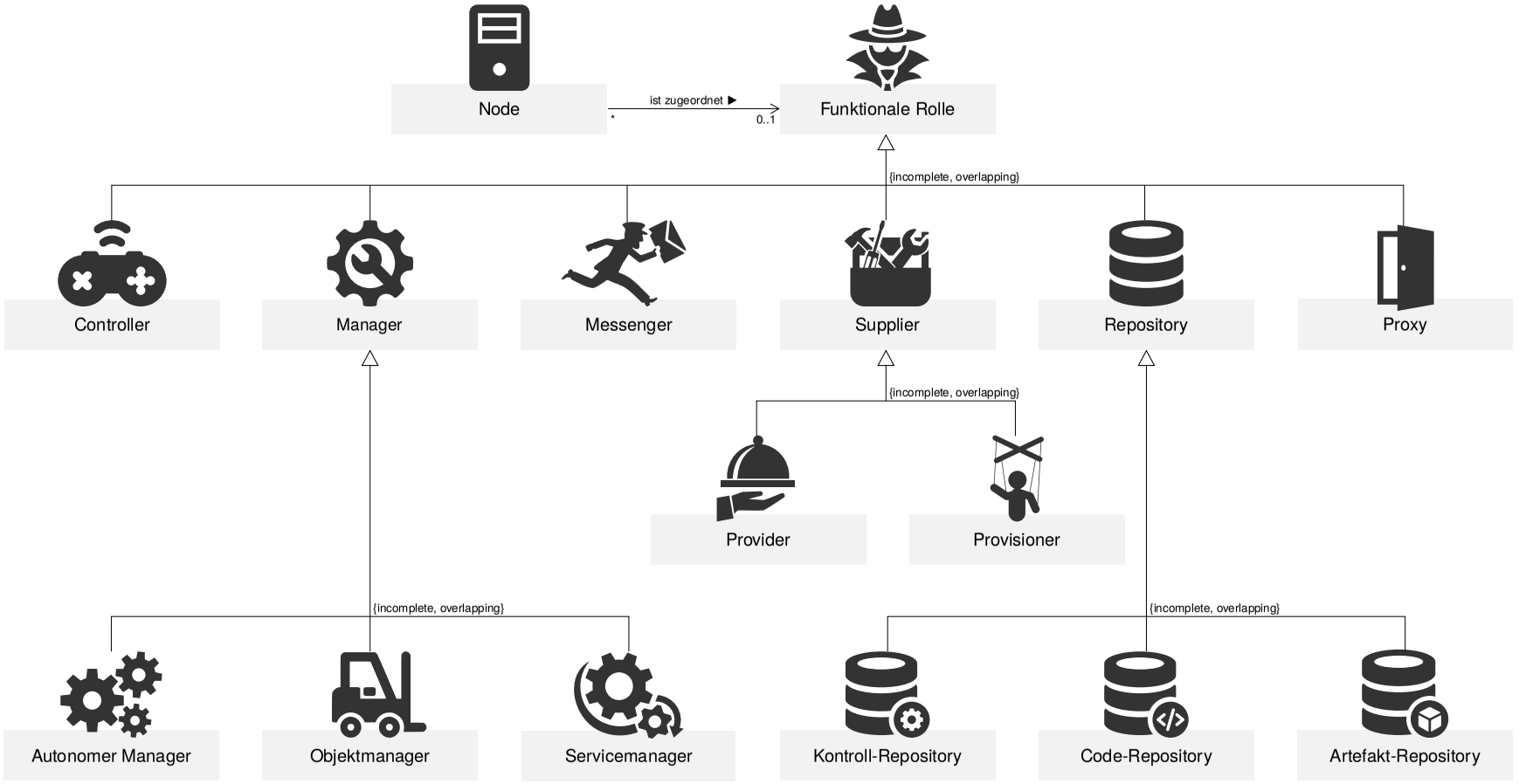


Abbildung 6.19: Managementobjekte zur Repräsentation funktionaler Rollen

her. Ihre Aufgabe besteht darin, Konfigurationswerte auf Ressourcen anzuwenden und umgekehrt den Zustand der Managementobjekte an den der Ressourcen anzupassen.

Servicemanager Die Rolle des *Servicemanagers* wird von einem Node übernommen, der die Funktion des autonomen Managements auf Grundlage der MESEA-Kontrollschleife erbringt. Der Servicemanager soll gewährleisten, dass die Servicegütevereinbarungen eingehalten sind. Zu diesem Zweck kommen Policys zum Einsatz, die die Überwachungs- und Steuerungslogik enthalten. Policys lassen sich zur Laufzeit installieren, deinstallieren, aktivieren, deaktivieren sowie rekonfigurieren. Somit lässt sich auch während des Betriebs die Managementlogik an veränderte Anforderungen und Rahmenbedingungen anpassen.

Messenger Ein *Messenger* stellt eine nachrichtenorientierte Middleware in Form eines Message-Bus zur Verfügung, über den Nodes und Controller untereinander Nachrichten austauschen.

Supplier Bei einem *Supplier* handelt es sich um eine abstrakte Rolle, die von Nodes eingenommen wird, die dem Servicemanager unterstützende Funktionen bereitstellen. Während der Servicemanager aktiv Entscheidungen trifft, übernehmen Supplier eine passive Rolle. Ihre Aufgabe besteht darin, die seitens des Managements festgelegten Aktionen auszuführen.

Provider Ein *Provider* ist ein Supplier, dessen Einsatzgebiet das Provider- und Infrastrukturmanagement umfasst. Er stellt unter anderem Operationen zur Verfügung, mittels derer die Infrastruktur anbieterübergreifend kontrolliert und gesteuert werden kann. Dies erfordert spezifische Client-Implementierungen, über die sich die unterschiedlichen Ausführungsumgebungen anbinden und integrieren lassen.

Provisioner Ein *Provisioner* stellt neben dem Provider einen weiteren Supplier dar. Sein Einsatzgebiet ist das Konfigurationsmanagement. Ihm obliegt die Verwaltung und Kontrolle der Applikationsstapel. Ein Node untersteht unmittelbar nach Erstellung durch den Provider der Verantwortung des Provisioner, der seinen Applikationsstapel gemäß Rollendefinition aufsetzt und konfiguriert. Innerhalb des Aufsetzungsprozesses finden sowohl die rollen- als auch die nodespezifischen Konfigurationen Berücksichtigung. Rollenspezifische Konfigurationen werden zur Entwicklungszeit festgelegt und finden bei allen Nodes Anwendung, denen die entsprechende funktionale Rolle zugeordnet ist. Nodespezifische Konfigurationen werden hingegen zur Laufzeit dynamisch erstellt und gelten ausschließlich für den jeweiligen Node.

Repository Bei einem *Repository* handelt es sich um eine abstrakte Rolle, die von einem Node übernommen wird, der eine zentrale Ablage für Daten, Dokumente, Objekte und Softwareartefakte bietet. Die Inhalte sind mit zusätzlichen Metadaten versehen, die das Auffinden von Informationen erleichtern. Der Zugriff ist ausschließlich autorisierten Nutzern vorbehalten. Repositorys kommen im Dokumentenmanagement, im Contentmanagement und in der Versionsverwaltung zum Einsatz.

Kontroll-Repository Das *Kontroll-Repository* verwendet ein verteiltes Versionskontrollsystem zur Speicherung und Verwaltung der projektseitigen Konfigurationen. Es dient als Ergänzung zum Code-Repository und ermöglicht eine projektabhängige Parametrisierung der Aufsetzungsprozesse. Die Umgebungsdefinition liegt als deklarative Beschreibung vor und ist Bestandteil der Konfigurationsdaten. Das Zurücksetzen des Kontroll-Repository auf einen früheren Versionsstand bewirkt gleichzeitig ein Zurücksetzen der Umgebung in

einen Vorgängerzustand. Des Weiteren unterstützt das Kontroll-Repository die Bildung von Node-Gruppen. Diese verfügen jeweils über eigene Konfigurationssätze, die sich unabhängig voneinander verwalten, kontrollieren und anpassen lassen. Die Klassifikation eines Nodes erfordert neben der Zuweisung einer funktionalen Rolle auch die Zuordnung zu einer dieser Gruppen.

Code-Repository Das *Code-Repository* enthält diejenigen Softwareartefakte, die für die Aufsetzungsprozesse erforderlich sind. Es erlaubt eine Trennung des Aufsetzungscodes von den projektspezifischen Konfigurationsdaten, die das Kontroll-Repository bereitstellt. Auf diese Weise lässt sich das Code-Repository projektübergreifend einsetzen, wohingegen jedes Projekt aufgrund spezieller Konfigurationserfordernisse ein individuelles Kontroll-Repository erfordert.

Artefakt-Repository Das *Artefakt-Repository* bietet eine zentrale Ablage für Softwareartefakte, die als Teil des Applikationsstapels auf einem Node installiert und gemäß Vorgabe konfiguriert werden. Innerhalb des Aufsetzungsprozesses werden vom Repository fehlende Softwareartefakte angefordert und lokal hinterlegt.

Proxy Ein *Proxy* ist ein Vermittler in einem Kommunikationsnetzwerk. Er nimmt auf der einen Seite Anfragen entgegen und richtet diese auf der anderen Seite stellvertretend für die Clients an die Ziel-Server. Für das Management einer Multi-Provider-Umgebung ist aber nicht die Vermittlerfunktion der vorrangige Verwendungszweck, sondern vielmehr die Möglichkeit Softwareartefakte zu cachen. Wiederkehrende Anfragen werden durch Zwischenspeicherung schneller abgearbeitet. Dadurch lässt sich das Aufsetzen des Applikationsstapels auf einem Node beschleunigen. Softwareartefakte müssen mitunter nicht mehr aus einem entfernten Repository über das Internet nachgeladen werden. Aufgrund vorheriger Anfragen können sie bereits im Proxy vorliegen, so dass eine direkte Auslieferung über das lokale Kommunikationsnetzwerk möglich ist.

6.5.3 Projektmanagement

Aufbauend auf dem funktionalen Rollenmodell wird ein Architekturmuster zur anbieterseitigen Projektorganisation entworfen. Ziel ist die automatisierte Bereitstellung von Services sowie der gütegesicherte Betrieb auf Grundlage vertraglicher Vereinbarungen. Dies setzt die Beherrschbarkeit einer Multi-Provider-Umgebung voraus und erfordert eine einheitliche Projektüberwachung und -steuerung. Dadurch lassen sich Services flexibel und bedarfsgerecht innerhalb der Ausführungsumgebungen erbringen. Zu diesem Zweck muss das technische Management nach dem Zero-Touch-Prinzip [AK15] arbeiten. Hierbei gilt es, einen möglichst hohen Automatisierungsgrad zu erreichen, der weitestgehend ohne manuelle Eingriffe auskommt.

6.5.3.1 Organisatorische Projektstruktur

Das funktionale Rollenmodell definiert die für einen automatisierten und gütegesicherten Servicebetrieb erforderlichen Funktionseinheiten. Ihr Zusammenwirken wird durch die organisatorische Projektstruktur festgelegt. Diese lässt sich der Abbildung 6.20 entnehmen. Sie zeigt eine einstufige Projekthierarchie, bestehend aus einem zentralen *Management-Projekt* sowie untergeordneten *Service-Projekten*. Beide Projekte besitzen eine ähnliche Struktur und sind wie folgt aufgebaut:

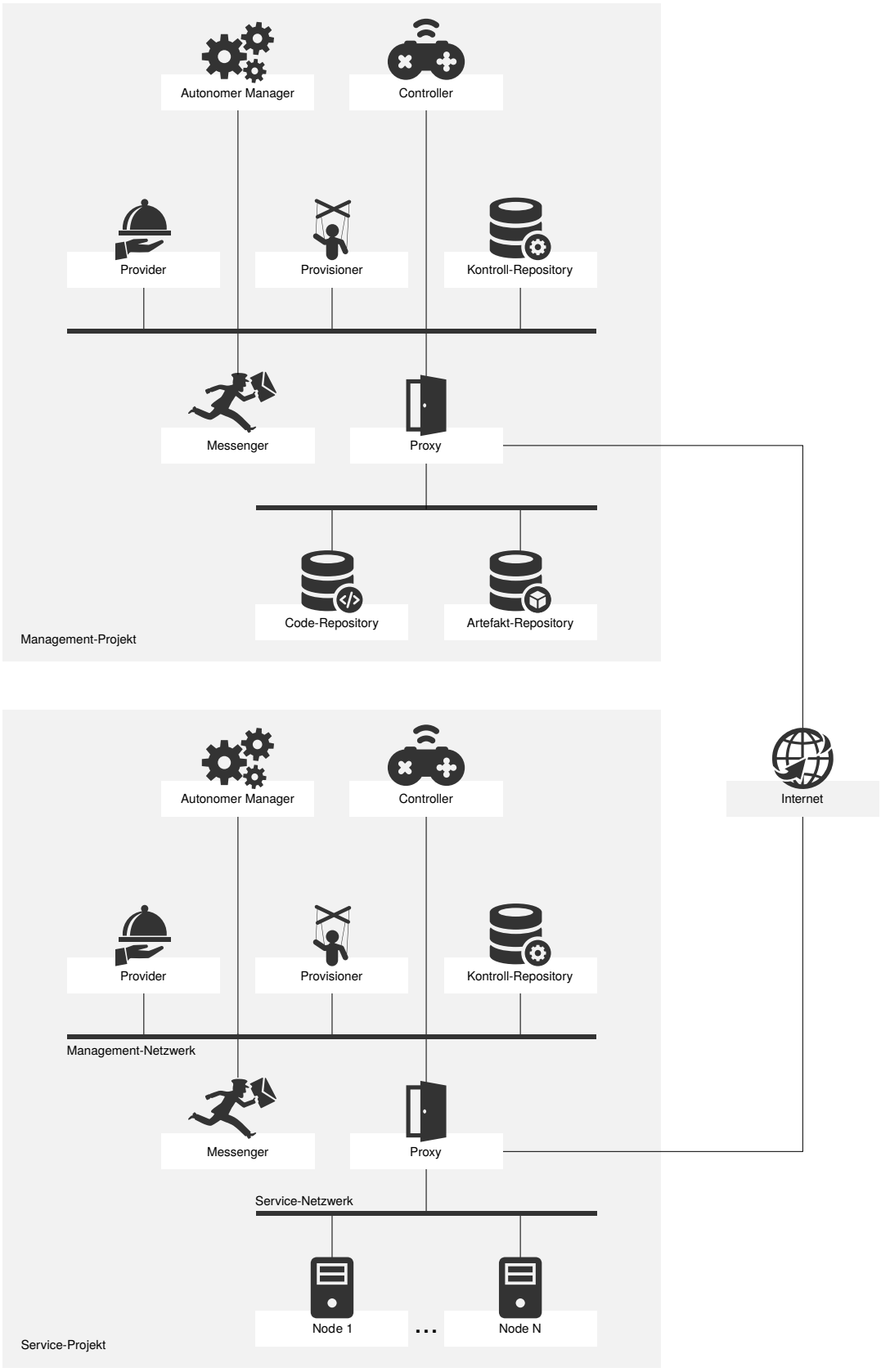


Abbildung 6.20: Projekthierarchie und organisatorische Struktur

Service-Projekt Innerhalb eines Service-Projekts kommen zwei geroutete Netzwerke zum Einsatz. Das *Management-Netzwerk* umfasst sechs Nodes, denen die Rollen des autonomen Managers, des Providers, des Provisioner, des Messenger, des Controllers und des Kontroll-Repository zugeordnet sind. Der Provider erstellt auf Anforderung durch den autonomen Manager Nodes und verbindet sie mit dem *Service-Netzwerk*. Im Anschluss setzt der Provisioner gemäß der vom Kontroll-Repository bereitgestellten Daten den Node auf und nimmt die notwendigen Konfigurationen vor. Das Kontroll-Repository verfügt über eine Floating IP und ist über das Internet erreichbar. Je nach Erfordernis kann das Service-Netzwerk in weitere Teilnetze zerlegt werden. Es muss jedoch stets gewährleistet sein, dass sich Daten zwischen Management- und Service-Netzwerk austauschen lassen. Ob hingegen die Kommunikation zwischen den Teilnetzen untereinander gestattet ist, muss der Service-Provider abhängig vom Schutzbedarf und den Zugriffsmöglichkeiten des Anwenders entscheiden.

Management-Projekt Bei einem *Management-Projekt* handelt es sich um ein dediziertes Service-Projekt, in dem der Service-Provider zentrale Services erbringt, die allen Service-Projekten zugänglich sind. Das Service-Netzwerk umfasst zwei Nodes, die die Rolle des Code- und Artefakt-Repository einnehmen. Jedes Repository verfügt über eine Floating IP. Das Code-Repository enthält die Code-Basis für die Aufsetzungsprozesse, das Artefakt-Repository die Softwareartefakte.

6.5.3.2 Interaktions- und Kommunikationsbeziehungen

Erst das Zusammenwirken der Funktionseinheiten ermöglicht die automatisierte Bereitstellung und den gütegesicherten Betrieb von Services in einer Multi-Provider-Umgebung. Zwischen den Funktionseinheiten bestehen Abhängigkeiten in Form von Interaktions- und Kommunikationsbeziehungen. Zu ihrer Veranschaulichung wird die in der Abbildung 6.20 skizzierte Projektstruktur um die logischen Nachrichtenflüsse ergänzt. Exemplarisch sei ein hierarchisch organisiertes Management angenommen. Das Ergebnis ist in der Abbildung 6.21 dargestellt. Sie zeigt einen aus Sicht des Service-Providers typischen Anwendungsfall. Dieser umfasst die Node-Erstellung in einem Service-Projekt sowie die Installation und Konfiguration des Applikationsstapels. Der Vorgang besteht aus mehreren Einzelschritten, die automatisiert ausgeführt werden. Der Ablauf ist wie folgt:

1. Die Servicebereitstellung wird von einem Service-Consumer angestoßen. Seine Anfrage ist an den autonomen Manager des Management-Projekts gerichtet. Dieser übernimmt die Koordination und Abwicklung. Auf Grundlage der funktionalen und nicht-funktionalen Anforderungen bestimmt er ein Service-Projekt, innerhalb dessen die Anfragebearbeitung erfolgen soll.

2. Der autonome Manager erzeugt eine Konfiguration für den autonomen Manager des Zielprojekts. Er fügt diese über die Managementobjekte in das vom Kontroll-Repository bereitgestellte Versionskontrollsystem ein.

3. Der Provisioner wird vom Kontroll-Repository dazu aufgefordert, eine Rekonfiguration durchzuführen.

4. Dieser gleicht dazu den Zustand seines lokalen Repository mit dem des entfernten Kontroll-Repository ab. Fehlende Aufsetzungsmodule werden aus dem zentralen Code-Repository

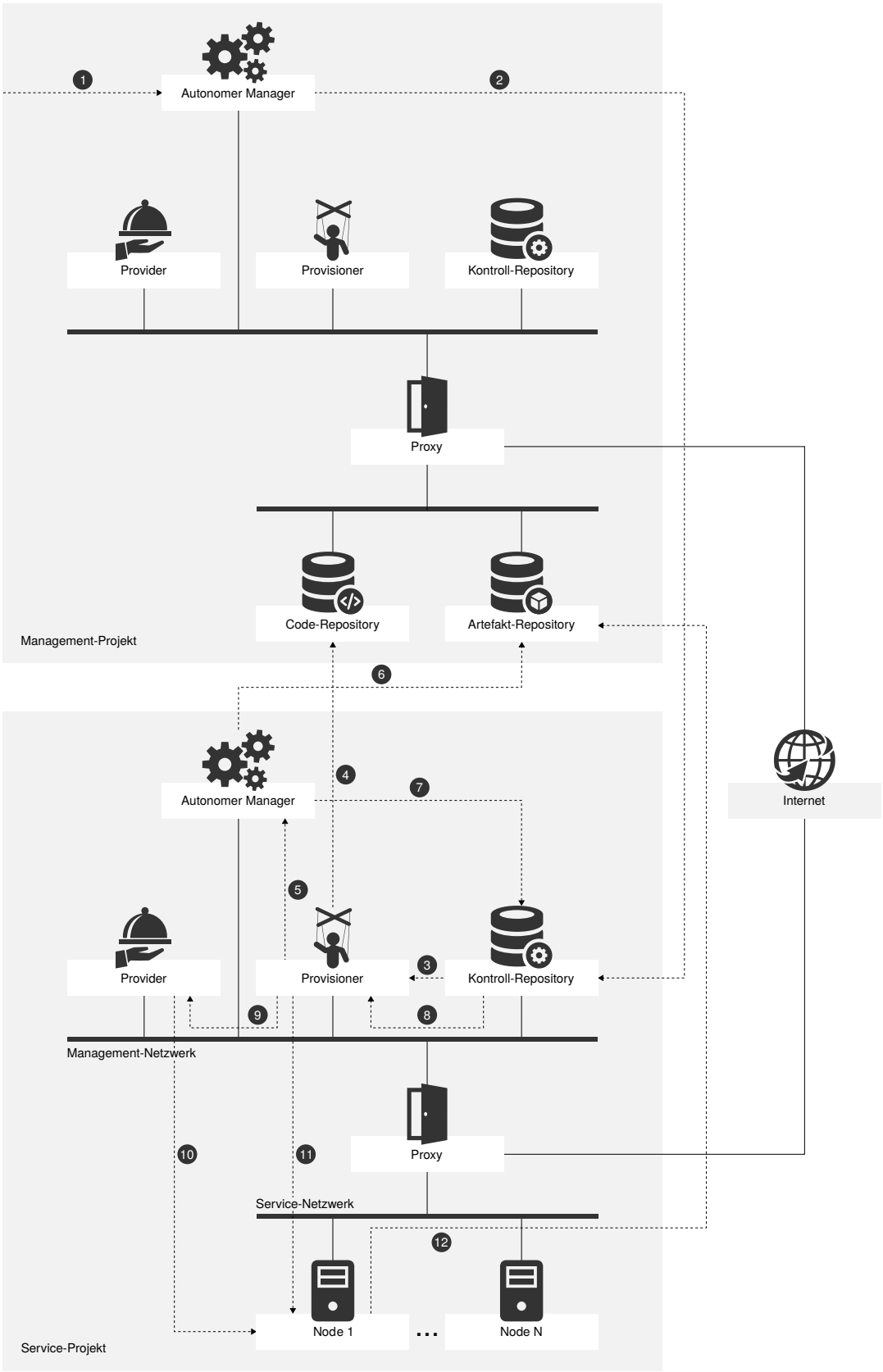


Abbildung 6.21: Interaktions- und Kommunikationsbeziehungen

nachgeladen. Im Anschluss werden die Nodes über den Controller dazu aufgefordert, einen Konfigurationslauf durchzuführen.

5. Der autonome Manager erhält vom Provisioner eine angepasste Konfiguration und den Code für die Anwendung.

6. Innerhalb des Konfigurationslaufs werden fehlende Softwareartefakte aus dem zentralen Artefakt-Repository nachgeladen und anschließend installiert. Dadurch erhält der autonome Manager die erforderliche Managementlogik in Form ausführbarer Überwachungs- und Steuerungs-Policys.

7. Basierend auf dem Projektzustand und der Anfrage werden Konfigurationen angepasst und über Managementobjekte in das Kontroll-Repository eingefügt. Im vorliegenden Fall wird die Konfiguration des Providers um eine zusätzliche Node-Definition ergänzt. Es sei angemerkt, dass nicht mit jeder Servicebereitstellung auch eine Node-Erstellung einhergeht. Es ist Bestandteil der Managementlogik zu entscheiden, in welchen Situationen Nodes rekonfiguiert oder neu erstellt werden.

8. Der Provisioner erhält vom Kontroll-Repository die Aufforderung, eine Rekonfiguration durchzuführen.

9. Er bestimmt daraufhin diejenigen Nodes, die von der Änderung betroffen sind und fordert sie über den Controller auf, einen Konfigurationslauf durchzuführen.

10. Auf diese Weise erhält der Provider die angepasste Konfiguration. Nach Abgleich des Ist-Zustands mit dem Sollzustand veranlasst er über die Managementschnittstelle der Ausführungsumgebung die Bereitstellung eines zusätzlichen Node. Im Zuge dessen wird ihm auch eine funktionale Rolle zugewiesen.

11. Im Anschluss erhält der Node vom Provisoner gemäß Rollendefinition seine Konfiguration.

12. Sollten beim Aufsetzen des Applikationsstapels Softwareartefakte noch nicht im Proxy vorliegen, werden diese automatisch aus dem zentralen Artefakt-Repository nachgeladen. Nach Abschluss verfügt der Node über die von der funktionalen Rolle vorgegebenen Softwarekomponenten, Services und Konfigurationen.

6.6 Policys

Die Steuerung der Projekte erfolgt mit Hilfe von Policys, die die Überwachungs- und Steuerungslogik enthalten. Sie werden von einem Servicemanager verwaltet und ausgeführt. Grundlage bilden die vom Objektmanager bereitgestellten Managementobjekte. In diesem Zusammenhang werden auch Policys durch Managementobjekte repräsentiert. Dadurch lassen sich wiederum Policys einsetzen, die die Steuerungslogik selbst überwachen und die gegebenenfalls als Reaktion auf ein unerwartetes oder unerwünschtes Verhalten Anpassungen vornehmen. Wird beispielsweise festgestellt, dass eine Ausführung nicht terminiert oder eine Zeitüberschreitung vorliegt, kann die betroffene Policy automatisch deaktiviert, neu gestartet oder rekonfiguriert werden. Dadurch verfügt das Managementsystem über grundlegende Fähigkeiten zur Selbstüberwachung und Selbstkonfiguration, auf deren Basis sich weitere Self-X-Eigenschaften wie Selbstheilung, Selbstoptimierung und Selbstschutz realisieren lassen. Die Abbildung 6.22 zeigt die Managementobjekte zur Repräsentation der Policys. Es handelt sich um:

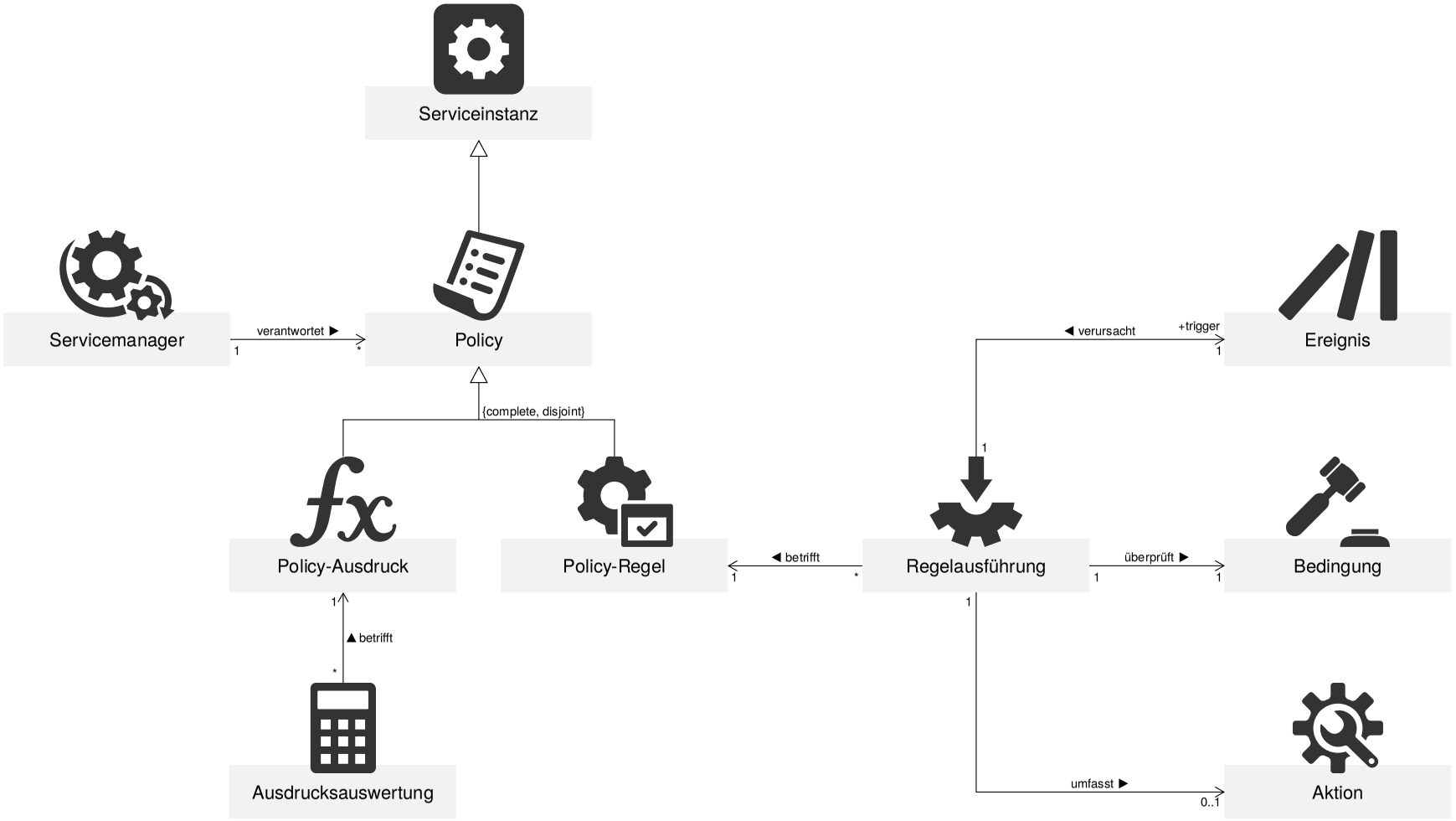


Abbildung 6.22: Managementobjekte zur Repräsentation von Policys

Policy Eine *Policy* stellt einen Policy-Ausdruck oder eine Policy-Regel dar. Policys unterliegen der Verantwortung genau eines Servicemanagers.

Policy-Ausdruck Ein *Policy-Ausdruck* repräsentiert eine Berechnungsfunktion, die über Status- und Konfigurationsvariablen von Managementobjekten formuliert ist. Ihre Auswertung kann zur Laufzeit angefordert werden. Zu diesem Zweck stellt der Servicemanager eine entsprechende Schnittstelle zur Verfügung. Policy-Ausdrücke dienen der Beurteilung des Umgebungszustands und vermeiden in der anfragenden Softwarekomponente Abhängigkeiten zur Ausführungsumgebung und zu den Managementobjekten. Sie können dadurch wiederverwendet und an andere Anwendungsszenarien angepasst werden.

Ausdrucksauswertung Eine *Ausdrucksauswertung* beschreibt das Ergebnis eines Policy-Ausdrucks. Das Managementobjekt repräsentiert eine Anforderung zur Ausdrucksauswertung. Ihre Repräsentation durch ein eigenes Managementobjekt hat den Vorteil, dass sich auch nebenläufige Auswertungsanfragen desselben Policy-Ausdrucks erfassen lassen. Das Managementobjekt stellt folgende Zustandsgrößen bereit:

- Status
- Start- und Endzeitpunkt
- Identifikator des Aufrufers
- Ergebnis des Policy-Ausdrucks, ggfs. Fehlerbeschreibung

Policy-Regel Eine *Policy-Regel* repräsentiert eine Adaptionsmaßnahme, die dem ECA-Prinzip (*Event-Condition-Action*) folgt. *Ereignisse* beschreiben Veränderungen in der Datenbasis und werden ausgelöst durch Hinzufügen, Löschen und Ändern von Managementobjekten. Dies stößt die Auswertung einer *Bedingung* an. Im positiven Fall folgt die Ausführung der *Aktion*. Dabei können Managementobjekte hinzugefügt, gelöscht oder verändert werden. Jedwede andere Art der Reaktion ist nicht gestattet. Policy-Regeln können zudem die Auswertung von Policy-Ausdrücken anfordern. Dies vermeidet redundanten Code innerhalb der Regelimplementierung und sorgt für klar voneinander abgegrenzte Zuständigkeiten.

Regelausführung Eine *Regelausführung* bildet die Ausführung einer Policy-Regel ab. In Übereinstimmung mit dem ECA-Prinzip werden die einzelnen Phasen durch Managementobjekte beschrieben. Die Repräsentation einer Regelausführung durch ein eigenes Managementobjekt erlaubt es, ähnlich wie bei Policy-Ausdrücken, nebenläufige Ausführungen derselben Policy-Regel zu erfassen. Das Managementobjekt stellt folgende Zustandsgrößen bereit:

- Status
- Start- und Endzeitpunkt

Ereignis Ein *Ereignis* beschreibt eine Struktur- oder Wertänderung innerhalb der Datenbasis. Die Änderung bezieht sich auf ein Managementobjekt, das eine Ausführungsumgebung oder ein Projekt repräsentiert. Das Ereignis stößt die Ausführung einer Policy-Regel an, vorausgesetzt ihre Filter veranlassen den Servicemanager nicht dazu, das Ereignis zu verwerfen. Zur Repräsentation der Regelausführung wird in der Datenbasis ein Managementobjekt erstellt und mit dem auslösenden Ereignis assoziiert. Dieses stellt folgende Zustandsgrößen zur Verfügung:

- Ereignistyp (Hinzufügen, Löschen, Ändern)
- Quelle
- Bezeichnung und Wert der Managementvariable (optional)

Bedingung Die *Bedingung* repräsentiert die Auswertung der Regelbedingung. Grundlage dafür sind das Ereignis selbst sowie die von Managementobjekten bereitgestellten Status- und Konfigurationsvariablen, auf die ausschließlich lesend zugegriffen werden darf. Das Managementobjekt stellt folgende Zustandsgrößen bereit:

- Status
- Start- und Endzeitpunkt
- Ergebnis der Bedingungsauswertung, ggfs. Fehlerbeschreibung

Aktion Die *Aktion* repräsentiert die Durchführung einer Adaptionsmaßnahme, die innerhalb der Regelaktion implementiert ist. Das Managementobjekt stellt folgende Zustandsgrößen bereit:

- Status
- Start- und Endzeitpunkt
- Protokollierung der Objektänderungen, ggfs. Fehlerbeschreibung

Policys sind wiederum Services und können Bindungen sowie qualitative Eigenschaften besitzen, die sich mit Hilfe von Servicegüteparametern erfassen lassen. Zur Formulierung von Zielvorgaben stehen Service-Level-Objectives zur Verfügung, die sich zu Servicegütevereinbarungen zusammenfassen lassen. Eine Policy unterscheidet sich somit nicht grundsätzlich von anderen Services. Adaptives Systemverhalten darf insgesamt nicht nur auf Services und Komponenten beschränkt sein, die für das kundenseitige Anwendungsszenario bestimmt sind, vielmehr muss auch das Managementsystem selbst Berücksichtigung finden. Letztlich besteht auch dieses aus interagierenden Services, die mit Hilfe von Softwarekomponenten bereitgestellt und auf Nodes betrieben werden. Mitunter lässt sich die Managementlogik nicht gänzlich durch Policys ausdrücken, so dass auch Managementanwendungen und -services zum Einsatz kommen müssen.

6.7 Domänen

Ein Service-Projekt untersteht dem Verantwortungsbereich mindestens eines Objektmanagers. Kommen mehrere Objektmanager zum Einsatz, müssen diese in einer hierarchischen Struktur angeordnet werden. Dies sorgt dafür, dass alle Managementobjekte zentral zugreifbar sind und sich die Multi-Provider-Umgebung durch ein einzelnes zusammenhängendes Managementobjekt repräsentieren lässt. Die Abbildung 6.23 zeigt die Managementobjekte zur Repräsentation von Managementdomänen. Es handelt sich um:

Managementdomäne Eine *Managementdomäne* repräsentiert entweder eine Verwaltungs- oder eine Funktionsdomäne. Eine Managementdomäne entspricht einer logischen Gruppierung von Managementobjekten, deren Zuordnung durch ein Bildungskriterium festgelegt ist, das sich dynamisch anpassen lässt.

Verwaltungsdomäne Die einem Objektmanager unterstellten Managementobjekte lassen sich disjunkten *Verwaltungsdomänen* zuordnen. Dabei muss gewährleistet sein, dass ein

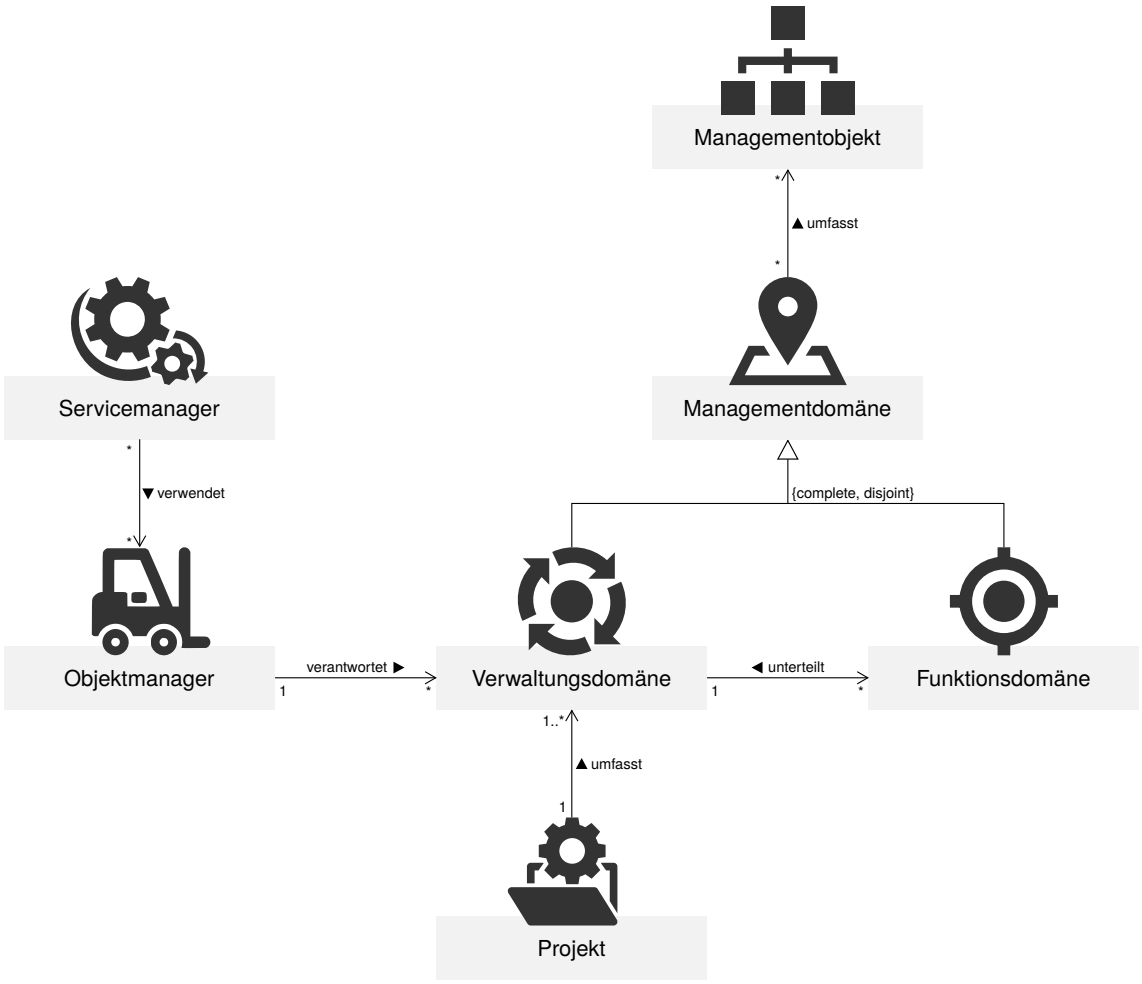


Abbildung 6.23: Managementobjekte zur Repräsentation von Managementdomänen

Managementobjekt genau einem Objektmanager und damit einer Verwaltungsautorität unterstellt ist.

Funktionsdomäne In einer *Funktionsdomäne* werden Managementobjekte nach funktionalen Gesichtspunkten gruppiert. Ausgangspunkt bildet eine Verwaltungsdomäne, die sich mit Hilfe von Funktionsdomänen unterteilen lässt. Eine Verschachtelung von Funktionsdomänen ist möglich. Es müssen aber die Überlappungsfreiheit sowie die Enthaltenseinbeziehung gewahrt bleiben.

Die Abbildung 6.24 zeigt den Einsatz von Funktionsdomänen innerhalb eines Service-Projekts. Die Ressourcen des Service-Projekts bilden eine gemeinsame Verwaltungsdomäne. Diese ist wiederum in die folgenden vier Funktionsdomänen unterteilt:

- Management
- Entwicklung
- Test
- Produktion

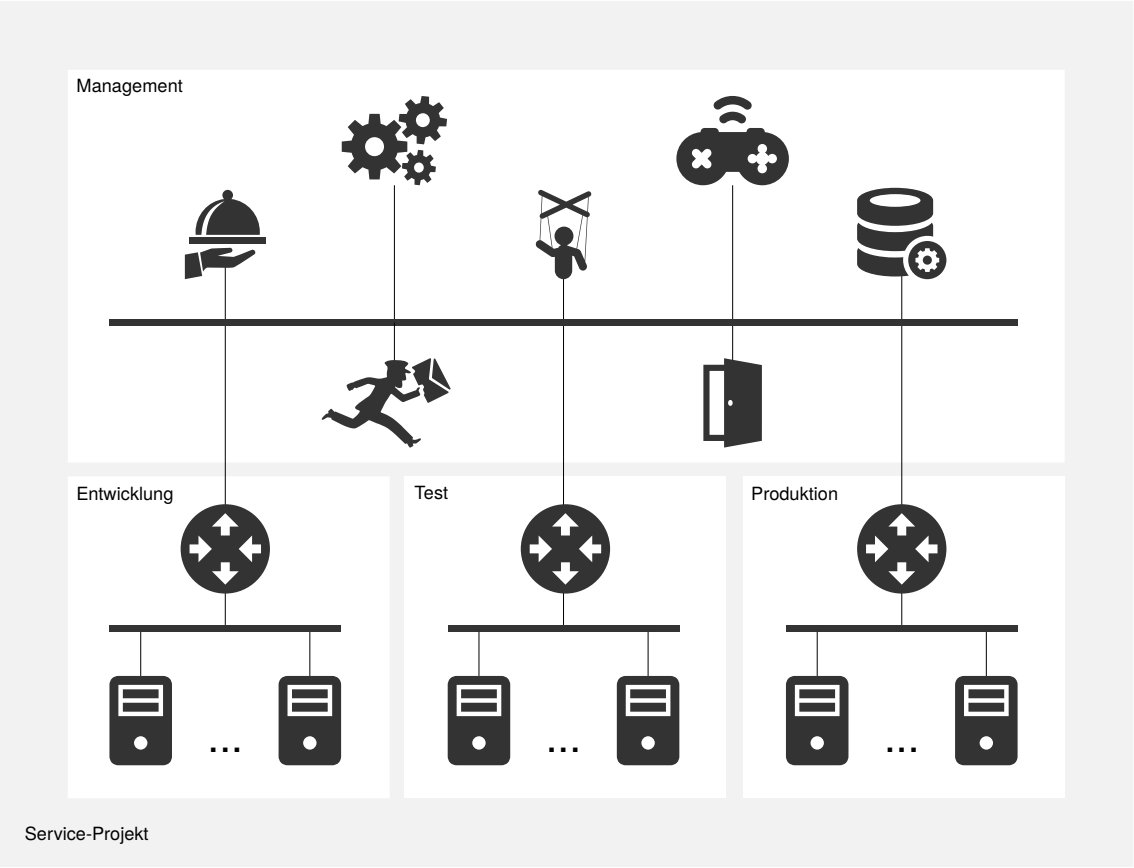


Abbildung 6.24: Funktionsdomänen zur Bildung logischer Projektunterstrukturen

Auf Grundlage dieser Funktionsdomänen lassen sich Varianten funktionaler Rollen definieren. Eine funktionale Rolle, die einem Produktivsystem zugewiesen ist, kann daher in den unterschiedlichen Umgebungen jeweils unterschiedlich definiert sein. Der Provisioner berücksichtigt die Domänenzugehörigkeit von Nodes bei der Konfigurationsableitung. Dies wirkt sich unmittelbar auf den Aufsetzungsprozess aus und führt zu Varianten innerhalb des Applikationsstapels. Dadurch lassen sich im Produktivbetrieb befindliche funktionale Rollen in der Entwicklungsumgebung weiterentwickeln und in der Testumgebung zwecks Überführung in den Produktivbetrieb auf Fehler testen.

6.8 Accounts

Im Rahmen des Informationsmodells gilt es zudem, Accounts und die Accountinhaber zu repräsentieren. Die Abbildung 6.25 zeigt die dafür erforderlichen Managementobjekte. Es handelt sich um:

Beteiligter Der *Beteiligte* stellt eine abstrakte Oberklasse dar und repräsentiert eine Person, Organisation oder Organisationseinheit.

Person Eine *Person* ist ein Mensch und entspricht einer unbeschränkt geschäftsfähigen natürlichen Person, die wirksame Rechtsgeschäfte abschließen kann. Gemäß § 2 BGB [Kö18]

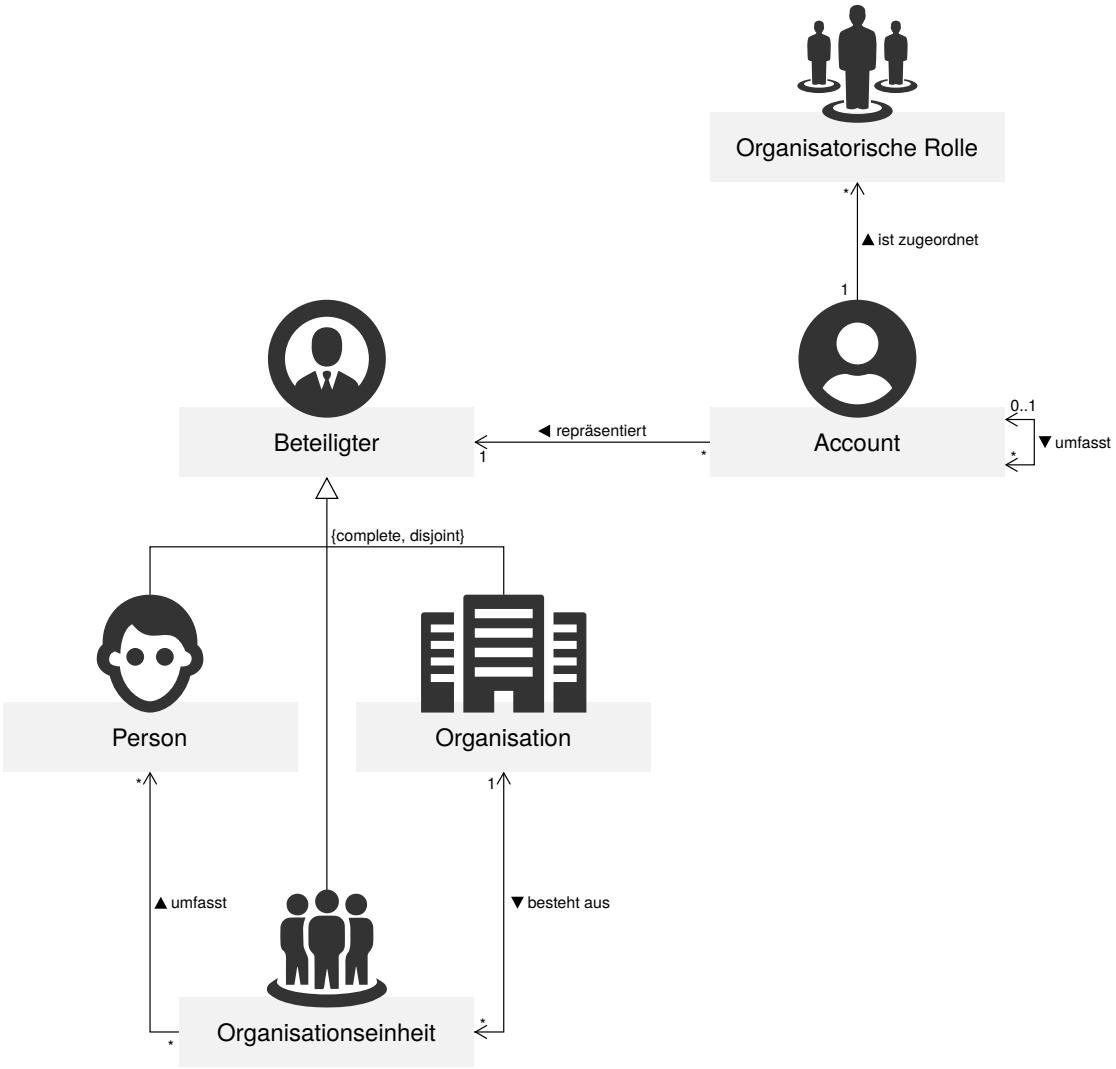


Abbildung 6.25: Managementobjekte zur Repräsentation von Accounts

gilt eine natürliche Person nach Vollendung des 18. Lebensjahres als unbeschränkt geschäftsfähig.

Organisation Eine *Organisation* ist eine juristische Person und bezeichnet eine Vereinigung von Personen oder Sachen zu einer rechtlich geregelten Einheit. Sie ist kraft gesetzlicher Bestimmung als solche rechts- und geschäftsfähig. Grundsätzlich unterscheidet man juristische Personen des privaten Rechts (Gesellschaften, Vereine und Genossenschaften) und juristische Personen des öffentlichen Rechts (Körperschaften, Anstalten und Stiftungen). Personen und Organisationen sind im juristischen Sinne Rechtssubjekte.

Organisationseinheit Eine *Organisationseinheit* ist ein Element der Aufbauorganisation und repräsentiert eine organisatorische Untereinheit wie eine Gemeinschaft, Arbeitsgruppe oder Abteilung. Die Organisationseinheit umfasst wiederum Personen, die in einem Vertragsverhältnis mit der Organisation stehen. Dabei handelt es sich um gewerbliche Mitarbeiter und Angestellte.

Account Der *Account* entspricht einem Benutzerkonto, das Personen, Organisationseinheiten und Organisationen im System repräsentiert. Er besitzt eine über Projektgrenzen hinweg eindeutige, unveränderliche Identifikation. Accounts lassen sich ineinander verschachteln. Damit können Unter-Accounts in Form einer Account-Hierarchie beschrieben werden. Mitarbeiter, Organisationseinheiten und Organisationen besitzen zwar in der Regel einen individuellen Kostenrahmen, durch die Festlegung eines Gesamtbudgets kann aber ein übergeordnetes Limit festgelegt werden, welches dafür sorgt, dass Mitarbeiter einer Abteilung eine Kostengrenze insgesamt nicht überschreiten.

Die Überwachungs- und Steuerungslogik muss Zugriff auf die Account-Informationen haben. Dadurch ist sie in der Lage zu entscheiden, ob einem Anwender ein Service zugänglich gemacht werden kann, ohne Kostenrahmen oder anderweitige Beschränkungen zu verletzen. Die Repräsentation der Verantwortlichen ist insbesondere im Zusammenhang mit Serviceverträgen unerlässlich. Hierbei reicht nicht aus, die Vertragspartner lediglich in Form ihrer organisatorischen Rollen zu erfassen. Vielmehr müssen sich die in den jeweiligen Rollen agierenden Rechtssubjekte eindeutig bestimmen lassen.

6.9 Berechtigungen

Managementobjekte sind schützenswerte Güter (engl. *Assets*) [Eck09], die zugleich einen hohen bis sehr hohen Schutzbedarf besitzen. Zum einen sind darüber personenbezogene Daten abrufbar, zum anderen lassen sich Umgebungsanpassungen anstoßen, die bei unsachgemäßer Anwendung Schäden verursachen können. Die Folgen hat der Service-Provider zu verantworten. Ziel ist es, die unautorisierte Einflussnahme und Informationsgewinnung zu unterbinden. Dies setzt eine Zugriffskontrolle voraus, die auf Grundlage einer Autorisierungsstrategie arbeitet.

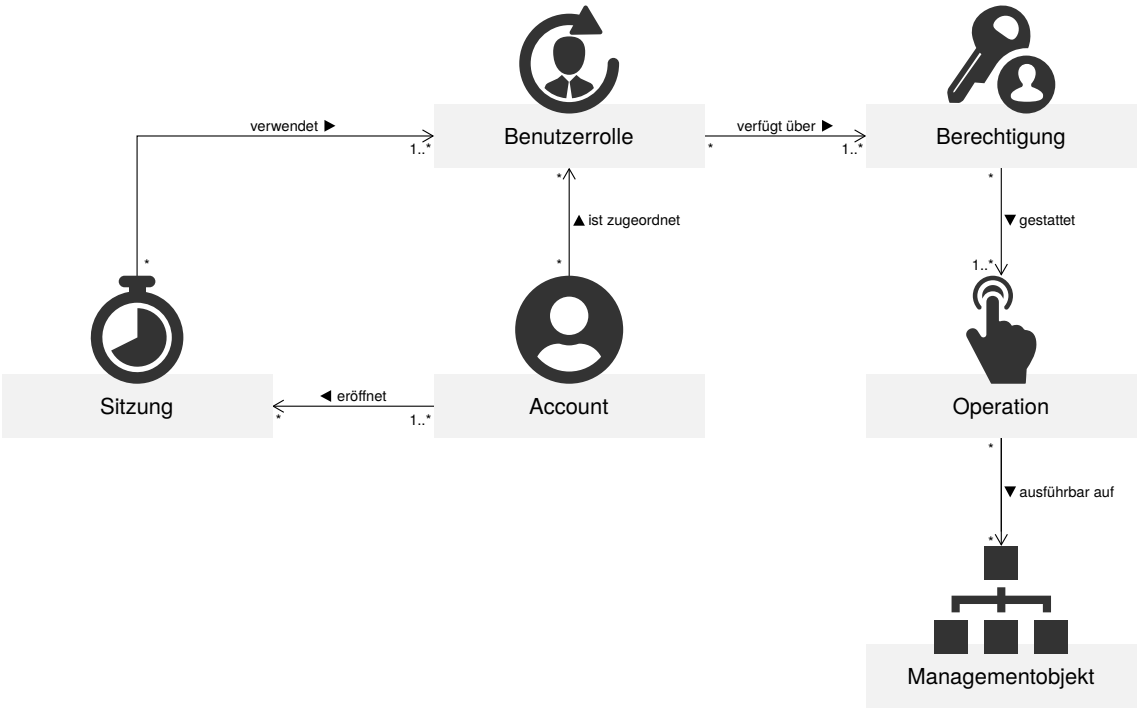


Abbildung 6.26: Managementobjekte zur Repräsentation von Berechtigungen

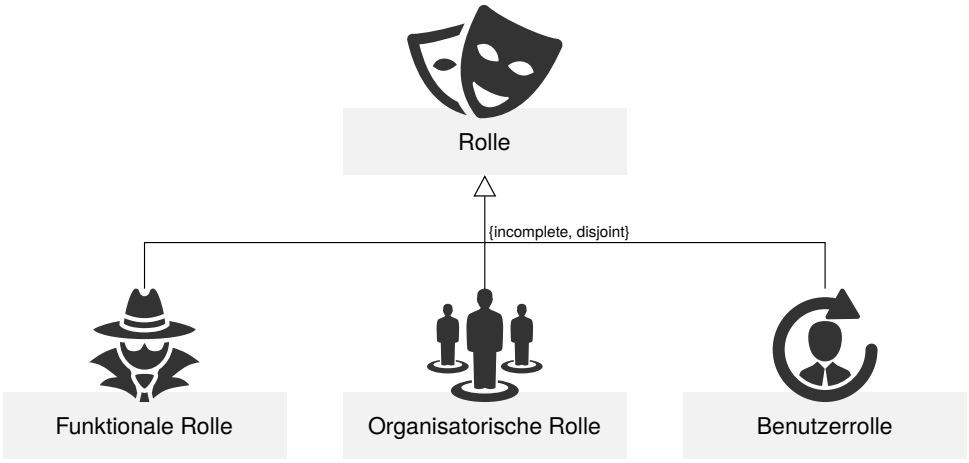


Abbildung 6.27: Managementobjekte zur Repräsentation von Rollen

Hierbei muss es dem Service-Customer möglich sein, die Einhaltung der Servicegütevereinbarungen eigenständig zu überprüfen. Demnach ist ihm Zugriff auf ausgewählte Managementobjekte zu gewähren. Dies erfordert den Einsatz eines feingranularen Berechtigungskonzepts. In Abhängigkeit zum Managementobjekt gilt es zu entscheiden, ob dem Subjekt die Ausführung der Operation gestattet ist. Als Autorisierungsstrategie hat sich die *rollenbasierte Zugriffskontrolle* (engl. *Role-Based Access Control*, RBAC) [FK92, SFK00] bewährt. Die Abbildung 6.26 zeigt die dafür erforderlichen Managementobjekte. Es handelt sich um:

Rolle Eine *Rolle* (engl. *Role*) entspricht einer Zusammenfassung von Funktionen, Kompetenzen und Verantwortlichkeiten. Funktionen beinhalten Tätigkeiten, Kompetenzen beschreiben Ausübungsrechte und Verantwortlichkeiten umfassen Pflichten über Handlungen, Ergebnisse und Folgen Rechenschaft abzulegen. Die Abbildung 6.27 zeigt die Vererbungshierarchie der Managementobjekte. Diese lassen sich in funktionale Rollen, organisatorische Rollen und Benutzerrollen unterteilen.

Benutzerrolle Eine *Benutzerrolle* dient der Gruppierung von Zugriffsrechten und fasst einzelne Berechtigungen zu einer Rolle zusammen. Accounts lassen sich beliebig viele dieser Rollen zuweisen. Mit Hilfe von Vererbungsbeziehungen kann eine Rollenhierarchie erstellt werden, die die Wiederverwendbarkeit und Anpassbarkeit der Rollen verbessert. Dadurch lassen sich Berechtigungen an untergeordnete Rollen weitervererben. Bei diesem Vorgehen besteht aber die Gefahr, dass ein Subjekt mehr Rechte erhält, als es für die Durchführung seiner eigentlichen Aufgabe benötigt.

Berechtigung Einer Benutzerrolle sind *Berechtigungen* (engl. *Permissions*) zugeordnet, die es dem Subjekt gestatten, auf Managementobjekten Operationen auszuführen. Es sind beliebige Kombinationen aus Managementobjekten und Operationen möglich.

Operation Die *Operation* steht stellvertretend für all diejenigen Aktionen, die Managementobjekte betreffen. Dabei handelt es sich in Übereinstimmung mit dem CRUD-Prinzip um das Hinzufügen (Create), Auslesen (Read), Ändern (Update) und Entfernen (Delete) von Managementobjekten. Das Hinzufügen umfasst das Erstellen eines Managementobjekts und seine Assoziation mit einem anderen. Eine Änderung beinhaltet das Erstellen, Löschen und Setzen von Konfigurationsvariablen. Das Auslesen bezieht sich auf die Abfrage von

Status- und Konfigurationsvariablen sowie die Traversierung von Assoziationen. Dazu muss das Subjekt über weitere Berechtigungen verfügen, die es ihm gestatten, auch auf diejenigen Managementobjekte lesend zuzugreifen, die sich jeweils am anderen Ende der Assoziation befinden.

Sitzung Eine *Sitzung* (engl. *Session*) repräsentiert eine aktive Arbeitseinheit eines Benutzers. Sie erlaubt die selektive Aktivierung und Deaktivierung von Rollen. Grundsätzlich kann ein Benutzer mehrere Sitzungen gleichzeitig besitzen. Eine Sitzung kann aber wiederum nur genau einem Benutzer zugeordnet sein.

Nach erfolgreicher Authentifizierung verfügt der Anwender über eine Menge aktiver Rollen, die sich innerhalb einer Sitzung dynamisch aktivieren und deaktivieren lassen. Die Durchführung einer Operation auf einem Managementobjekt muss ihm gewährt werden, wenn mindestens eine seiner aktiven Rollen über eine entsprechende Berechtigung verfügt.

6.10 Strukturmuster

Während bei der Informationsmodellierung grundlegende Managementobjektklassen und Assoziationen identifiziert werden, legen Strukturmuster spezifische Zusammenhänge zwischen Managementobjekten fest. Ein Strukturmuster entspricht einer Schablone, die einen Teilausschnitt des Objektmodells spezifiziert. Strukturmuster schränken die seitens des Informationsmodells gewährten Freiheitsgrade dahingehend ein, dass die Datenbasis über bestimmte Objektstrukturen verfügt. Diese bilden wiederum die Ausgangsbasis zur Formulierung von Steuerungsmustern. Mit Hilfe von Strukturmustern lassen sich der Aufbau der Managementobjekte, die Belegung der Managementvariablen und die Beziehungen zwischen Managementobjekten festlegen. Die nachfolgenden Abschnitte stellen vier Strukturmuster vor, die sich auf die Servicegüteanforderungen, die Serviceerbringung, das Infrastrukturmanagement und das Konfigurationsmanagement beziehen.

6.10.1 Servicegüteanforderungen

Die Abbildung 6.28 zeigt das Strukturmuster zur Repräsentation der Servicegüteanforderungen zwischen Nutzer, Anbieter und Zulieferer. Es ordnet die Serviceerbringung in einen rechtlichen Rahmen ein und erfasst die Vertragsbeziehungen zwischen den beteiligten organisatorischen Rollen. Den Ausgangspunkt bildet ein nicht-interaktiver Service. Darunter wird ein Service verstanden, der Arbeitsaufträge sequentiell abarbeitet und bei dem sich ein einmal gestarteter Verarbeitungsvorgang nachträglich nicht mehr beeinflussen lässt [SWC+07]. Es besteht lediglich die Möglichkeit, die Ausführung abzubrechen und zu einem späteren Zeitpunkt fortzusetzen oder neu zu starten. Die Verarbeitung terminiert entweder mit einem Ergebnis oder endet vorzeitig aufgrund eines Fehlers. Das Hauptanliegen des Service-Providers ist es, den Service anforderungsgerecht bereitzustellen und gütegesichert zu betreiben. Dies setzt die Definition eines Managementobjekts voraus, das den managementrelevanten Zustandsraum mit Hilfe von Statusvariablen beschreibt. Neben einer Bezeichnung verfügt es über einen Zustandswert, der angibt, in welcher Phase des Lebenszyklus sich der Service derzeit befindet. Zudem liefert es Informationen über Fortschritt, Startzeit und Endzeit. Der Fortschritt wird als Ganzzahl in Prozent gemessen und beschreibt den Fortgang der Auftragsverarbeitung. Diesem Wert wird ein besonderer Stellenwert beigemessen, da sich auf seiner Grundlage Aussagen über das zu erwartende Bearbeitungsende ableiten lassen. Der Arbeitsbereich enthält eine Pfadangabe, die

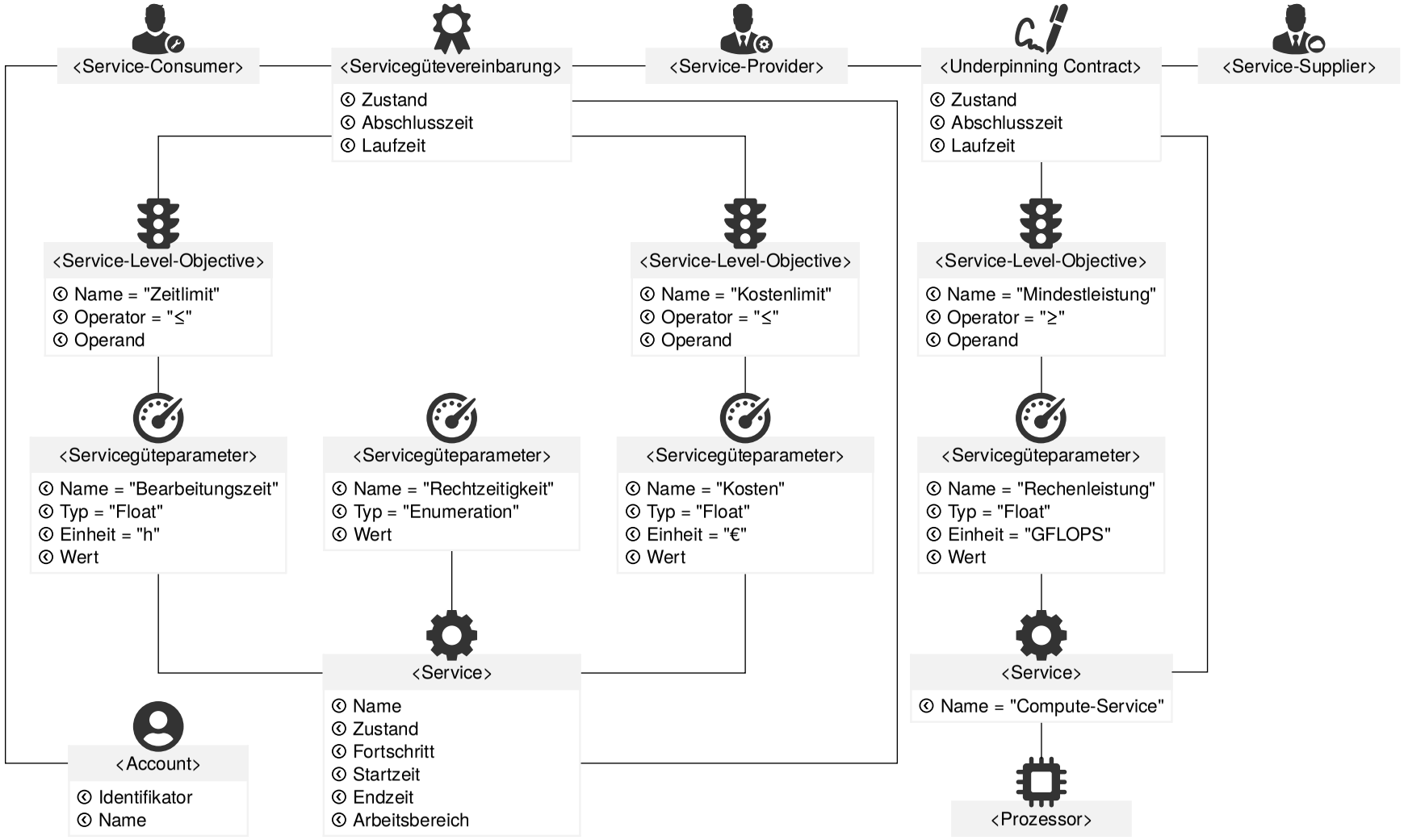


Abbildung 6.28: Strukturmuster zur Repräsentation der Servicegüteanforderungen zwischen Nutzer, Anbieter und Zulieferer

sich auf das Arbeitsverzeichnis des Services bezieht. Darüber kann auf die bei der Auftragserteilung bereitgestellten Eingabedaten zugegriffen werden. Zugleich dient der Arbeitsbereich als Ablageort für Zwischen- und Endergebnisse. Neben den Statusvariablen verfügt der Service über drei Servicegüteparameter. Es handelt sich hierbei um:

Bearbeitungszeit Die Bearbeitungszeit (engl. *Execution Time*) entspricht derjenigen Zeitspanne, die der Service zur Verarbeitung einer Eingabe benötigt und innerhalb derer er ein oder mehrere Betriebsmittel benötigt [SK98]. Ihre Leistungsfähigkeit wirkt sich unmittelbar auf die Bearbeitungszeit aus.

Rechtzeitigkeit Die Rechtzeitigkeit der Ausführung (engl. *Timeliness*) bewertet den Verarbeitungsfortschritt in Bezug zu einer Zielvorgabe und macht Angaben über den Einhaltungsgrad. Das Parameter kann folgende Werte annehmen:

- Erfüllt (Grün)
- Bedenklich (Gelb)
- Gefährdet (Orange)
- Kritisch (Rot)

Kosten Die Kosten entsprechen dem bis zum Abfragezeitpunkt angefallenen Gesamtbetrag, der dem Verantwortlichen nach Terminierung des Services in Rechnung gestellt wird. Er ist zusammengesetzt aus variablen und fixen Kosten. Fixe Kosten werden von ein- und ausgehenden Datenübertragungen sowie von Floating IPs verursacht, variable Kosten fallen insbesondere für Host-Systeme an und werden in der Regel pro Zeiteinheit erhoben.

Nicht-funktionale Anforderungen sind durch Service-Level-Objectives repräsentiert, die sich auf Servicegüteparameter beziehen und Zielvorgaben enthalten. Es kommen ein Zeitlimit und ein Kostenlimit zum Einsatz, die sich auf die Bearbeitungszeit bzw. den Kostenbetrag beziehen. Der Operator legt die Zielvorgabe als obere Schranke fest, der Operand enthält den Maximalwert, den es im Zuge der Serviceerbringung nicht zu überschreiten gilt. Zeit- und Kostenlimit sind Bestandteil einer Servicegütevereinbarung, die zwischen Service-Consumer und Service-Provider geschlossen wird und die sich auf einen einzelnen Verarbeitungsvorgang bezieht. Sie enthält neben einem Zustandswert und dem Zeitpunkt des Vertragsschlusses auch die Vertragslaufzeit. Damit ein Service-Provider die zugesicherte Leistung erbringen kann, schließt er mit externen Dienstleistern Zulieferverträge in Form von Underpinning Contracts ab. Vertragsgegenstand ist die von einem Flavor bereitgestellte Rechenleistung. Da sich Service-Verträge ausschließlich auf Services beziehen, ist die Definition eines Compute-Services erforderlich, der die Rechenkomponente aus einer serviceorientierten Sicht beschreibt. Diesem lässt sich ein Servicegüteparameter zuordnen, der seine Leistungsfähigkeit repräsentiert und in der Einheit *Gleitkommaoperationen pro Sekunde* (engl. *Floating Point Operations Per Second*, FLOPS) gemessen wird. Das Service-Level-Objective entspricht einer unteren Schranke in Form einer Mindestleistung. Die Flavors dienen dem Service-Provider als Ausgangsbasis zur anforderungsgerechten Serviceerbringung.

6.10.2 Serviceerbringung

Die Abbildung 6.29 zeigt das Strukturmuster der Serviceerbringung. Hierbei wird ein Service von einer Serviceinstanz erbracht, die ihrerseits von einer Softwarekomponente bereitgestellt wird und die wiederum auf einem Node installiert ist. Die Zuweisung einer funktionalen Rolle

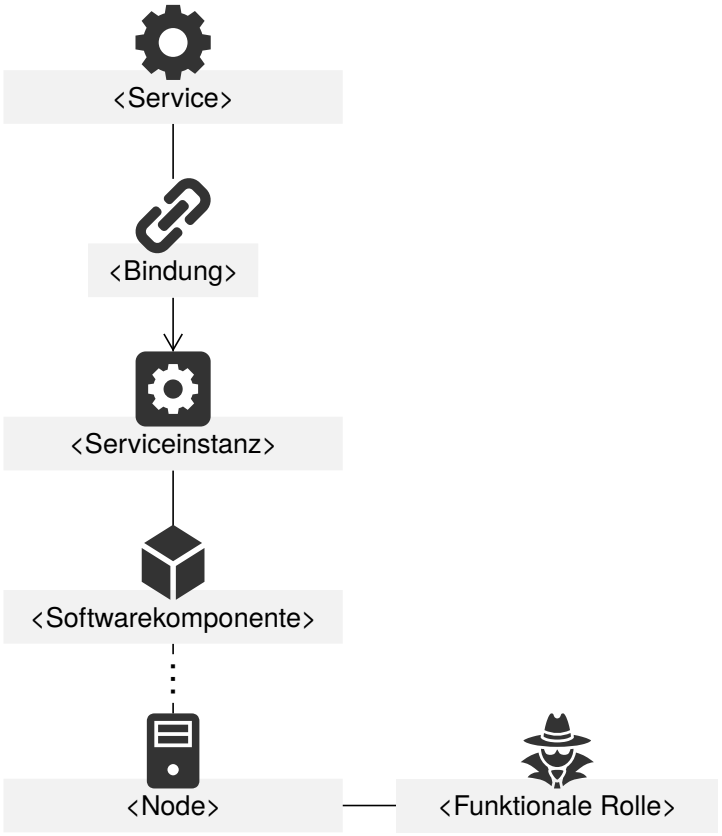


Abbildung 6.29: Strukturmuster zur Serviceerbringung

führt zum Aufsetzen des Applikationsstapels. Dieser wird in der Regel zusätzliche Laufzeitumgebungen und Middleware-Frameworks beinhalten, die zur Ausführung der Softwarekomponente erforderlich sind. Aufbau und Struktur sind jedoch von der funktionalen Rolle abhängig und lassen sich nicht allgemein festlegen. So kann eine Softwarekomponente direkt auf dem Betriebssystem aufsetzen oder eine komponenten- bzw. serviceorientierte Middleware voraussetzen.

Services und Serviceinstanzen

Grundsätzlich repräsentieren der Service und die Serviceinstanz dieselbe Funktion. Die beiden Managementobjekte bieten aber zwei unterschiedliche Sichten auf die Funktion. Der Service abstrahiert von den technischen Details der Serviceinstanz, die implementierungs- und technologiespezifische Status- und Konfigurationsvariablen bereithält. Er stellt die vereinheitlichten und insbesondere die für einen gütegesichterten Servicebetrieb erforderlichen Managementvariablen zur Verfügung. Dadurch lässt sich die Implementierungsebene sehr viel leichter anpassen, ohne dass damit zugleich Änderungen an den Steuerungsregeln einhergehen. Im Zusammenhang mit einem nicht-interaktiven Service beschreibt der Service den Verarbeitungsvorgang, wohingegen die Serviceinstanz den technischen Service der Ausführung repräsentiert. Der Verarbeitungsvorgang existiert grundsätzlich unabhängig von seiner Ausführung, erst die Bindung an eine Serviceinstanz bewirkt einen Bearbeitungsfortschritt. Zudem kann die Bindung jederzeit aufgelöst und neu erstellt werden. Dementsprechend können an einem einzelnen Verarbeitungsvorgang im Zeitverlauf unterschiedliche Serviceinstanzen, Implementierungen und Ausführungsknoten beteiligt sein.

6.10.3 Infrastrukturmanagement

Das Infrastrukturmanagement ist für das einheitliche Management heterogener Ausführungsumgebungen von zentraler Bedeutung. Die nachfolgende Abbildung 6.30 zeigt das Strukturmuster für das Infrastrukturmanagement:

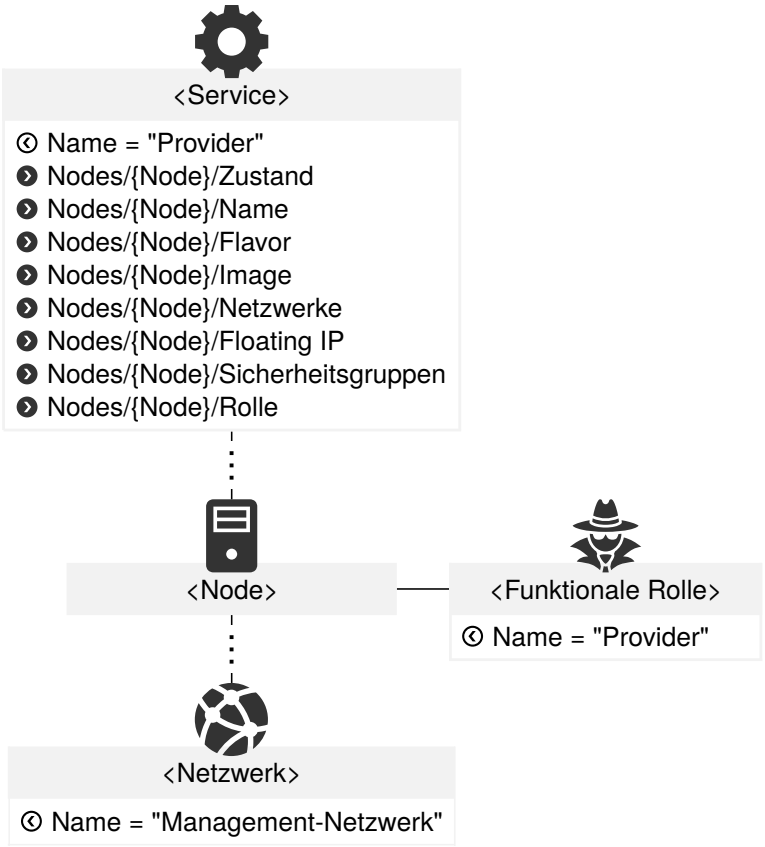


Abbildung 6.30: Strukturmuster für das Infrastrukturmanagement

Jedes Projekt besitzt einen dedizierten Node, der mit dem Management-Netzwerk verbunden ist und der zugleich die funktionale Rolle des Providers einnimmt. Das Managementobjekt des gleichnamigen Services verfügt über Konfigurationsvariablen, mittels derer sich die Infrastrukturschicht beeinflussen lässt. Sie abstrahieren von der Managementschnittstelle der virtuellen bzw. cloudbasierten Umgebung. Der Datensatz zur Beschreibung eines Nodes entspricht einem Struct, das als Elemente die folgenden Konfigurationsvariablen enthält:

Zustand Der Wert entspricht dem Sollzustand eines Nodes. Eine Wertänderung bewirkt eine Zustandstransition und spiegelt sich in der Statusvariablen des Managementobjekts wider, das den Node innerhalb der Infrastrukturschicht repräsentiert. Zulässige Belegungen sind:

- Aktiv
- Pausiert
- Gestoppt
- Gelöscht

Name Der Name legt die projektweit eindeutige Bezeichnung des Nodes fest.

Flavor Das Flavor entspricht der Bezeichnung des zu nutzenden Maschinentyps.

Image Das Image benennt das einzusetzende Abbild eines Betriebssystems.

Netzwerke Eine Liste von Netzwerken, an die der Node angebunden werden soll.

Floating IP Ein Wahrheitswert, der die Zuweisung einer öffentlichen IP-Adresse veranlasst. Die Angabe ist optional.

Sicherheitsgruppen Eine Liste von Sicherheitsgruppen, die dem Node zugeordnet werden sollen. Die Angabe ist optional.

Rolle Die funktionale Rolle, die der Node im Nutzungsszenario einnehmen soll. Die Angabe ist optional.

Beim Einsatz physikalischer Umgebungen können über das Management keine Nodes erstellt werden. Dementsprechend lassen sich dem Managementobjekt keine Datensätze hinzufügen. Unabhängig davon verfügt der Service für jeden Node über einen Satz von Konfigurationsvariablen. Dabei lassen sich der Zustand, die Floating IP, die Sicherheitsgruppen und die Rolle jederzeit verändern und neu setzen. Durch die Neuzuweisung einer Rolle kann die Bereitstellung einer anderen Anwendungsfunktion veranlasst werden. Dies ist sowohl in cloudbasierten wie auch virtuellen und physikalischen Umgebungen möglich. Nodes lassen sich wiederverwenden und unmittelbar in anderen Nutzungsszenarien einsetzen. Diese Fähigkeit ist insbesondere beim Einsatz physikalischer Umgebungen eine Grundvoraussetzung. In cloudbasierten und virtuellen Umgebungen lassen sich dadurch die Bereitstellungszeiten von Applikationen und Services mitunter deutlich verkürzen, indem ungenutzte oder unausgelastete Nodes mit der Ausführung beauftragt werden.

Aufbau eines Host-Systems

Die Abbildung 6.31 zeigt die bereits aus Abschnitt 6.3.1 bekannten Managementobjekte der Infrastrukturschicht. Sie sind Bestandteil eines Strukturmusters, das den Aufbau eines Host-Systems beschreibt. Der Node bildet das zentrale Element. Er verfügt über eine eindeutige Bezeichnung, einen Zustand, einen Typen und einen Startzeitpunkt. Der Typ gibt an, ob es sich um einen Bare-Metal-Server, eine virtuelle Maschine oder einen Container handelt. Das einem Node zugeordnete Block-Device besitzt eine eindeutige Bezeichnung, einen Mountpoint sowie Angaben zur Gesamtkapazität und zum freien Speicherplatz. Mit Hilfe von Lese- und Schreibgeschwindigkeit sowie Lese- und Schreiblatenz kann die vom Block-Device erbrachte Leistung überwacht werden. Zudem ermöglicht die Anzahl der bisher aufgetretenen Lese- und Schreibfehler Rückschlüsse, ob alsbald mit einem Ausfall zu rechnen ist. Das Netzwerk-Interface verfügt über einen Namen und macht Angaben zur Auslastung. Zudem liefern Empfangs- und Übertragungsrate wichtige Leistungswerte, wohingegen die Anzahl der bisher aufgetretenen Paketfehler auf Probleme bei der Datenübertragung hindeutet. Über eine dem Netzwerk-Interface zugeordnete IP-Adresse lassen sich Adresse, Sichtbarkeit und Zuweisungsmethode abfragen. Die Zuweisungsmethode legt fest, ob die Adresse fest an das Netzwerk-Interface gebunden ist oder unmittelbar neu vergeben wird, sollte es vom Node getrennt werden. Das Netzwerk verfügt über eine eindeutige Bezeichnung, eine Adresse sowie eine Subnetzmaske. Angaben über Paketverzögerungen, Jitter, Durchsatz, Paketverluste und Paketvertauschungen dienen

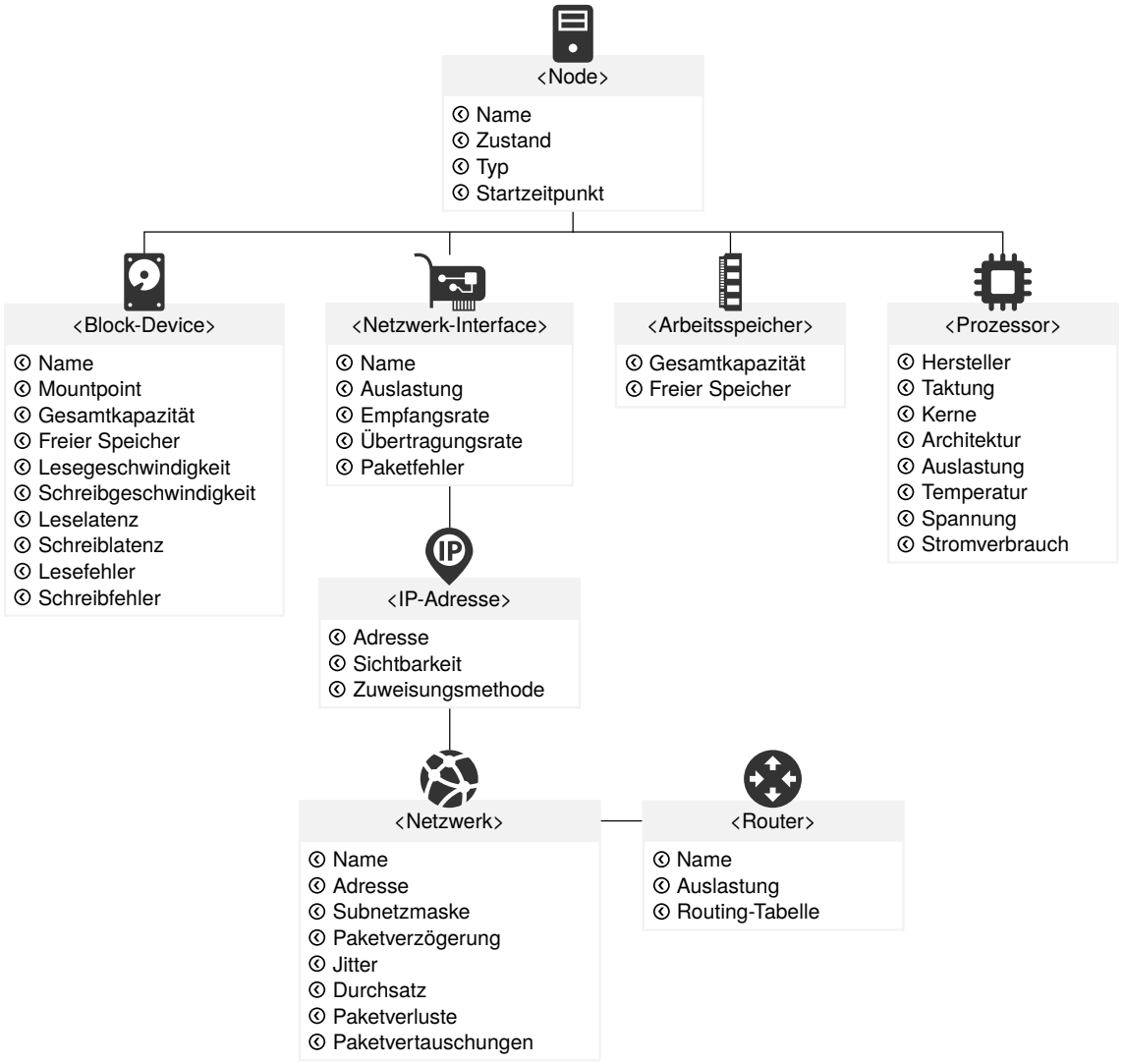


Abbildung 6.31: Managementobjekte der Infrastrukturschicht

der Leistungsüberwachung. Über den Router lassen sich neben der Bezeichnung auch die Auslastung und die Routing-Tabelle auslesen. Der einem Node zugeordnete Arbeitsspeicher enthält Angaben zur Gesamtkapazität sowie zum freien Speicherplatz. Der Prozessor stellt Informationen über Hersteller, Taktfrequenz, Anzahl der Kerne, Architektur, Auslastung, Temperatur, Spannung und Stromverbrauch zur Verfügung.

6.10.4 Konfigurationsmanagement

Neben dem Infrastrukturmanagement stellt das Konfigurationsmanagement eine weitere zentrale Managementfunktion dar. Über sie lassen sich die Applikationsstapel der Nodes beeinflussen. Dabei dient die funktionale Rolle als Bindeglied zwischen Infrastruktur- und Konfigurationsmanagement. Sie bestimmt zwar größtenteils den Applikationsstapel, dennoch ist zumeist noch eine nodespezifische Konfiguration erforderlich, die sich erst zur Laufzeit erstellen lässt. Die von Managementobjekten bereitgestellten Status- und Konfigurationsvariablen bilden die Datenbasis zur Erstellung einer derartigen Konfiguration. Damit das technische Management

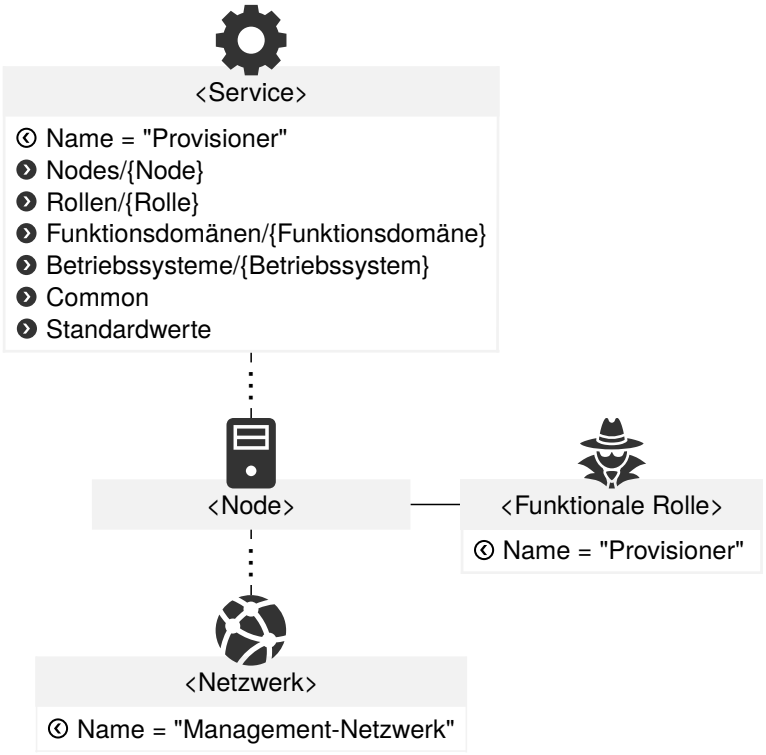


Abbildung 6.32: Strukturmuster für das Konfigurationsmanagement

auf die Applikationsstapel der Nodes einwirken kann, gilt es, ein Managementobjekt und insbesondere die dafür geeigneten Konfigurationsvariablen festzulegen. Die Abbildung 6.32 zeigt das Strukturmuster für das Konfigurationsmanagement. Ähnlich wie beim Infrastrukturmanagement wird in jedem Projekt ein dedizierter Node erstellt und mit dem Management-Netzwerk verbunden. Ihm ist die funktionale Rolle des Provisioners zugeordnet, was wiederum die Erstellung eines gleichnamigen Services bewirkt. Das Managementobjekt des Provisioners besitzt folgende Konfigurationsvariablen:

Nodes/{Node} Für jeden Node lässt sich eine gleichnamige Konfigurationsvariable erstellen, die jeweils die nodespezifische Konfiguration aufnimmt.

Rollen/{Rolle} Zudem kann für jede funktionale Rolle die Belegung einer gleichnamigen Konfigurationsvariablen angepasst werden. Sie umfasst diejenigen Konfigurationswerte, die für alle Nodes zur Anwendung kommen, denen die entsprechende Rolle zugewiesen ist.

Funktionsdomänen/{Funktionsdomäne} Auch lassen sich Konfigurationswerte hinterlegen, die für alle Nodes gelten, die sich in einer gemeinsamen Funktionsdomäne (Entwicklung, Test, Produktion …) befinden. Die Konfigurationswerte ermöglichen eine umgebungsspezifische Konfiguration, unabhängig von der Rolle, die ein Node im Rahmen des Anwendungsfalls einnimmt. Der Name der Konfigurationsvariablen entspricht der Bezeichnung der Funktionsdomäne.

Betriebssysteme/{Betriebssystem} Des Weiteren verfügt diese Konfigurationsvariable über Konfigurationswerte, die sich auf diejenigen Nodes beziehen, die über ein bestimmtes

Betriebssystem verfügen. Dabei handelt es sich unter anderem um Einstellungen für die Paketverwaltung. Der Name der Konfigurationsvariablen entspricht der Bezeichnung der Betriebssystemfamilie (Debian, RedHat, Suse ...).

Common Die Konfigurationsvariable besitzt eine feste, unveränderliche Bezeichnung und umfasst Konfigurationswerte, die für alle Nodes Gültigkeit haben, die sich innerhalb derselben Verwaltungsdomäne befinden. Sie umfassen unter anderem Zertifikate sowie Sprach- und Zeiteinstellungen.

Standardwerte Mit Hilfe dieser Konfigurationsvariablen lassen sich Standardwerte festlegen, die im Sinne eines Fallbacks immer dann Anwendung finden, falls für einen Konfigurationsparameter in keiner der zuvor festgelegten Konfigurationen eine Wertzuweisung erfolgt ist.

6.11 Steuerungsmuster

Steuerungsmuster spezifizieren das Verhalten des Managementprozesses und beschreiben adaptive Maßnahmen zur Systemkontrolle. Sie orientieren sich am Lösungsansatz zur externen Adaption [FLL+09, GBE+09]. Steuerungsmuster definieren Policy-Ausdrücke und Policy-Regeln, die innerhalb des Managementsystems durch ausführbaren Code zu repräsentieren sind. Hierbei kann die Code-Erstellung manuell oder mittels (teil-)automatisierter Generierung erfolgen. Ein Steuerungsmuster spezifiziert die Regeleinrichtung innerhalb eines Regelkreises, die den Systemzustand analysiert und Fehlfunktionen sowie Leistungsengpässe behebt. Die Regeleinrichtung gilt es, vor dem Einsatz auf das technische System und den jeweiligen Anwendungsfall hin abzustimmen. Zudem setzt ihre Anwendung die Existenz spezifischer Managementobjekte und deren Objektbeziehungen innerhalb der Datenbasis voraus, die als Eingabe bzw. Ausgabe dienen. Diese Zusammenhänge werden von den in Abschnitt 6.10 modellierten Strukturmustern erfasst. Auf deren Grundlage erfolgt nun der Entwurf der Steuerungsmuster für das einheitliche Management einer Multi-Provider-Umgebung.

6.11.1 Servicebereitstellung und -migration

Mit Hilfe des Steuerungsmusters zur Servicebereitstellung und -migration lassen sich Services innerhalb eines heterogenen Umgebungsverbunds gütegesichert betreiben. Hierbei schließt die Migration auch die vertikale Skalierung mit ein, bei der die Serviceausführung zwar in derselben Ausführungsumgebung, aber auf einem leistungsfähigeren Node fortgesetzt wird. Das Steuerungsmuster verantwortet die automatisierte Erzeugung und Migration von Services bei einer sich abzeichnenden Leistungsverschlechterung. Zu diesem Zweck lassen sich die am Verbund beteiligten Ausführungsumgebungen flexibel einsetzen. Das Steuerungsmuster besteht aus einem Policy-Ausdruck zur Flavorauswahl, auf den sich zwei Policy-Regeln stützen. Die Policys werden in den nachfolgenden Abschnitten vorgestellt.

6.11.1.1 Flavorauswahlausdruck

Die Serviceerbringung in einer Multi-Provider-Umgebung wird maßgeblich vom Flavorauswahlausdruck beeinflusst, der in der Abbildung 6.33 dargestellt ist. Er setzt auf dem in Abschnitt 6.10.1 spezifizierten Strukturmuster zur Repräsentation der Servicegüteanforderungen auf. Beim Flavorauswahlausdruck handelt es sich um eine Berechnungsvorschrift in Form eines Policy-Ausdrucks, der für einen nicht-interaktiven Service das zur Ausführung am besten geeignete

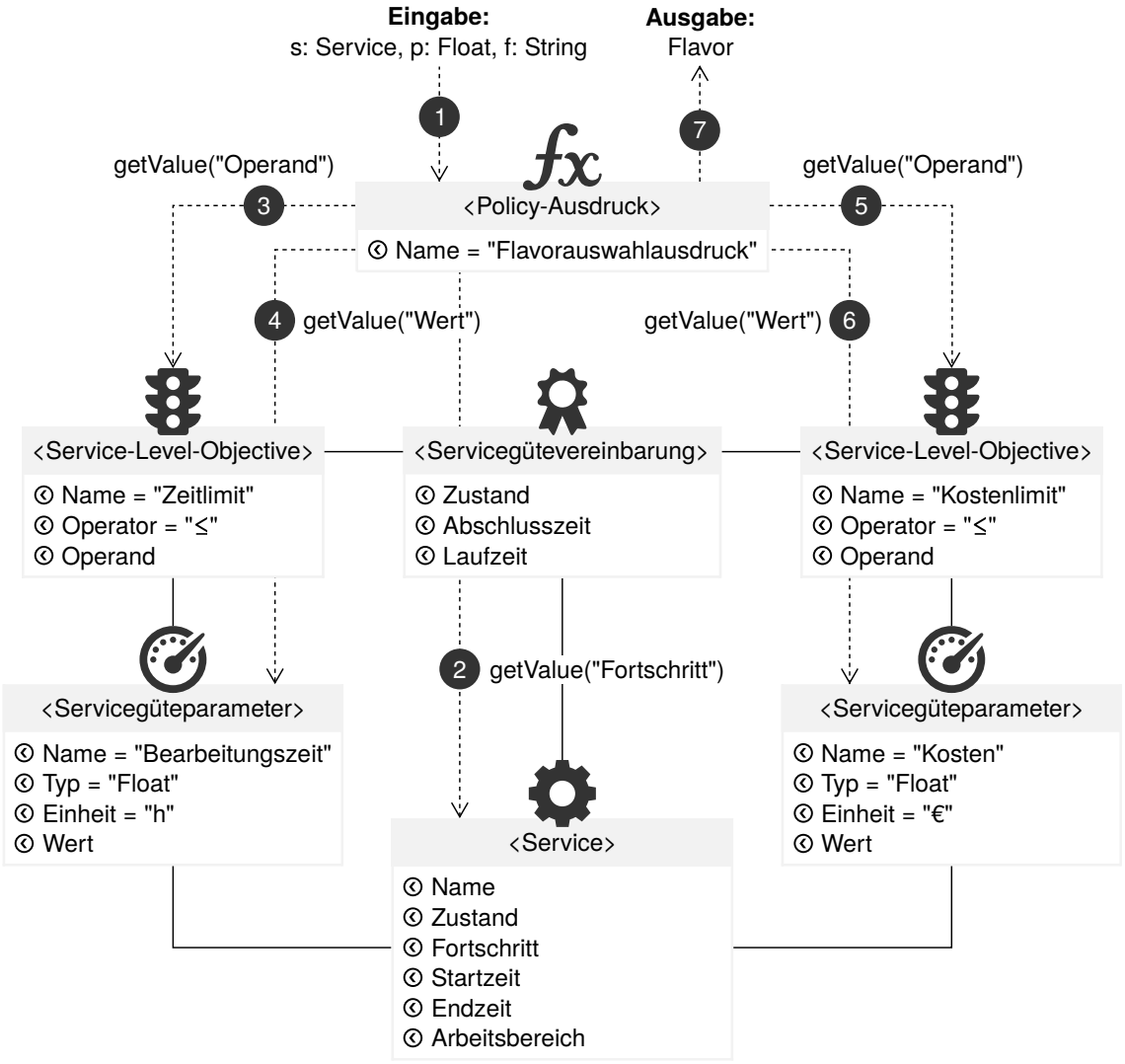


Abbildung 6.33: Flavorauswahlausdruck

Flavor ermittelt. Es gilt zum einen, die vom Nutzer vorgegebenen zeit- und kostenbezogenen Zielvorgaben einzuhalten, zum anderen hat die Serviceerbringung insgesamt zu minimalen Kosten zu erfolgen. Bei der Auswertung des Policy-Ausdrucks kommen die folgenden Einzelschritte zur Ausführung:

1. Der Flavorauswahlausdruck erfordert als Eingabe eine Referenz auf einen nicht-interaktiven Service *s*, einen Anteilswert *p* zwischen 0 und 1 zur Quantil-Bestimmung sowie einen optionalen Filterausdruck *f*.
2. Angestoßen durch eine Auswertungsanfrage liest der Flavorauswahlausdruck zunächst den Fortschrittswert der Auftragsverarbeitung aus.
3. Anschließend greift der Policy-Ausdruck über die mit dem Service assoziierte Servicegütevereinbarung auf das Zeitlimit zu und fragt den Operanden ab.
4. Zudem greift er auf den Wert des Servicegüteparameters zu, der die bisher verstrichene Bearbeitungszeit enthält.

5. Danach liest der Flavorauswahlausdruck den Operanden des Kostenlimits aus.

6. Im vorletzten Schritt greift der Policy-Ausdruck auf denjenigen Servicegüteparameter zu, der den bisher angefallenen Kostenbetrag enthält und fragt den Wert der Statusvariablen ab.

7. Abschließend ermittelt der Flavorauswahlausdruck auf Basis der Eingabe- und Zustandswerte das Endergebnis. Der Policy-Ausdruck verfügt für jedes innerhalb der Multi-Provider-Umgebung einsetzbare Flavor über eine separate Wahrscheinlichkeitsverteilung. Damit lässt sich die Bearbeitungszeit eines auf dem Flavor ausgeführten Arbeitsauftrags näherungsweise bestimmen. Zunächst werden diejenigen Flavors verworfen, die der Filterausdruck ablehnt. Anschließend berechnet der Policy-Ausdruck jeweils das Quantil $Q(p)$ und vergleicht das Ergebnis mit dem Zeitlimit. Es werden diejenigen Flavors verworfen, die die Zielvorgabe überschreiten. Die Berechnungsvorschrift lautet wie folgt:

```
Verbleibende Bearbeitungszeit = Q(p) / 100 * (100 - Fortschritt);
Geschätzte Bearbeitungszeit = Bisherige Bearbeitungszeit + Verbleibende
    ↪ Bearbeitungszeit;
```

   Ein Flavor wird genau dann in die engere Auswahl genommen, falls gilt:

```
Geschätzte Bearbeitungszeit ≤ Zeitlimit
```

   Im Anschluss berechnet der Policy-Ausdruck die beim Einsatz des Flavors zu erwartenden Kosten. Das Ergebnis wird auf den bisher angefallenen Kostenbetrag aufgeschlagen und mit der Zielvorgabe verglichen. Aus der Ergebnismenge werden nun auch diejenigen Flavors entfernt, die zu einer Überschreitung des Kostenlimits führen. Für die verbleibenden Flavors gilt:

```
Bisherige Kosten + Erwartete Kosten ≤ Kostenlimit
```

   Der Flavorauswahlausdruck gibt abschließend dasjenige Flavor zurück, das im Vergleich zu denen der Ergebnismenge die insgesamt geringsten Kosten verursacht.

Zusammenfassend lässt sich der Flavorauswahlausdruck wie folgt darstellen:

Eingabe	s: Service	Referenz auf einen nicht-interaktiven Service
	p: Float	Anteilswert zwischen 0 und 1 zur Quantil-Bestimmung
	f: String	Optionaler Filterausdruck
Ausgabe	Flavor	Referenz auf ein zur Serviceerbringung geeignetes Flavor

Funktion

- Führe für jedes Flavor der Multi-Provider-Umgebung folgende Schritte durch:
  - Wende den Filterausdruck an und überspringe das Flavor gegebenenfalls.
  - Bestimme das p-Quantil der Wahrscheinlichkeitsverteilung.
  - Schätze die Bearbeitungszeit ab.
  - Schätze die anfallenden Kosten der Serviceausführung ab.
  - Füge das Flavor zur Ergebnismenge hinzu, falls gilt:
    - Geschätzte Bearbeitungszeit $\leq$ Zeitlimit
    - Geschätzte Kosten $\leq$ Kostenlimit
- Gebe aus der Ergebnismenge dasjenige Flavor zurück, das im Vergleich zu den anderen insgesamt die geringsten Kosten verursacht.

6.11.1.2 Bereitstellungsregel

Die Bereitstellungsregel stellt eine der beiden Policy-Regeln zur Steuerung der Multi-Provider-Umgebung dar. Die Policy-Regel ist in der Abbildung 6.34 dargestellt. Zu den Aufgaben der Bereitstellungsregel zählen die Erzeugung und die initiale Provisionierung eines Host-Systems zur Serviceausführung. Die Funktionsweise der Policy-Regel ist folgende:

1. Die Bereitstellungsregel wird durch das Hinzufügen eines nicht-interaktiven Services angestoßen. Im Zuge der Bedingungsausführung erfolgt zunächst eine Überprüfung, ob es sich hierbei um einen Service handelt, für dessen Erbringung die Policy-Regel verantwortlich ist. Zudem muss sich der Verarbeitungsvorgang in einem inaktiven Zustand befinden. Nur wenn beide Anforderungen zugleich erfüllt sind, kommt die Regelaktion zur Ausführung. Die Regelbedingung lautet wie folgt:

```
service.getValue("Name") == ${Servicename} and service.getValue("
    ↪ Zustand") == "Inaktiv"
```

2. Die Bereitstellungsregel fordert im Zuge der Aktionsausführung den Flavorauswahlausdruck dazu auf, ein zur Serviceerbringung geeignetes Flavor zu ermitteln. Als Eingabe dienen die Referenz auf den neu hinzugefügten Service *s*, ein Anteilswert *p* und ein Filterausdruck *f*.

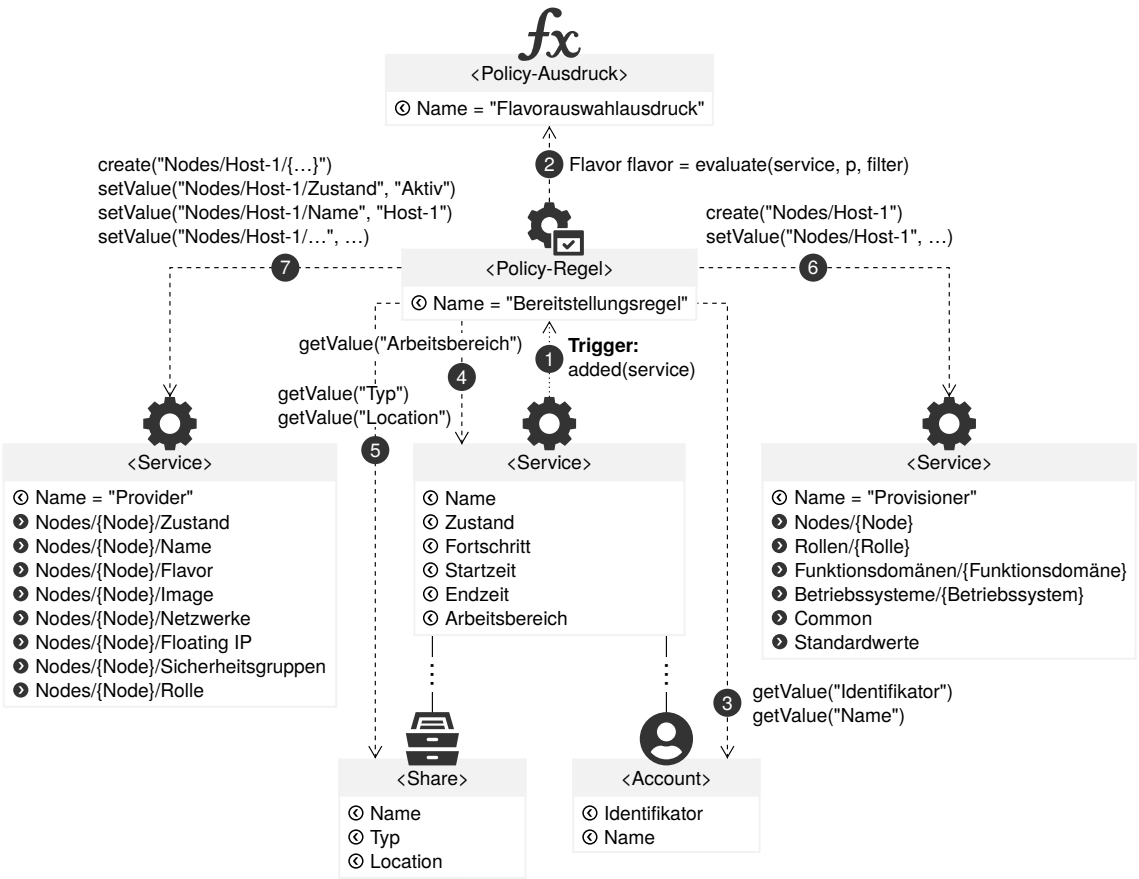


Abbildung 6.34: Bereitstellungsregel

3. Danach traversiert die Policy-Regel über die mit dem Service assoziierte Servicegütevereinbarung zum Account, der den Nutzer innerhalb des Systems repräsentiert und fragt den Identifikator und die Bezeichnung ab.

4. Der Service ist mit einem Share assoziiert, auf dem sich der Arbeitsbereich befindet. Aus dem Managementobjekt des Shares liest die Policy-Regel die Speicheradresse aus.

5. Anschließend greift sie auf das Managementobjekt des Services zu und fragt die Pfadangabe des Arbeitsbereichs ab.

6. Auf Grundlage des vom Flavorauswahlausdruck bestimmten Flavors ermittelt die Bereitstellungsregel den für das Zielprojekt verantwortlichen Provisioner- und Provider-Service. Zudem erzeugt sie basierend auf den zuvor ausgelesenen Statusvariablen eine nodespezifische Konfiguration, die unter anderem das Share in das lokale Dateisystem einhängt. Das Codefragment, das diese Teilschritte umfasst, lautet wie folgt:

```
Flavor flavor = flavorauswahlausdruck.evaluate(service, p, f);
Projekt projekt = flavor.projekt;
Ausführungsumgebung ausführungsumgebung = projekt.ausführungsumgebung;
List<Service> services = projekt.services;
Service provisioner = services.select(s | s.getValue("Name") == "
    ↪ Provisioner");
Service provider = services.select(s | s.getValue("Name") == "Provider
    ↪ ");
String configuration = createConfiguration(...);
```

Im Anschluss überprüft die Bereitstellungsregel, ob es sich bei der Ausführungsumgebung um eine physikalische, virtuelle oder cloudbasierte Umgebung handelt. Liegt eine physikalische Umgebung vor, der sich über das Management per Definition keine weiteren Nodes hinzufügen lässt, wird innerhalb des Zielprojekts derjenige Node ermittelt, der vom entsprechenden Flavor abstammt und der momentan am wenigsten ausgelastet ist. Im Anschluss wird die Konfiguration gesetzt. Das folgende Codefragment verdeutlicht das Vorgehen:

```
List<Node> nodes = projekt.nodes.filter(n | n.flavor == flavor);
Node node = nodes.min(Comparator.comparing(n | n.prozessor.getValue("
    ↪ Auslastung")));
String name = node.getValue("Name");
provisioner.setValue("Nodes" + "/" + name, configuration);
```

Handelt es sich bei der Ausführungsumgebung hingegen um eine virtuelle oder cloudbasierte Umgebung, wird eine neue Node-Bezeichnung generiert und seine Konfiguration über den Provisioner-Service gesetzt. Das Codefragment dafür lautet wie folgt:

```
String name = "Host-" + UUID.randomUUID();
provisioner.setValue("Nodes" + "/" + name, configuration);
```

7. Bei einer physikalischen Umgebung muss sichergestellt sein, dass der Node über die zur Serviceausführung erforderliche Rolle verfügt. Ist dies nicht der Fall, erfolgt die entsprechende Rollenzuweisung. Die Neuklassifikation des Nodes und damit die Anpassung seines Applikationsstapels bewirkt die folgende Codezeile:

```
provider.setValue("Nodes" + "/" + name + "/" + "Rolle", ${Rolle});
```

Bei einer virtuellen oder cloudbasierten Umgebung wird der Provider-Service, ausgehend von der zuvor generierten Node-Bezeichnung dazu aufgefordert, einen neuen Node zu erstellen. Die Bereitstellungsregel veranlasst zunächst die Erzeugung der Konfigurationsvariablen und führt anschließend die Wertzuweisungen durch. Dies veranschaulicht das folgende Codefragment:

```
provider.create("Nodes + "/" + name + "/" + {Zustand, Name, Flavor,
    ↪ Image, Rolle}");
provider.setValue("Nodes + "/" + name + "/" + Zustand", "Aktiv");
provider.setValue("Nodes + "/" + name + "/" + Name", name);
provider.setValue("Nodes + "/" + name + "/" + Flavor", flavor.getValue
    ↪ ("Name"));
provider.setValue("Nodes + "/" + name + "/" + Image", ${Image});
provider.setValue("Nodes + "/" + name + "/" + Rolle", ${Rolle});
```

Zusammenfassend lässt sich die Bereitstellungsregel wie folgt darstellen:

Ereignis Ein Service wurde hinzugefügt.

Bedingung Die Serviceerbringung unterliegt der Regelzuständigkeit, und der Service befindet sich in einem inaktiven Zustand.

Aktion

- Bestimme durch Auswertung des Flavorauswahlausdrucks ein zur Serviceerbringung geeignetes Flavor *f*.
- Ermittle Account, Share und Arbeitsbereich des Nutzers.
- Bestimme den für das Zielprojekt zuständigen Provisioner- und Provider-Service.
- Erzeuge die Konfiguration des Nodes, stelle dabei sicher, dass über das lokale Dateisystem auf den Arbeitsbereich des Nutzers zugegriffen werden kann.
- Erfolgt die Bereitstellung in einer physikalischen Umgebung, so gilt:
  - Bestimme alle Nodes mit dem Flavor *f*.
  - Ermittle den Node mit der geringsten Prozessorauslastung.
  - Hinterlege am Provisioner-Service die Konfiguration für diesen Node.
  - Weise dem Node gegebenenfalls seine neue funktionale Rolle zu.
- Erfolgt die Bereitstellung in einer virtuellen oder cloudbasierten Umgebung, so gilt:
  - Erzeuge einen projektweit eindeutigen Node-Namen.
  - Hinterlege am Provisioner-Service die entsprechende Konfiguration.
  - Fordere den Provider-Service zur Erzeugung des Nodes mit dem Flavor *f* auf.
  - Weise dem Node seine funktionale Rolle zu.

6.11.1.3 Migrationsregel

Die Bereitstellungsregel wird um eine Migrationsregel ergänzt, die in der Abbildung 6.35 dargestellt ist. Es handelt sich hierbei um eine Policy-Regel, die Services bei einer sich abzeichnenden

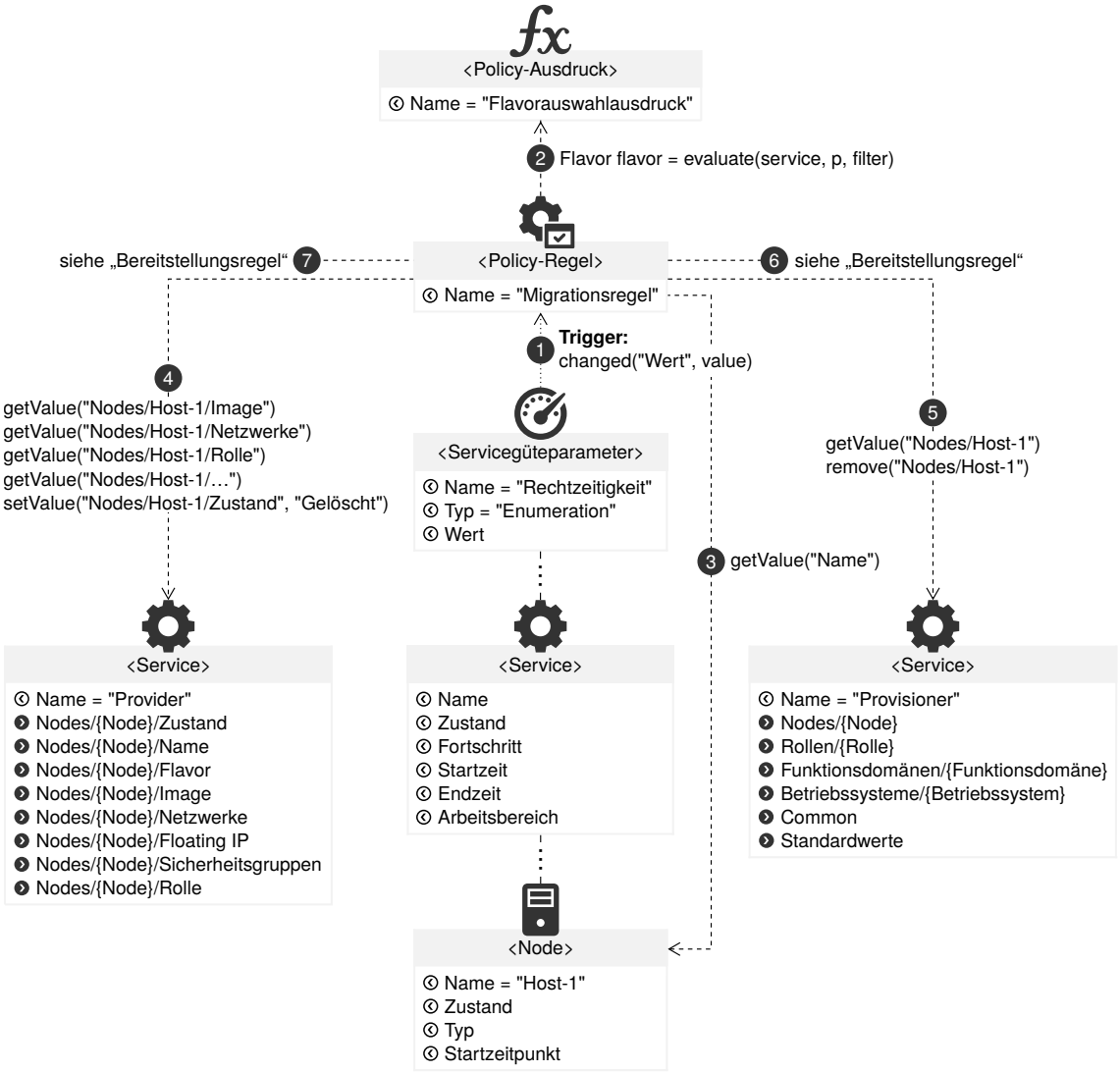


Abbildung 6.35: Migrationsregel

Zielverfehlung auf einen leistungsfähigeren Node migriert. Grundsätzlich entspricht die Servicemigration einer Maßnahme zur Selbstadaption und dient der perfektiven bzw. korrektiven Adaption. Die Funktionsweise der Policy-Regel ist folgende:

1. Die Migrationsregel wird durch eine Wertänderung des Servicegüteparameters zur Bewertung der Rechtzeitigkeit der Ausführung angestoßen. Im Zuge der Bedingungsausführung erfolgt zunächst eine Überprüfung, ob die Wertänderung einem solchen Servicegüteparameter zuzuordnen ist. Dazu liest die Migrationsregel seine Bezeichnung aus und führt einen Vergleich durch. Zudem überprüft sie, ob die Zielabweichung einen kritischen Wert erreicht hat und der Einsatz einer adaptiven Maßnahme erforderlich ist. Nur wenn beide Anforderungen zugleich erfüllt sind, kommt die Regelaktion zur Ausführung. Die Regelbedingung lautet wie folgt:

```
servicegüteparameter.getValue("Name") == "Rechtzeitigkeit" and serviceg
    ↪ üteparameter.getValue("Wert") == "Kritisch"
```

2. Im Zuge der Aktionsausführung macht die Migrationsregel Gebrauch vom Flavorauswahlausdruck und fordert die Ermittlung eines zur Serviceerbringung geeigneten Flavors an. Dazu bestimmt die Policy-Regel zunächst denjenigen Service, der über eine Assoziation direkt mit dem Servicegüteparameter verbunden ist. Als Eingabe dienen die Servicereferenz *s*, ein Anteilswert *p* und ein Filterausdruck *f*. Da der Flavorauswahlausdruck den aktuellen Istzustand sowie den Sollzustand der Serviceausführung berücksichtigt, wird er dem Aufrufer typischerweise ein leistungsfähigeres Flavor zurückliefern. Dennoch überprüft die Migrationsregel vorab, ob sich das berechnete Flavor vom derzeitig verwendeten unterscheidet. Nur im positiven Fall wird eine Servicemigration durchgeführt. Das folgende Codefragment verdeutlicht die Arbeitsweise:

```
Flavor flavor = flavorauswahlausdruck.evaluate(service, p, f);
Node node = (Node) service.traverse(o | o instanceof Node);
if (node.flavor != flavor) then
  ...
end if
```

3. Zur Vorbereitung der Servicemigration liest die Policy-Regel zunächst die Bezeichnung des Nodes aus, auf dem der Verarbeitungsvorgang derzeit ausgeführt wird.

4. Sie greift über den Provider-Service auf die dem Node zugeordneten Konfigurationsvariablen zu und liest ihre Wertebelegungen aus. Im Anschluss veranlasst sie eine Löschung des Nodes.

5. Sie liest die am Provisioner-Service hinterlegte nodespezifische Konfiguration aus und fordert ihre Löschung an.

6. Danach wird die Konfiguration dem für das Zielprojekt verantwortlichen Provisioner-Service hinzugefügt.

7. Abschließend veranlasst die Migrationsregel bei einer virtuellen oder cloudbasierten Umgebung die Erstellung eines neuen Nodes im Zielprojekt. Die Policy-Regel erzeugt einen Satz von Konfigurationsvariablen und setzt sie auf die zuvor ausgelesenen Werte. Dabei muss allerdings anstelle der vorherigen Flavor-Bezeichnung nun der Name des im Zielprojekt zu nutzenden Flavors verwendet werden. Bei einer physikalischen Umgebung gilt es sicherzustellen, dass dem Node die zur Fortsetzung des Verarbeitungsvorgangs notwendige Rolle zugeordnet ist.

Zusammenfassend lässt sich die Migrationsregel wie folgt darstellen:

Ereignis Der Wert des Servicegüteparameters zur Erfassung der Rechtzeitigkeit der Serviceausführung wurde geändert.

Bedingung Die Rechtzeitigkeit gilt als kritisch.

Aktion
- Ermittle denjenigen Service, der über den Servicegüteparameter verfügt.
- Bestimme durch Auswertung des Flavorauswahlausdrucks ein zur Serviceerbringung geeignetes Flavor *f*.
- Ermittle den Node, der an der Serviceausführung derzeit beteiligt ist.
- Ist ihm nicht das Flavor f zugeordnet, so gilt:

- Ermittle den für den Node zuständigen Provider-Service.
- Lies dort die Konfigurationsvariablen des Nodes aus.
- Wird der Node in einer virtuellen oder cloudbasierten Umgebung ausgeführt, fordere seine Löschung an.
- Ermittle den für den Node zuständigen Provisioner-Service.
- Lies die dort hinterlegte Konfiguration aus und fordere ihre Löschung an.
- Füge dem Provisioner-Service des Zielprojekts die zuvor ausgelesene Konfiguration hinzu.
- Soll die Serviceausführung auf einem Node fortgesetzt werden, der in einer virtuellen oder cloudbasierten Umgebung betrieben wird, so veranlasse seine Bereitstellung.

6.11.2 Rechtzeitigkeits- und Fortschrittskontrolle

Das Steuerungsmuster zur Rechtzeitigkeits- und Fortschrittskontrolle bewertet die Rechtzeitigkeit der Serviceausführung und überwacht den Verarbeitungsfortschritt. Es stellt einen funktionalen Zusammenhang her zwischen dem Servicegüteparameter, der Zielvorgabe und den Zustandsinformationen. Dazu gilt es, die Zustandsinformationen mit Hilfe einer Berechnungsvorschrift auf den Wertebereich des Servicegüteparameters abzubilden. Das Ergebnis wird in der Datenbasis hinterlegt und steht über Managementvariablen zur Abfrage bereit. Es bildet die Ausgangsbasis zur automatisierten Servicebereitstellung und -migration. Zentraler Bestandteil des Steuerungsmusters ist ein Policy-Ausdruck zur Ermittlung der gegenwärtigen Zielabweichung, auf den sich zwei Policy-Regeln stützen. Die Policys werden in den nachfolgenden Abschnitten vorgestellt.

6.11.2.1 Zielabweichungsausdruck

Die Abbildung 6.36 zeigt den Zielabweichungsausdruck. Die Aufgabe des Policy-Ausdrucks ist es, ausgehend vom Fortschritt und der bisher verstrichenen Bearbeitungszeit, die prozentuale Zielabweichung vom Zeitlimit zu bestimmen. Die Zielabweichung lässt sich lediglich abschätzen, das Ergebnis ist eine Näherung der tatsächlichen Zielverfehlung. Bei der Auswertung des Policy-Ausdrucks kommen folgende Einzelschritte zur Ausführung:

1. Der Zielabweichungsausdruck erfordert als Eingabe eine Referenz auf einen nicht-interaktiven Service *s*.
2. Im Anschluss liest der Policy-Ausdruck den Fortschrittswert der Auftragsverarbeitung aus.
3. Zudem greift er auf den Servicegüteparameter zu, der die bisher verstrichene Bearbeitungszeit repräsentiert und liest den Wert der Statusvariablen aus.
4. Der Policy-Ausdruck ermittelt das mit dem Servicegüteparameter assoziierte Service-Level-Objective, welches das seitens des Nutzers festgelegte Zeitlimit enthält und fragt den Wert des Operanden ab.
5. Basierend auf den Zustandswerten schätzt der Zielabweichungsausdruck die derzeitige prozentuale Zielverfehlung der Serviceausführung ab. Unter der Annahme einer linearen Wachstumsrate gilt:

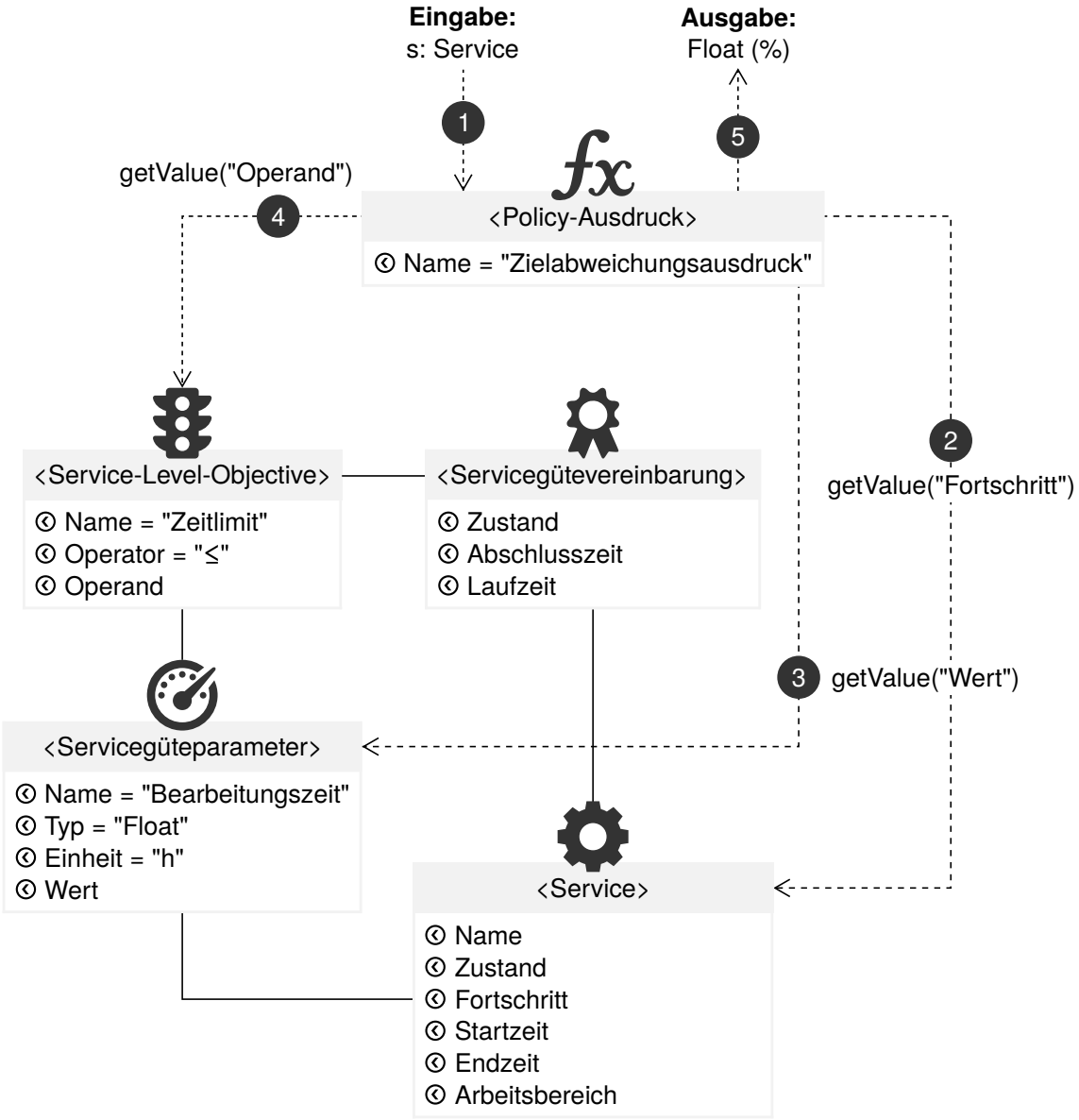


Abbildung 6.36: Zielabweichungsausdruck

```
Geschätzte Bearbeitungszeit = Bisherige Bearbeitungszeit * 100 /
    ↪ Fortschritt;
Prozentuale Zielverfehlung = 100 - 100 * Geschätzte Bearbeitungszeit /
    ↪ Zeitlimit;
```

Das Ergebnis ist negativ, falls die prognostizierte die vereinbarte Bearbeitungszeit unterschreitet. In diesem Fall ist die Berechnung schneller als zuvor eingeplant. Das Ergebnis ist hingegen positiv, falls eine Überschreitung der vereinbarten Bearbeitungszeit und demzufolge eine Zielverfehlung erwartet wird.

Zusammenfassend lässt sich der Zielabweichungsausdruck wie folgt darstellen:

Eingabe	s: Service	Referenz auf einen nicht-interaktiven Service
Ausgabe	Float	Prozentuale Zielverfehlung der Serviceausführung
Funktion	• Ermittle den Verarbeitungsfortschritt, die bisher verstrichene Bearbeitungszeit und das vereinbarte Kostenlimit. • Schätze das Ende der Serviceausführung ab. • Berechne die prozentuale Zielverfehlung der Serviceausführung.	

6.11.2.2 Rechtzeitigkeitsregel

Die Abbildung 6.37 zeigt die Rechtzeitigkeitsregel. Es handelt sich um eine Policy-Regel, deren Aufgabe darin besteht, den Servicegüteparameter zur Bewertung der Rechtzeitigkeit einer

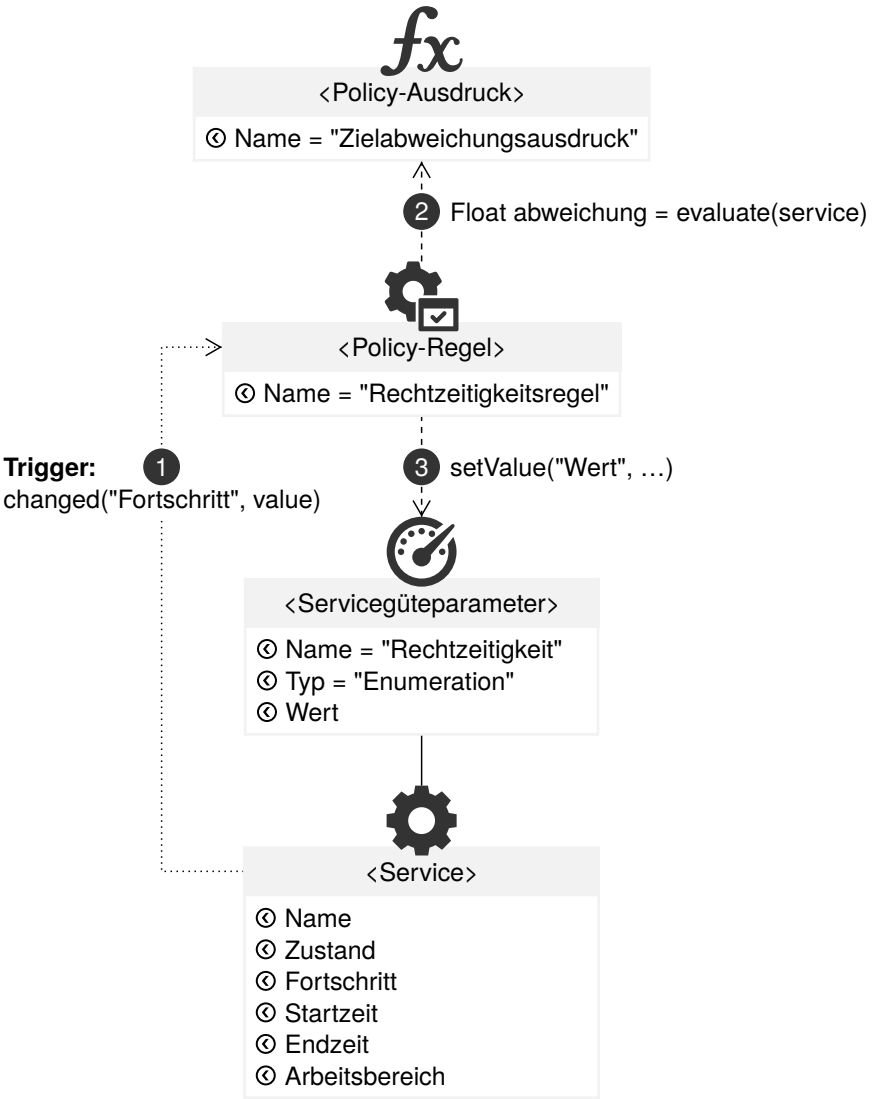


Abbildung 6.37: Rechtzeitigkeitsregel

Serviceausführung zu setzen. Dies bewirkt die Auswertung und gegebenenfalls die Ausführung der Migrationsregel. Die Funktionsweise der Rechtzeitigkeitsregel ist folgende:

1. Die Policy-Regel wird durch die Änderung der Statusvariablen angestoßen, die den Fortschritt der Auftragsverarbeitung repräsentiert. Die Policy-Regel verfügt über keine gesonderte Bedingung, so dass jede Wertänderung unmittelbar die Ausführung der Regelaktion bewirkt.
2. Im Zuge der Aktionsausführung fordert die Rechtzeitigkeitsregel den Zielabweichungsausdruck zur Abschätzung der Zielverfehlung auf. Als Eingabe dient die Referenz auf den ereignisauslösenden Service *s*.
3. Für das Treffen von Steuerungsentscheidungen ist eine Bewertung der Zielabweichung erforderlich. Dazu muss der Zahlenwert auf den Wertebereich des Servicegüteparameters abgebildet werden. Die Intervalleinteilung ist davon abhängig, ob es sich um harte oder weiche Vorgaben handelt und als wie schwerwiegend eine Zielverfehlung gilt. Gemäß dieser Abbildungsvorschrift erfolgt eine Aktualisierung des Servicegüteparameters.

Zusammenfassend lässt sich die Rechtzeitigkeitsregel wie folgt darstellen:

Ereignis	Der Wert der Statusvariablen zur Erfassung des Verarbeitungsfortschritts wurde geändert.
Bedingung	Keine
Aktion	• Fordere durch Auswertung des Zielabweichungsausdrucks eine Abschätzung der Zielverfehlung der Serviceausführung an. • Bewerte die prozentuale Zielabweichung. • Aktualisiere den Wert des Servicegüteparameters zur Erfassung der Rechtzeitigkeit der Serviceausführung.

6.11.2.3 Fortschrittsregel

Die Rechtzeitigkeitsregel wird um eine Fortschrittsregel ergänzt, die in der Abbildung 6.38 dargestellt ist. Bei der Fortschrittsregel handelt sich um eine Policy-Regel, die periodisch den Wert der Zielabweichung neu setzt. Da die Rechtzeitigkeitsregel ausschließlich durch eine Fortschrittsänderung angestoßen wird, würde ein ausbleibender Fortschritt sich nicht im Wert des Servicegüteparameters niederschlagen. Dadurch bliebe dem Managementprozess eine kritische Situation verborgen. Sie kann auftreten, wenn beispielsweise der Service abstürzt oder sich aus unvorhersehbaren Gründen der Fortschrittswert nicht mehr ermitteln lässt. Durch die zeitgesteuerte Ausführung der Fortschrittsregel ist sichergestellt, dass auch ein ausbleibender Fortschritt sich nach einer gewissen Zeitspanne auf das Qualitätsmaß auswirkt. Auf Grundlage der Bearbeitungszeit steht dem Managementprozess ein zeitbasierter Ereignisgeber zur Verfügung, der dafür sorgt, dass ein laufender und damit aktiver Verarbeitungsvorgang zugleich auch durch eine Fortschrittsregel abgesichert ist. Die Funktionsweise der Policy-Regel ist folgende:

1. Die Fortschrittsregel wird durch die Wertänderung des Servicegüteparameters angestoßen, der die Bearbeitungszeit repräsentiert. Wie bei der Rechtzeitigkeitsregel verfügt auch die

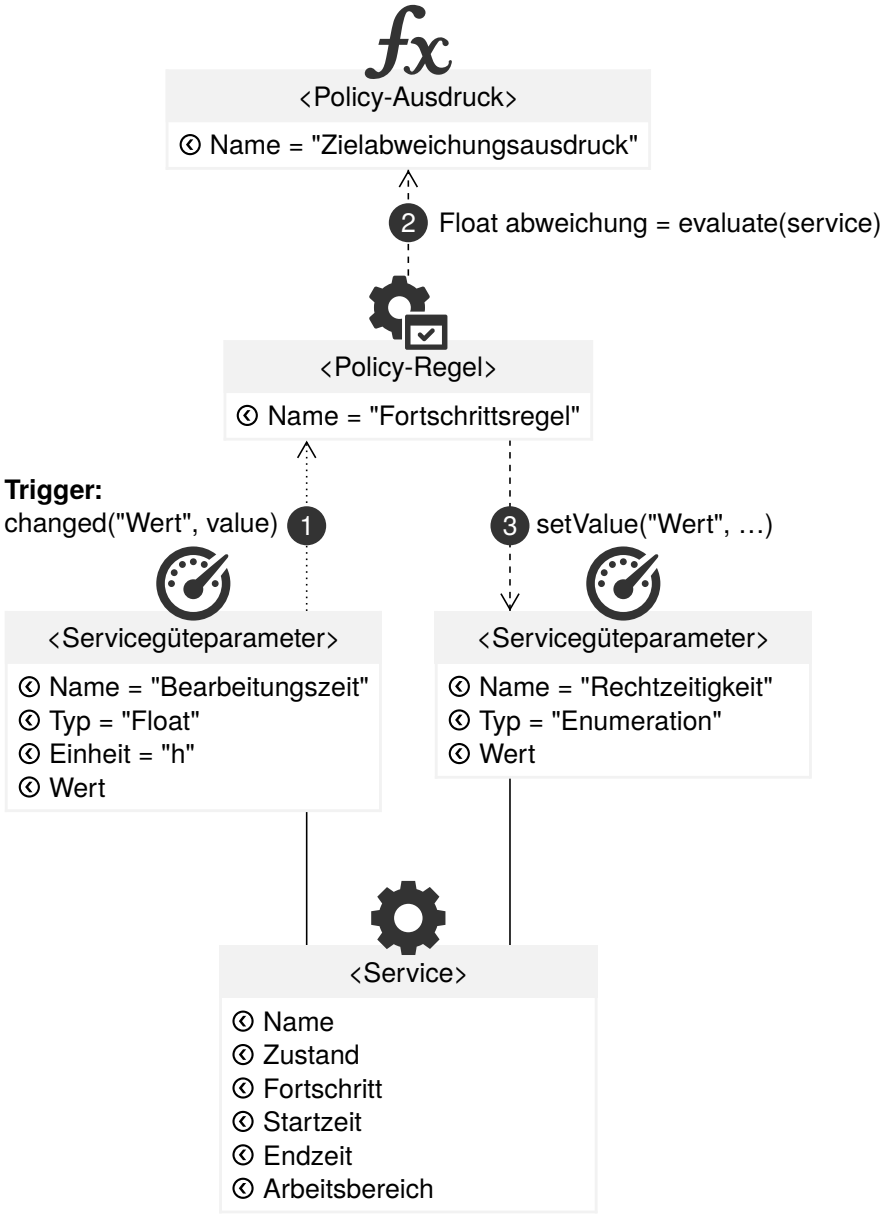


Abbildung 6.38: Fortschrittsregel

Fortschrittsregel über keine gesonderte Bedingung, so dass jede Wertänderung unmittelbar die Ausführung der Regelaktion bewirkt.

2 Im Zuge der Aktionsausführung fordert die Policy-Regel den Zielabweichungsausdruck zur Abschätzung der prozentualen Zielverfehlung auf. Als Eingabe dient der mit dem Servicegüteparameter assoziierte Service *s*.

3 Gemäß einer vorab festgelegten Intervalleinteilung nimmt die Fortschrittsregel eine Aktualisierung des Servicegüteparameters vor.

Zusammenfassend lässt sich die Fortschrittsregel wie folgt darstellen:

Ereignis	Der Wert des Servicegüteparameters zur Erfassung der Bearbeitungszeit wurde geändert.
Bedingung	Keine
Aktion	• Fordere durch Auswertung des Zielabweichungsausdrucks eine Abschätzung der Zielverfehlung der Serviceausführung an. • Bewerte die prozentuale Zielabweichung. • Aktualisiere den Wert des Servicegüteparameters zur Erfassung der Rechtzeitigkeit der Serviceausführung.

7 Managementsystem

Das Management einer Multi-Provider-Umgebung erfordert den Einsatz eines Managementsystems, das den fehlerfreien Servicebetrieb innerhalb des Umgebungsverbunds sicherstellt. Dazu gilt es, die vom Informationsmodell festgelegten Managementobjekte zur Laufzeit bereitzustellen und zu verwalten. Die Umsetzung wird dadurch erschwert, dass zur Überwachung und Steuerung nicht nur eine einzelne Managementarchitektur zum Einsatz kommt, sondern zumeist mehrere. Die Managementobjekte gehören jeweils unterschiedlichen Managementarchitekturen an. Zudem verwenden die Managementsysteme und -agenten unterschiedliche Beschreibungssprachen und Protokolle, wobei sowohl standardisierte als auch proprietäre Lösungen Anwendung finden. Die Bereitstellung der vereinheitlichten Managementobjekte setzt eine Datenstruktur voraus, die zur Informationsintegration verschiedenste Datenquellen anbinden kann. Dadurch lässt sich die technische Heterogenität verdecken und eine einheitliche Sicht auf die Managementinformationen erzeugen. Es sind insbesondere die Steuerungsmuster, die mit dieser Sicht interagieren und durch Einwirken auf die Managementobjekte den Servicebetrieb im Umgebungsverbund kontrollieren.

7.1 Architektur

Zum Einsatz kommt ein verteilter *Managementbaum*, der in Grundzügen nach der Device-Management-Spezifikation der *Open Mobile Alliance* (OMA) [Ope16c] arbeitet und dessen technischer Entwurf von der Dmt-Admin-Service-Spezifikation [OSG15] der OSGi Alliance beeinflusst ist. Die Abbildung 7.1 zeigt die Architektur des Managementsystems. Es besteht aus einem Managementbaum und einem Policy-Manager. Der Managementbaum entspricht einem gewurzelten Baum, in dem ein Knoten als Wurzel ausgezeichnet ist und von dem aus sich jeder andere Knoten erreichen lässt. Der Managementbaum ist hierarchisch organisiert und stellt über sogenannte *Data-/Execute-Handler* die Managementobjekte in Form von Teilbäumen zur Verfügung. Die Data-/Execute-Handler übernehmen die protokoll- und implementierungsspezifische Datenabfrage und leiten Modifikationen an die Datenquelle weiter. Zur Interaktion mit dem Managementbaum stehen Pull- und Push-Services zur Verfügung, die nach dem Request-Response- bzw. Publish-Subscribe-Prinzip arbeiten. Sie bieten die Möglichkeit der Abfrage

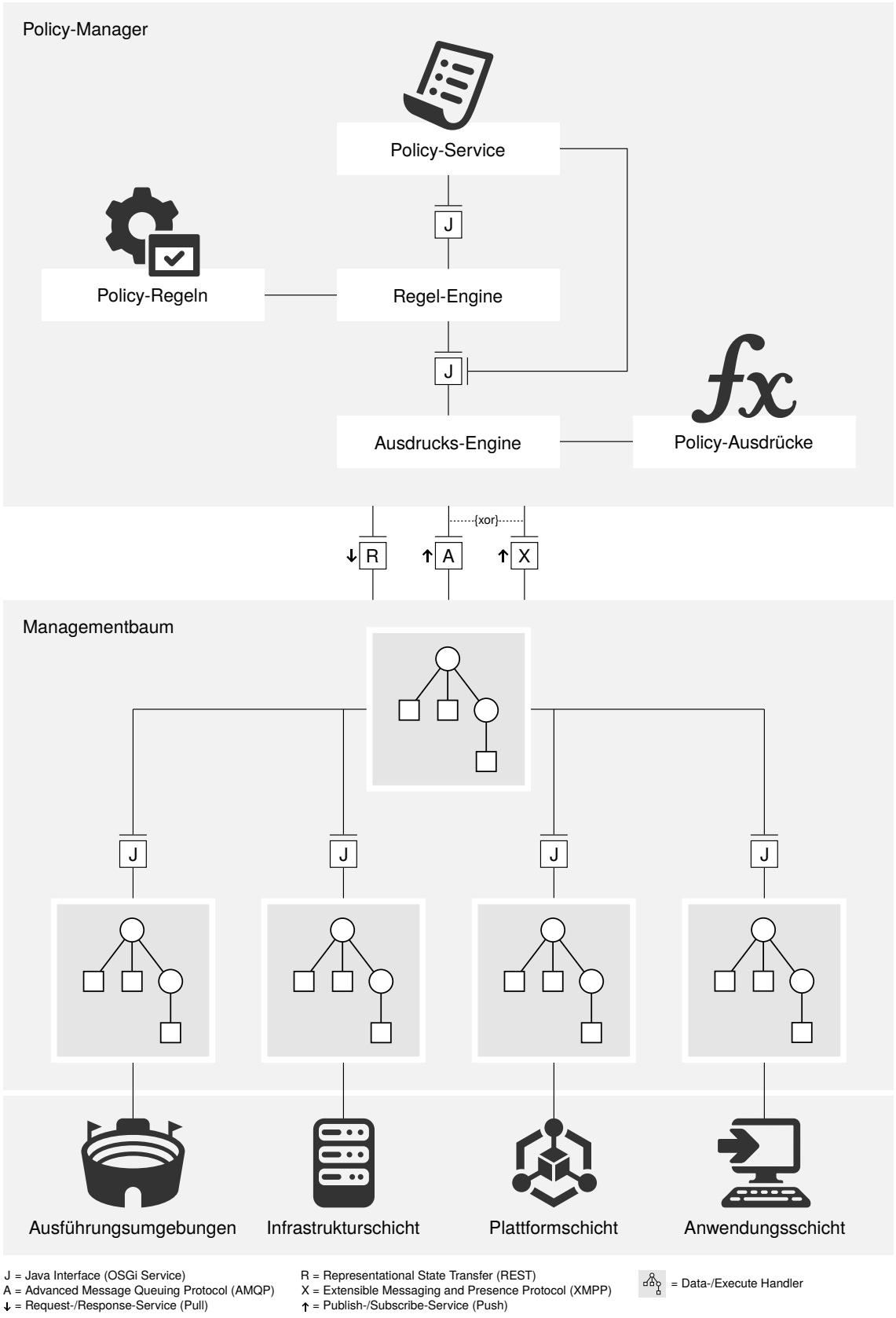


Abbildung 7.1: Architektur des Managementsystems

sowie der ereignisbasierten Benachrichtigung bei Änderung. Dadurch sind Managementservices und -anwendungen in der Lage, mit dem Managementbaum zu interagieren. Ein Großteil der Überwachungs- und Steuerungslogik ist innerhalb von Policys implementiert, die von einem Policy-Manager verwaltet und ausgeführt werden. Dabei können sie die über den Managementbaum zugänglichen Informationen abfragen und verändern. Die Regel-Engine verantwortet die Policy-Regeln, die Ausdrucks-Engine die Policy-Ausdrücke.

7.2 Managementbaum

Der Managementbaum dient der Informationsintegration und damit der Zusammenführung von Managementinformationen aus heterogenen Datenquellen in eine übergeordnete Zugriffsstruktur. Data-/Execute-Handler arbeiten nach dem Prinzip der logischen Integration und sind zustandslos. Ihre Aufgabe besteht darin, die Managementinformationen in eine hierarchische Baumstruktur zu überführen und Änderungen auf die von der Datenquelle bereitgestellten Operationen abzubilden. Der Managementbaum bietet eine einheitliche Sicht auf die Managementobjekte und kann zur Laufzeit um zusätzliche Data-/Execute-Handler erweitert werden. Diese kapseln die technischen Details der Datenabfrage und -manipulation und stellen die Managementinformationen vereinheitlicht als gewurzelten Teilbaum zur Verfügung. Die Teilbäume werden untereinander so angeordnet, dass sie einen zusammenhängenden Gesamtbaum bilden. Dieser lässt sich über eine Schnittstelle traversieren, abfragen und verändern. Zu den Aufgaben des Managementbaums zählen die Weiterleitung der Anfragen an die jeweils verantwortlichen Data-/Execute-Handler sowie die Verwaltung ihrer hierarchischen Anordnung.

7.2.1 Objektbasierung

Die Abbildung 7.2 veranschaulicht den Zusammenhang zwischen Ressourcen, Managementobjekten und Teilbäumen. Ressourcen werden im Informationsmodell durch Managementobjekte repräsentiert, die zur Laufzeit durch Teilbäume beschrieben werden. Der Übergang zwischen zwei Managementobjekten ist fließend, die Objektgrenzen werden nicht explizit in der Datenstruktur erfasst. Der Managementbaum ist objektbasiert, jedoch nicht objektorientiert. Zugriffe

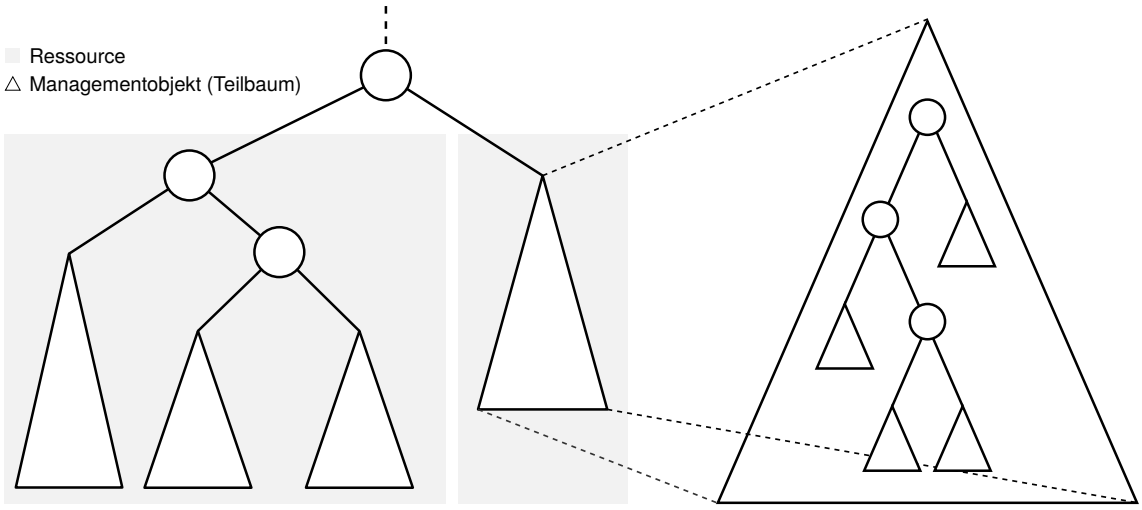


Abbildung 7.2: Managementobjekte als Teilbäume innerhalb des Managementbaums

und Veränderungen erfolgen nicht auf Managementobjekten, sondern auf Baumknoten. Des Weiteren werden Vererbungsbeziehungen nicht repräsentiert. Es ergibt sich eine hierarchische Struktur, in der Teilbäume anderen Teilbäumen untergeordnet sind [ISO89].

7.2.2 Knoten

Ein Teilbaum bildet das durch ihn repräsentierte Managementobjekt mit Hilfe von Baumknoten ab. Diese stehen untereinander in einer Eltern-Kind-Beziehung. Ein Knoten lässt sich eindeutig einer der beiden folgenden Gruppen zuordnen:

Strukturknoten *Strukturknoten* entsprechen inneren Knoten und verfügen über keinerlei Wertebelegungen. Sie dienen der Strukturdefinition und können eine unbeschränkte Anzahl direkter Nachfolgeknoten besitzen.

Blattknoten Im Gegensatz dazu verfügen *Blattknoten* über keine direkten Nachfolgeknoten, sie können aber einen Datenwert aufnehmen und bieten Zugang zu den Managementinformationen.

7.2.2.1 Eigenschaften

Struktur- und Blattknoten verfügen über eine Reihe von Eigenschaften, die lesbar und teilweise auch schreibbar sind. Dazu zählen:

Name Der Name bezeichnet einen Knoten eindeutig unter seinen Geschwistern.

Titel Der Titel eines Knotens entspricht einer menschenlesbaren Bezeichnung, die sich von seinem Namen unterscheiden kann.

ACL Die *Zugriffssteuerungsliste* (engl. *Access Control List*, ACL) legt die für den Knoten geltenden Zugriffsrechte fest und dient der Autorisation.

Version Die Versionsnummer beginnt bei 0 und wird mit jeder Veränderung inkrementiert, dazu zählen das Hinzufügen und Entfernen von Knoten sowie das Setzen von Eigenschaftswerten.

Zeitstempel Der Zeitstempel markiert den Zeitpunkt der zuletzt durchgeführten Änderung.

Schema Der Eigenschaftswert benennt ein Schema, das Aufbau und Struktur des an diesem Knoten beginnenden Teilbaums definiert.

Datentyp Der Eigenschaftswert bezieht sich auf den Datentypen des Knotens.

MIME-Type Der MIME-Type dient der Klassifikation des Datenwerts und macht Angaben zu Inhalt, Datenformat und Kodierung.

Datenwert Der Datenwert ermöglicht das Lesen und Setzen der Managementinformation.

7.2.2.2 Adressierung

Innerhalb des Managementbaums muss sich jeder Knoten eindeutig identifizieren und adressieren lassen. Zu diesem Zweck wird ein *Pfad* oder *Knotenpfad* verwendet. Anhand eines Pfades kann der Managementbaum von der Wurzel ausgehend durchlaufen und ein Knoten aufgesucht

werden. *Absolute Pfade* referenzieren einen Knoten ausgehend vom Wurzelknoten. *Relative Pfade* enthalten die Bezeichnungen aller Knoten in Bezug zu einem Referenzknoten. Syntax und Aufbau folgen den Pfadnamen in UNIX-/Linux-Dateisystemen [NSH+17].

7.2.2.3 Datentypen

Jedem Knoten ist ein Datentyp zugeordnet. Es kann zwischen skalaren und komplexen Datentypen unterschieden werden. Skalare Datentypen umfassen Ganzzahlen, Fließkommazahlen, Wahrheitswerte, Zeitangaben, Zeichen und Verweise, komplexe Datentypen hingegen Tupel, Sequenzen und Mappings (vgl. [OSG15]).

7.2.2.4 Operationen

Der Managementbaum verwendet *CRUD* (Create, Read, Update, Delete) als Abstraktionsprinzip für Managementoperationen. Knoten können erzeugt, ausgelesen, verändert, gelöscht und ausgeführt werden. Eine Operation lässt sich nur dann ausführen, falls der Knoten die Operation unterstützt und falls der Anwender über die notwendigen Berechtigungen verfügt. Das Schema gibt vor, welche Operationen auf einem Knoten ausführbar sind, und der Managementbaum verwaltet die knotenspezifischen Berechtigungen.

7.2.3 Enthaltenseinsbaum

Um ein objektorientiertes Informationsmodell, das die in Abschnitt 6.1 beschriebene Grundstruktur aufweist, in eine hierarchische Baumstruktur überführen zu können, ist die Definition eines *Enthaltenseinsbaums* erforderlich. Dieser ordnet die Managementobjektklassen innerhalb einer Enthaltenseinshierarchie an und gibt die Knotenhierarchie für den Managementbaum vor. Dieses Vorgehen hat den Vorteil, dass das Informationsmodell keinerlei Abhängigkeiten zu einer bestimmten Managementarchitektur aufweist und sich auch in andere Lösungen, wie WBEM/CIM (objektorientiert) oder SNMP (objektbasiert), integrieren ließe. Voraussetzung dafür sind Abbildungsvorschriften, gemäß derer sich die Managementobjektklassen und Assoziationen aus dem Informationsmodell in die Zielstruktur überführen lassen.

7.2.3.1 Managementobjekte

Zur Integration des Informationsmodells in den Managementbaum wird ein Managementobjekt durch einen Strukturknoten repräsentiert. Dieser bildet die Wurzel eines Teilbaums, dem die Status- und Konfigurationsvariablen als Blattknoten untergeordnet sind. Datenwerte von Blattknoten, die Statusvariablen entsprechen, sind von außen nur lesbar, wohingegen Datenwerte von Blattknoten, die Konfigurationsvariablen entsprechen, sowohl lesbar als auch schreibbar sind. Managementobjekte, die über keinen eigenständigen Lebenszyklus verfügen, lassen sich als Teilbäume an das jeweils übergeordnete Managementobjekt bzw. den entsprechenden Strukturknoten hängen. Die verbleibenden Managementobjekte werden als typisierte Sequenzen der Wurzel untergeordnet. Diese repräsentiert die Multi-Provider-Umgebung. Ihr sind diejenigen Teilbäume untergeordnet, die die Ausführungsumgebungen beschreiben. Diese enthalten Projekte, die ihrerseits Managementobjekte der Infrastruktur-, Plattform- und Anwendungsschicht umfassen. Das Informationsmodell verfügt bereits über eine natürliche Enthaltenseinshierarchie. Zur Veranschaulichung skizziert die Abbildung 7.3 den Aufbau des Managementbaums eines Projekts. Die Infrastrukturschicht ist zur Verbesserung der Strukturierung zusätzlich in einen Storage-, Compute- und Networking-Teilbaum unterteilt. Die Teilbäume

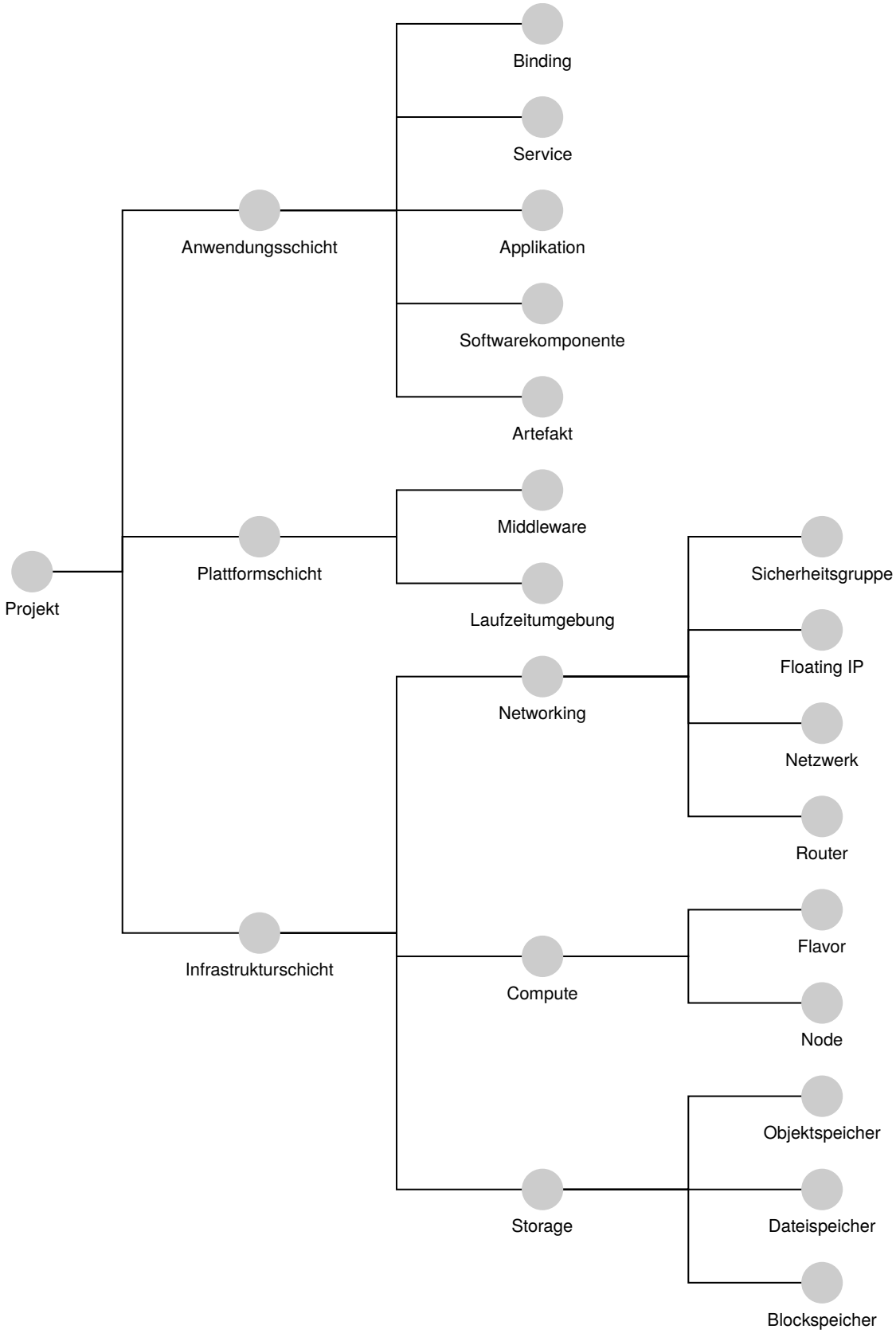


Abbildung 7.3: Managementbaum eines Projekts

beinhalten die Managementobjekte für Objekt-, Datei- und Blockspeicher sowie für Nodes, Flavors, Router, Netzwerke, Floating IPs und Sicherheitsgruppen. Die Plattformschicht umfasst die Managementobjekte für Laufzeitumgebungen und Middleware, die Anwendungsschicht die für Artefakte, Softwarekomponenten, Applikationen, Services und Bindungen.

7.2.3.2 Vererbungshierarchien

Vererbungshierarchien sind im Managementbaum nicht explizit repräsentiert. Managementobjekte, die Teil derselben Vererbungshierarchie sind, werden standardmäßig auf einer Baumebene zusammengefasst und als Ausprägung der obersten Managementobjektklasse aufgefasst. Es obliegt den Managementobjekten Informationen bereitzustellen, die eine eindeutige Typzuordnung ermöglichen, sollte dies zur Laufzeit erforderlich sein. Dieses Vorgehen gewährleistet, dass sich Objekte polymorph behandeln lassen. Fallspezifische Anpassungen sind aber möglich. Dazu muss der Enthaltenseinsbaum um gruppierende Strukturknoten ergänzt werden, die die Vererbungshierarchie nachbilden. Auch in diesem Fall lässt sich die Polymorphie aufrechterhalten, erfordert aber Traversierungen, da sich die Managementobjekte nicht mehr länger auf nur einer einzelnen Baumebene befinden.

7.2.3.3 Assoziationen

Assoziationen zwischen Managementobjekten, bei denen es sich nicht um Kompositionsbeziehungen handelt, sind in Abhängigkeit zur Navigationsrichtung von einem bzw. beiden Assoziationsenden zu verwalten. Dazu kommen Blattknoten zum Einsatz, die als Verweis eine absolute Pfadangabe zum assoziierten Managementobjekt bzw. der Position seines Strukturknotens in der Knotenhierarchie enthalten. Ist eine Assoziation zu mehreren Managementobjekten möglich, wird auf eine Sequenz zurückgegriffen.

7.2.4 Ereignisse

Neben dem Nachrichtenaustausch nach dem Request-Response-Prinzip besteht auch die Möglichkeit, sich nach dem Publish-Subscribe-Prinzip synchron und asynchron über Änderungen innerhalb des Managementbaums informieren zu lassen. Um ein Ereignis an einem Knoten empfangen zu können, muss der Empfänger am ereignisauslösenden Knoten bzw. an einem seiner übergeordneten Teilbäume angemeldet sein und über entsprechende Berechtigungen verfügen. Ereignisse lassen sich in zwei Gruppen einteilen. Die eine Gruppe umfasst diejenigen Ereignisse, die im Zusammenhang mit dem Lebenszyklus einer Sitzung stehen und den Empfänger über das Öffnen und Schließen einer Sitzung informieren. Die andere Gruppe umfasst Ereignisse, die sich auf Struktur- bzw. Eigenschaftsveränderungen beziehen und den Empfänger über das Hinzufügen, Löschen und Ändern eines Knotens informieren.

Eigenschaften

Zur genaueren Beschreibung der Zustandsänderung verfügt ein Ereignis über allgemeine sowie typspezifische Eigenschaften. Allgemeine Eigenschaften beinhalten den Knotenpfad, den Ereignistypen, den Zeitpunkt der Ereigniserzeugung und den Sitzungsidentifikator, typspezifische Eigenschaften machen nähere Angaben über die ereignisauslösende Operation (vgl. [OSG15]). Bei einer Registrierung lassen sich neben Pfaden auch ein Filterausdruck angeben. Die inhaltsbasierte Filterung gestattet es, Ereignisse vor der Zustellung auszufiltern. Dazu werden die Eigenschaften als Menge von Attribut-Wert-Paaren aufgefasst und Prädikate definiert, die mittels

boolescher Operatoren verbunden werden. Der Filterausdruck entspricht einer Zeichenkette und basiert auf dem RFC 1960 [How96].

7.2.5 Metadaten

Metadaten stellen beschreibende Daten über Knoten zur Verfügung und besitzen einen eigenständigen Lebenszyklus. Sie sind für struktur- und wertverändernde Operationen von zentraler Bedeutung und ermöglichen es, Anfragen vor ihrer Durchführung zu validieren [OSG15]. Dabei handelt es sich unter anderem um:

Name Der Name gibt die gültige Knotenbezeichnung in Form eines regulären Ausdrucks vor.

Operationen Die Operationen geben an, ob der Knoten das Hinzufügen, Auslesen, Verändern, Löschen und Ausführen unterstützt.

Multiplizität Die Multiplizität legt die minimal und maximal zulässige Knotenanzahl innerhalb der Ebene fest.

Knotentyp Der Knotentyp gibt an, ob es sich bei dem Knoten um einen Struktur- oder Blattknoten handelt.

Standardwerte Die Standardwerte dienen der initialen Belegung von Blattknoten.

7.2.6 Schema

Ein Schema gibt die zu Managementzwecken erforderlichen Knoten vor und beschreibt die gültigen Ausprägungen des Managementbaums. Neben dem strukturellen Aufbau der Knotenhierarchie spezifiziert es die Struktur- und Blattknoten. Die Definition des Schemas kann auf zwei Arten erfolgen. Zum einen deklarativ mittels textueller Beschreibung, zum anderen imperativ mittels ausführbarem Programmcode. Die deklarative Art setzt eine Beschreibungssprache voraus. Das entsprechende Dokument wird als *Domain Description File* (DDF) bezeichnet. Die deklarative Form hat den Vorteil, dass sich durch Austauschen der DDF das Managementsystem ohne größeren Aufwand an unterschiedliche Nutzungsszenarien anpassen lässt. Der Managementbaum unterstützt die von der OMA-DM-Device-Description-Framework-Spezifikation vorgegebene Schemastruktur [Ope16b]. Bei der programmatischen Umsetzung wird das Schema mit Hilfe von Programmcode direkt von den Data-/Execute-Handlern festgelegt.

7.2.7 Nebenläufkeitskontrolle

Der Managementbaum steht in der Regel zu keinem Zeitpunkt nur einem einzelnen Nutzer exklusiv zur Verfügung, so dass zur Konsistenzerhaltung bei gleichzeitigen Lese- und Schreibzugriffen Mechanismen zur Nebenläufigkeitskontrolle erforderlich sind.

7.2.7.1 Sperrverfahren

In diesem Zusammenhang kommt ein pessimistisches Sperrverfahren zum Einsatz, das im Zuge der Sitzungserstellung parallel stattfindende Lese- und Schreibzugriffe in sich überlappenden Teilbäumen verhindert. Es lassen sich die folgenden Sperrtypen unterscheiden:

Nicht-exklusive Sperre Eine nicht-exklusive Sperre wird immer dann gewährt, falls in keinem der sich überlappenden Teilbäume eine exklusive Sperre gewährt wurde, d. h. es dürfen

entweder keine Sperren oder aber ausschließlich nicht-exklusive Sperren vorliegen. Wurde ein Teilbaum durch eine exklusive Sperre gesperrt, so lässt sich vorübergehend keine nicht-exklusive Sperre einrichten. Nicht-exklusive Sperren werden immer dann angefordert, wenn eine Sitzung ausschließlich lesend auf den Managementbaum zugreift. Der gesperrte Teilbaum wird mit einer *geteilten Lesesperre* (engl. *Shared Read Lock*) belegt.

Exklusive Sperre Eine exklusive Sperre kann nur von genau einem Nutzer gleichzeitig gehalten werden. Ihre Zuweisung setzt voraus, dass keiner der untergeordneten Teilbäume mit einer Sperre versehen ist. Eine exklusive Sperre ist für einen schreibenden Zugriff auf den Managementbaum erforderlich und unterbindet parallel stattfindende Sitzungen in einem der untergeordneten Teilbäume. Der gesperrte Teilbaum wird mit einer *exklusiven Schreibsperre* (engl. *Exclusive Write Lock*) belegt.

7.2.7.2 Sperrgranularität

Je feiner die Sperrgranularität gewählt wird, umso höher ist der Verwaltungsaufwand zur Gewährung und Freigabe von Sperren. Im Rahmen eines Minimalverfahrens dient der gesamte Managementbaum als Sperrgranulat, hierbei sperrt eine exklusive Sperre diesen vollständig. Das Verfahren schließt parallel stattfindende Schreibvorgänge gänzlich aus, auch wenn diese im Falle disjunkter Teilbäume grundsätzlich unbeeinflusst voneinander ausführbar wären. Es zeichnet sich durch einen geringen Verwaltungsaufwand aus, da sich Sperren effizient verwalten, gewähren und wieder freigeben lassen. Hierarchische Sperrverfahren stellen einen alternativen Lösungsansatz dar und bieten einen Kompromiss zwischen Parallelität und Verwaltungsaufwand [GLP75, GLPT94].

7.2.7.3 Sitzungen

Ein Baumzugriff erfolgt ausschließlich im Kontext einer gültigen und aktiven Sitzung. Für deren Erstellung ist die Angabe eines Pfades erforderlich, der den Wurzelknoten der Sitzung bildet. Dieser Knoten wird als *Sitzungswurzel* bezeichnet. Er ist unveränderlich mit der Sitzung verbunden und bildet für Interaktionen einen Referenzpunkt innerhalb des Managementbaums. Absolute Adressen werden stets zur Sitzungswurzel relativiert, wobei jeder Zugriff, der einen Knoten außerhalb des geschützten Teilbaums referenziert, unterbunden ist.

Typen

Neben den beiden Sperrtypen existieren unterschiedliche Sitzungstypen, die jeweils spezifische Operationen zum Zugriff auf den Managementbaum bereitstellen. Nach dem Prinzip des Preclaiming ist der Sitzungstyp vor dem Aufbau festzulegen und lässt sich nachträglich nicht mehr verändern. Mit der Entscheidung geht auch der zur Interaktion erforderliche Sperrtyp einher. Dabei können Sitzungen nur dann erfolgreich aufgebaut werden, wenn die Sperre auf dem Managementbaum gewährt werden kann. Es sind folgende Sitzungstypen vorgesehen:

Nicht-exklusive Sitzung Eine nicht-exklusive Sitzung gestattet ausschließlich lesenden Zugriff und erfordert eine nicht-exklusive Sperre auf dem Teilbaum. Aufgrund der Sperrkompatibilität sind parallele nicht-exklusive Sitzungen in sich überlappenden Teilbäumen erlaubt, exklusive und transaktionale Sitzungen sind hingegen unterbunden.

Exklusive Sitzung Eine exklusive Sitzung gewährt Operationen, die Modifikationen am Managementbaum vornehmen und setzt eine exklusive Sperre auf dem Teilbaum voraus.

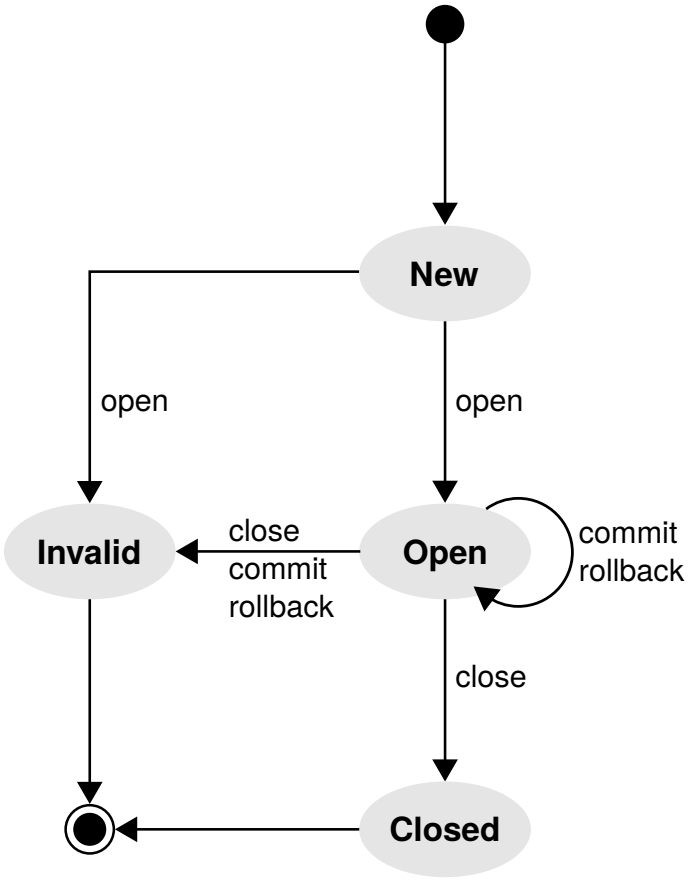


Abbildung 7.4: Lebenszyklus einer Sitzung im Managementbaum

Gleichzeitig bietet sie alle Operationen, die innerhalb einer nicht-exklusiven Sitzung möglich sind und erlaubt einen lesenden wie auch schreibenden Zugriff.

Transaktionale Sitzung Eine transaktionale Sitzung erweitert die exklusive Sitzung um die Möglichkeiten, Änderungen im Sitzungsablauf festschreiben (Commit) und wieder rückgängig machen (Rollback) zu lassen. Die sich aus dem Sitzungsverlauf ergebenden Änderungsereignisse werden verzögert und erst mit dem Schließen der Sitzung veröffentlicht.

Lebenszyklus

Eine Sitzung durchläuft von ihrer Erzeugung bis zu ihrem Ende unterschiedliche Zustände. Die Abbildung 7.4 stellt den Lebenszyklus in Form eines Zustandsübergangsdiagramms dar. Dabei ist die Nutzung der dargestellten Operationen mitunter vom Sitzungstyp abhängig. Im Folgenden werden die Zustände näher beschrieben:

New In diesem Zustand gilt die Sitzung zunächst als erzeugt, inaktiv sowie als nicht geöffnet. Sie ist der Sitzungsverwaltung noch nicht durch Registrierung bekannt gemacht worden. Wird die Sitzung geöffnet, erfolgt die Sperranfrage, die solange verzögert wird bis sich entweder die Sperre zuweisen lässt oder eine Zeitüberschreitung eintritt. Mögliche Nachfolgezustände sind **Open** und **Invalid**.

Open Eine geöffnete Sitzung ist aktiv und registriert. Die Sperre konnte der Sitzung erfolgreich zugewiesen werden. Dies gewährleistet einen lesenden bzw. schreibenden Zugriff innerhalb des Teilbaums. Zusätzlich stehen alle vom Sitzungstyp unterstützten Operationen zur Verfügung. Tritt während der Ausführung ein schwerwiegender Fehler auf, wird die Sitzung automatisch geschlossen und es erfolgt eine Transition in den Zustand **Invalid**. Gemäß Sperrkompatibilität werden parallele Sitzungsanfragen entweder blockiert oder gewährt. Der Schließvorgang selbst kann ebenfalls eine Fehlersituation hervorrufen, dementsprechend sind **Closed** und **Invalid** mögliche Nachfolgezustände.

Closed Eine geschlossene Sitzung gilt als inaktiv und jeder Operationsaufruf führt unmittelbar zu einem Fehler, da sich aufgrund fehlender Sperrzuweisungen kein konsistenter Zugriff auf den Managementbaum gewährleisten lässt. Prinzipiell könnte, ähnlich wie im Zustand **New**, automatisch eine neue Sperranfrage gestellt werden. Dies widerspricht jedoch dem Prinzip, dass eine Sitzung eine in sich abgeschlossene logische Arbeitseinheit darstellt. Ihre Wiederaufnahme ist als neue Arbeitseinheit aufzufassen und muss innerhalb einer neuen Sitzung erfolgen.

Invalid Dieser Zustand entspricht dem eines Fehlerzustands, der aufgrund einer fatalen Ausnahmesituation während des Öffnens, Schließens oder Nutzens eingetreten ist. Eine solche Sitzung wird ebenfalls als inaktiv klassifiziert und erlaubt keine weiteren Zugriffe auf den Managementbaum. Im Gegensatz zum Zustand **Closed** signalisiert **Invalid**, dass die Arbeitseinheit aufgrund einer Fehlersituation nicht erfolgreich abgeschlossen werden konnte. Je nach Ausnahme und Schweregrad müssen korrigierende Maßnahmen ergriffen werden. Das Scheitern der Commit- und Rollback-Anweisung stellt einen solch schwerwiegenden Fehler dar, der mitunter Inkonsistenzen in der Datenstruktur zur Folge hat.

7.2.8 Zugriffskontrolle

Jeder Knoten bzw. Teilbaum kann durch eine ACL geschützt werden, die dem Whitelist-Ansatz folgt und einer Positivliste entspricht. Die ACL umfasst eine Menge von Paaren, wobei jedes Paar aus einer Operation und einer Liste von Principals besteht. Es lassen sich die Rechte zum Hinzufügen, Auslesen, Verändern, Löschen und Ausführen unterscheiden. Einem Subjekt wird nach erfolgreich durchgeführter Authentifikation eine Menge von Principals zugewiesen. Bei einer rollenbasierten Zugriffskontrolle entspricht ein Principal derjenigen Rolle, in der das Subjekt im System tätig ist. Eine ACL ist stets an einen Pfad gebunden und gibt die Berechtigungen zum Zugriff des Knotens vor. Um die Zugriffssteuerung leicht anpassen zu können, kommt das Prinzip der Vererbung zur Anwendung. Eine *effektive ACL* (engl. effective ACL) entspricht dabei entweder einer explizit definierten oder einer geerbten ACL. Verfügt ein Knoten über keine ACL, gilt die des Vorgängerknotens. Eine ACL gibt die Zugriffsrechte für den untergeordneten Teilbaum vor. Sie kann grundsätzlich an jeder Stelle überschrieben und gänzlich neu festgelegt werden (vgl. [OSG15]).

7.2.9 Standardkonformität

Die Grundlage der Implementierung bildet die *OSGi-Service-Plattform*, eine serviceorientierte Middleware für Java [OSG14]. Sie definiert eine Laufzeitumgebung oberhalb der JVM und bietet neben einem Versions- und Abhängigkeitsmanagement Basisdienste zur Überwachung,

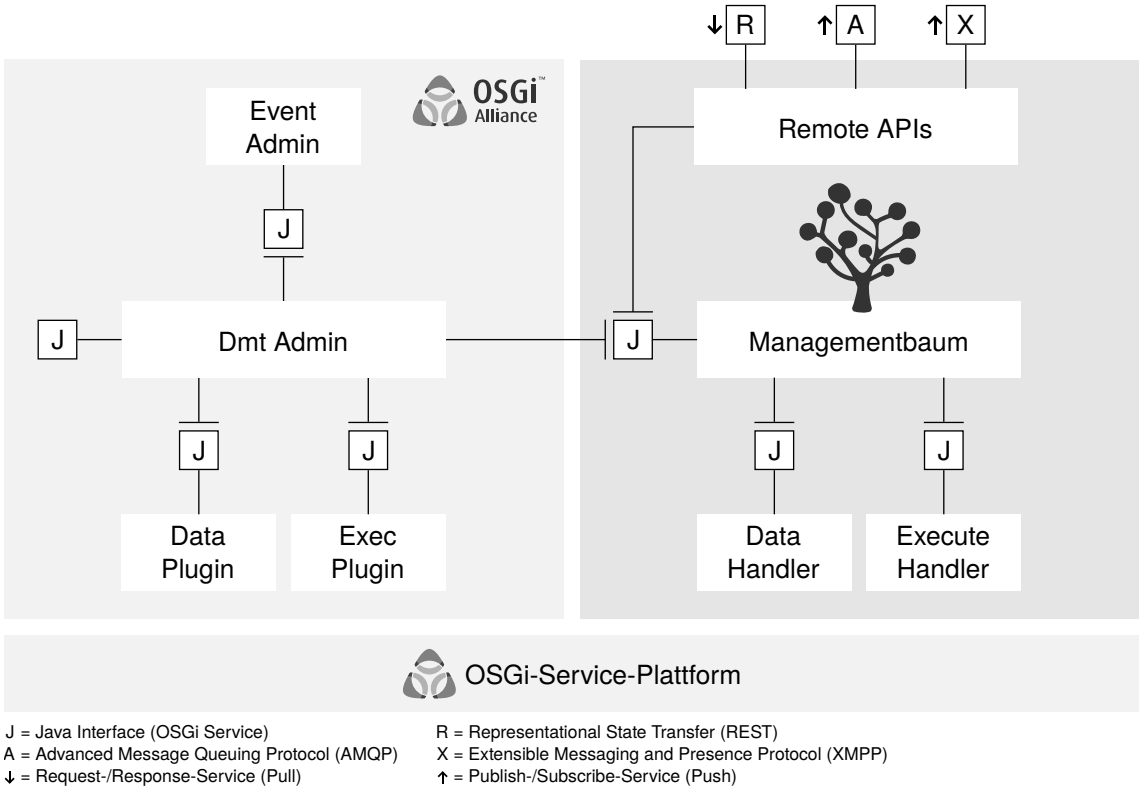


Abbildung 7.5: Technische Umsetzung des Managementsystems

Konfiguration, Servicenutzung und -bereitstellung. Die Abbildung 7.5 enthält eine Komponentensicht auf den Managementbaum. Sie zeigt einen Teilausschnitt der Softwarekomponenten und ihrer Interaktionsbeziehungen. Der Managementbaum wird mit Hilfe mehrerer Softwarekomponenten realisiert. Zur Nutzung stehen entsprechende OSGi-Services zur Verfügung. Der Managementbaum verfügt neben einer API über eine SPI (engl. *Service Provider Interface*), die von den Data-/Execute-Handlern zur Bereitstellung der Teilbäume implementiert wird. Die Anbindung an den Managementbaum erfolgt über einen vom Data-/Execute-Handler registrierten OSGi-Service. Damit ein Zugriff auf die Managementinformationen außerhalb der JVM möglich ist, lässt sich der Managementbaum über einen *RESTful Webservice* abfragen. Zudem stehen zwei alternative Implementierungen zur entfernten Ereignisbenachrichtigung zur Verfügung. Verwendung finden das *Advanced Message Queuing Protocol* (AMQP) sowie das *Extensible Messaging and Presence Protocol* (XMPP). Um den standardkonformen Zugriff auf die Managementinformationen innerhalb der OSGi-Service-Plattform zu gewährleisten, steht eine Adapter-Implementierung zur Verfügung, die konform zur Dmt-Admin-Service-Spezifikation der OSGi Alliance ist.

7.3 Policy-Manager

Der Policy-Manager verwaltet die zur Überwachung und Steuerung vorgesehenen Policy-Regeln und Policy-Ausdrücke. Er besteht aus einem Policy-Service, der auf die Funktionen einer Regel- und Ausdrucks-Engine zurückgreift. Dabei handelt sich um einen Service, der als Adapter realisiert ist und Anfragen per Delegation an die Regel- bzw. Ausdrucks-Engine weiterleitet. Der

Policy-Service verfügt sowohl über die Fähigkeiten zur Ausführung von Policy-Regeln als auch zur Auswertung von Policy-Ausdrücken. Aufgrund der Trennung von Regel- und Ausdrucks-Engine lassen sich beide Komponenten unabhängig voneinander bereitstellen. Bei Umgebungen, die manuell verwaltet werden oder deren Überwachungs- und Steuerungslogik keinen Gebrauch von Policy-Ausdrücken macht, kann somit auf die Bereitstellung der jeweiligen Komponente verzichtet werden. Der Policy-Manager bindet sich an die Schnittstelle des Managementbaums und ermöglicht der Regel- und Ausdrucks-Engine Zugang zu den Managementobjekten.

7.3.1 Regel-Engine

Verwaltung und Ausführung der Policy-Regeln obliegt der Regel-Engine. Sie bietet eine Schnittstelle zur Registrierung und Deregistrierung der in Form von OSGi-Services implementierten Policy-Regeln. Das Lebenszyklusmanagement der Softwareplattform ermöglicht es, Policy-Regeln zur Laufzeit hinzuzufügen, auszutauschen und anzupassen. Diese können zudem dynamisch aktiviert und deaktiviert werden. Dadurch lassen sich abhängig vom Systemzustand Überwachungs- und Steuerungsfunktionen hinzuschalten und wieder entfernen.

Policy-Regeln

Policy-Regeln werden von der Regel-Engine als Ereignisempfänger am Managementbaum angemeldet. Die Grundlage dafür bilden *Path-Patterns*, deren Repräsentation sich am JSR 370 [BPG17] orientiert. Es handelt sich hierbei um reguläre Ausdrücke, die beim Pfad-Matching Anwendung finden. Der Grund für ihren Einsatz ist folgender: Policy-Regeln werden zur Entwicklungszeit erstellt. Zu diesem Zeitpunkt sind zumeist noch nicht alle Knotenbezeichnungen bekannt, viele werden dynamisch bestimmt und liegen daher erst zur Laufzeit vor. Empfängt der Policy-Manager vom Managementbaum ein Änderungsereignis, überprüft er zunächst die Path-Patterns der aktiven Policy-Regeln auf Übereinstimmung und wertet anschließend die Filterausdrücke aus. Liegen Übereinstimmungen vor, erstellt er einen Ausführungsplan. Die Ausführungsreihenfolge lässt sich über *Runlevel* beeinflussen. Runlevel sind nicht-negative ganze Zahlen, beginnend bei 0. Je niedriger das Runlevel gewählt ist, desto früher kommt eine Policy-Regel zur Ausführung. Policy-Regeln, die demselben Runlevel zugeordnet sind, gelten als unabhängig und sind nebenläufig ausführbar. Erst wenn alle Policy-Regeln eines Runlevels terminiert sind, fährt die Regel-Engine mit der Bearbeitung des nächsthöheren Runlevels fort.

Lebenszyklus

Die Abbildung 7.6 stellt den Lebenszyklus einer Policy-Regel in Form eines Zustandsübergangsdiagramms dar. Es legt die folgenden Zustände fest:

Registered Eine Policy-Regel befindet sich unmittelbar nach ihrer Bereitstellung im Zustand der Registrierung. Sie gilt als inaktiv und kann weder Ereignisse verarbeiten noch von der Regel-Engine zur Ausführung gebracht werden. Nachfolgezustände sind **Unregistered** und **Enabled**.

Unregistered Wird eine Policy-Regel nicht mehr länger benötigt oder soll diese durch eine fehlerbereinigte Version ersetzt werden, kann sie zur Laufzeit von der Regel-Engine abgemeldet werden.

Enabled In diesem Zustand befinden sich alle aktiven Policy-Regeln, die derzeit nicht ausgeführt werden. Nachfolgezustände sind **Disabled** und **Triggered**.

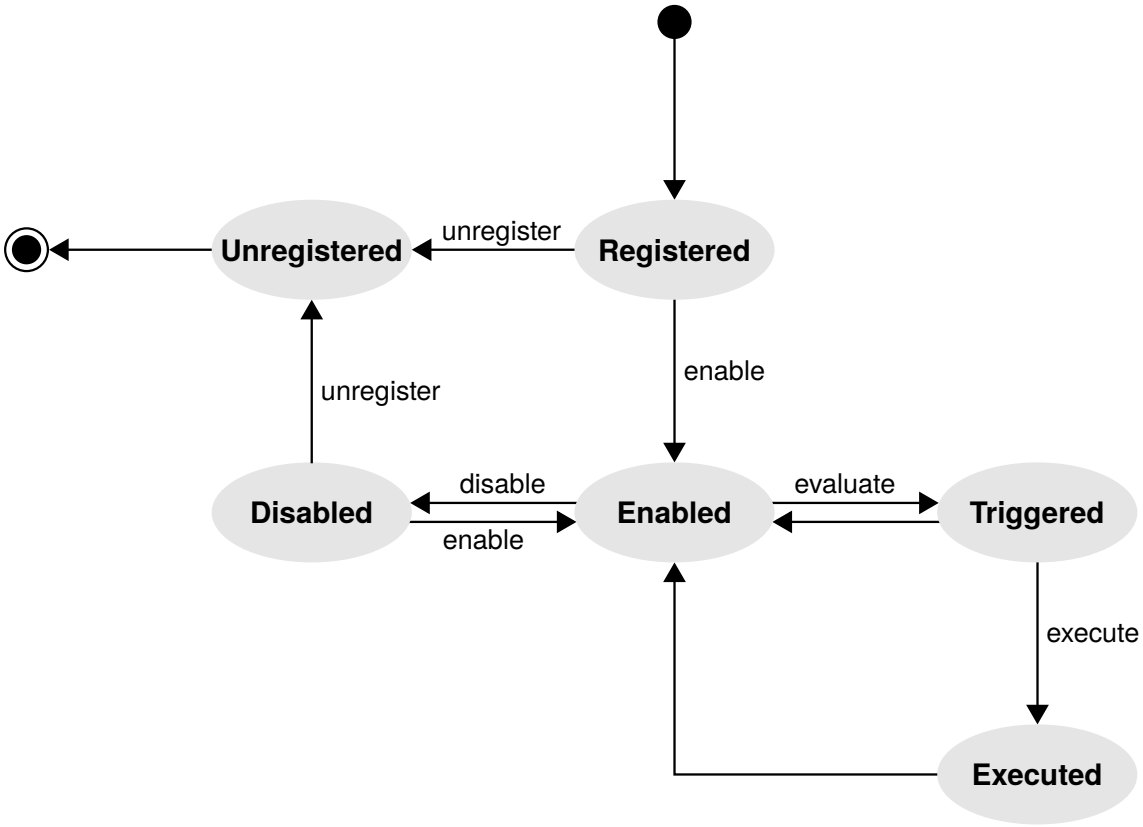


Abbildung 7.6: Lebenszyklus einer Policy-Regel

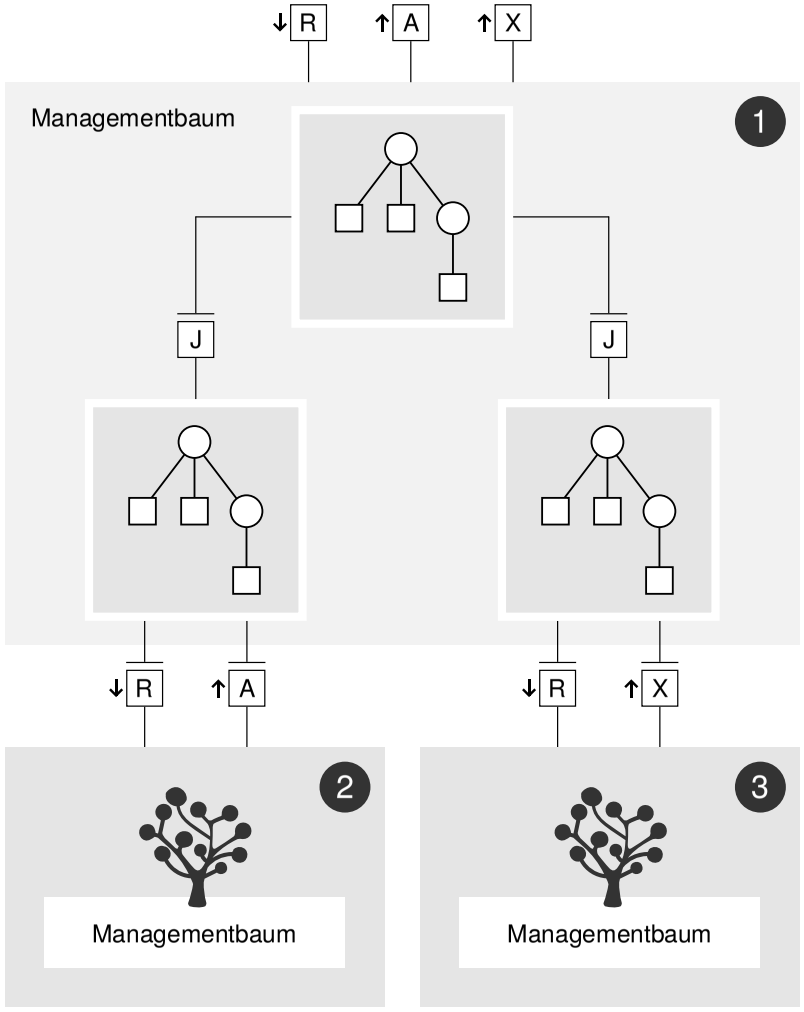
Disabled Eine inaktive Policy-Regel ist zwar weiterhin Teil der Regelmenge, ihre Ausführung ist aber nicht möglich. Nachfolgezustände sind **Unregistered** und **Enabled**.

Triggered Kommt eine Policy-Regel zur Ausführung, erfolgt zunächst die Auswertung ihrer Bedingung. Dabei kann sie lesend auf die Knoten im Managementbaum zugreifen. Je nach Ergebnis der Überprüfung kehrt die Policy-Regel entweder in den Zustand **Enabled** zurück, oder aber die Ausführung der Aktion wird veranlasst.

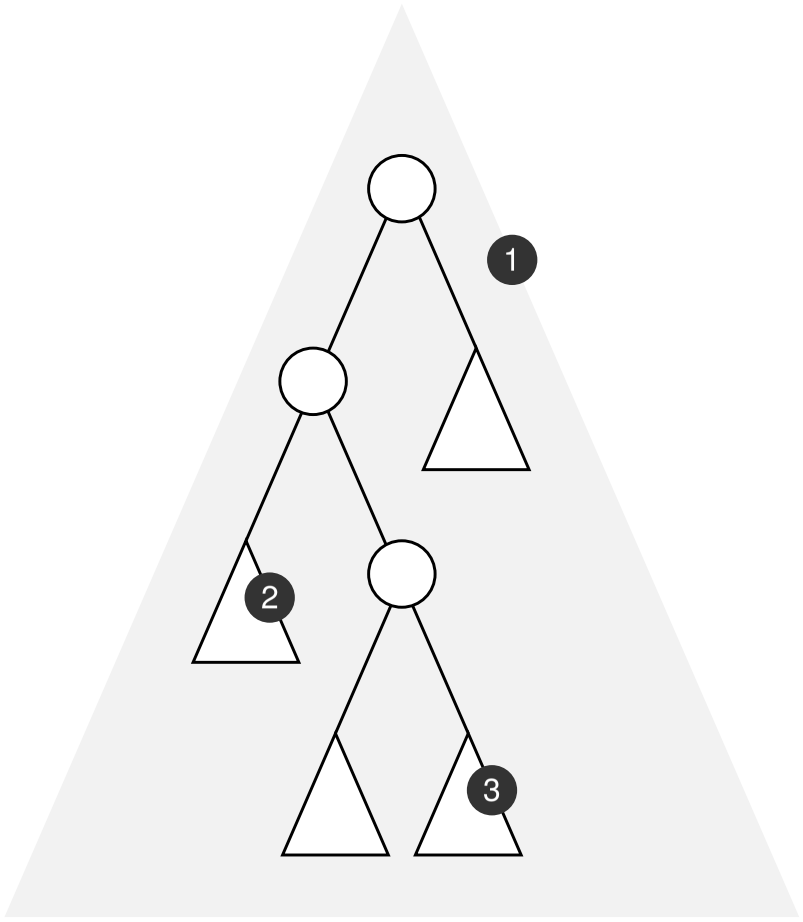
Executed Signalisiert die Policy-Regel im Zuge der Bedingungsüberprüfung die Notwendigkeit zur Aktionsausführung, wird dies von der Regel-Engine entsprechend veranlasst. Dabei kann grundsätzlich lesend und schreibend auf den Managementbaum zugegriffen werden. Unmittelbar nach der Ausführung kehrt die Policy-Regel in den Zustand **Enabled** zurück.

7.3.2 Ausdrucks-Engine

Die Ausdrucks-Engine bietet die Möglichkeit zur synchronen und asynchronen Auswertung von Policy-Ausdrücken. Zur Ergebnisberechnung können sie auf die vom Managementbaum bereitgestellten Informationen zugreifen. Abschließend wird dem Aufrufer das Ergebnis in Form eines Rückgabeparameters zur Verfügung gestellt. Auswertungsanfragen lassen sich zusätzlich mit einem Zeitlimit versehen. Gelingt es nicht, die Auswertung innerhalb der vorgegebenen Zeitspanne abzuschließen, wird diese abgebrochen und der Aufrufer darüber in Kenntnis gesetzt. Der Programmablauf innerhalb der anfragenden Komponente kann fortgesetzt und entsprechend angepasst werden.



(a) Technische Sicht



(b) Logische Sicht

Abbildung 7.7: Hierarchische Anordnung von Managementbäumen

7.4 Verteilung

Managementbäume bilden eigenständige Verwaltungsdomänen, die sich untereinander hierarchisch verschachteln lassen. Diese Aufgabe übernehmen spezielle Data-/Execute-Handler. Das Vorgehen basiert darauf, dass Managementbäume wiederum Datenquellen darstellen und über eine Schnittstelle Managementinformationen bereitstellen. Der einzige Unterschied besteht darin, dass die Managementinformationen bereits durch Managementobjekte repräsentiert sind und nicht mehr in eben solche überführt werden müssen. Der Lösungsansatz ist in der Abbildung 7.7 dargestellt. Die Abbildung 7.7a zeigt drei Managementbäume, die drei eigenständige Verwaltungsdomänen bilden. Es liegt keine zusammenhängende Baumstruktur vor, so dass der Zugriff auf die Managementobjekte Kenntnisse erfordert, in welchem der Managementbäume sich die dazugehörigen Baumknoten befinden. Übergänge zwischen den Datenstrukturen existieren nicht, so dass die Repräsentation der Ausführungsumgebung durch ein zusammenhängendes Managementobjekt nicht gegeben ist.

7.4.1 Verschachtelung

Zu dessen Herstellung werden in den übergeordneten Managementbaum zwei Data-/Execute-Handler eingebunden. Jeder dieser Data-/Execute-Handler bindet sich an die Schnittstelle des ihm zugeordneten Managementbaums und stellt dessen Knoten als Teilbaum im übergeordneten Managementbaum zur Verfügung. Anfragen, die diesen Teilbaum betreffen, werden per Delegation an den untergeordneten Managementbaum weitergeleitet. Dabei ist es auch möglich, lediglich einen Teilbaum aus dem untergeordneten Managementbaum einzubinden. Die Abbildung 7.7b zeigt die logische Sicht auf die so entstandene zusammenhängende Baumstruktur. Traversierung, Abfrage und Anpassung erfolgen über die öffentliche Schnittstelle des übergeordneten Managementbaums. Der Übergang zwischen den Datenstrukturen ist fließend und die Aufteilung der Managementobjekte auf die Managementbäume bleibt verborgen. Um die Atomarität transaktionaler Sitzungen zu gewährleisten, kommt das *Zwei-Phasen-Commit-Protokoll* (2PC) [FGJ+78, MLO86] zum Einsatz.

7.4.2 Steuerung und Überwachung

Neben der Festlegung der Verwaltungsdomänen müssen diese Policy-Managern unterstellt werden. Einer Verwaltungsdomäne kann grundsätzlich mehr als ein Policy-Manager zugeordnet sein, der für Überwachung und Steuerung verantwortlich ist. Ein Policy-Manager lässt sich aber nur genau einer Verwaltungsdomäne zuordnen. Diese Flexibilität ermöglicht es, Organisationsformen wie das Multipoint-Control, das Multicenter-Control, das hierarchische Management, das zentralisierte Management sowie ein funktional unterteiltes Management zu realisieren [HAN99]. Je nach Bedarf können auch mehrere Organisationsformen gleichzeitig zum Einsatz kommen.

7.5 Kennzahlen

Softwaremetriken liefern wichtige Kennzahlen zur Beurteilung der qualitativen Eigenschaften einer Implementierung [SSB10, Kan14]. Die Grundlage bilden Produktmetriken, die in traditionelle und objektorientierte Softwaremetriken unterteilt sind [ED96]. Die wichtigsten Metriken zur Messung der Programmgröße und Komplexität sind die *Zeilenmetrik* (LOC) und *Halstead-Metrik*, die wiederum aus *Vokabulargröße* (n), *Programmlänge* (N), *Volumen* (V),

Schwierigkeitsgrad (D) und *Implementierungsaufwand* (E) besteht. Die Messung der Programmstruktur erfolgt mit Hilfe der *durchschnittlichen zyklomatischen Komplexität* (ACC), die auch als *McCabe-Metrik* bezeichnet wird. Bei den objektorientierten Metriken finden die spezifischen Merkmale der Objektorientierung Berücksichtigung, wie die *Klassenanzahl* (NOC), die *durchschnittliche Methodenanzahl pro Klasse* (AMC), die *durchschnittliche Zeilenanzahl einer Methode* (ALCM) sowie die *Instabilität* (I). Mit Hilfe dieser Metriken lassen sich Aussagen über die Strukturiertheit, Verständlichkeit und Wartbarkeit einer Implementierung ableiten. Die Messergebnisse für das Managementsystem sind in der Tabelle 7.1 aufgeführt. Eine grafische Einordnung der Kennzahlen ist in der Abbildung 7.8 dargestellt.

7.5.1 Umfang

Der Gesamtumfang der Software beläuft sich auf 71 374 Zeilen Code und 1 720 Klassen. Grundsätzlich ist die Zeilenmetrik von der Programmiersprache beeinflusst, so dass die Halstead-Metrik in Form des Programmvolumens von insgesamt 2 629 756,38 einen besseren Vergleichswert liefert. Gemäß McConnell [McC04] ist ein einzelner Entwickler in der Lage, pro Tag etwa 10–50 Zeilen Code zu erzeugen. Für die vorliegende Implementierung ergibt dies eine Zeitspanne von etwa 4–20 Jahren. Die fünfjährige Entwicklungszeit liegt damit am unteren Ende der Skala.

7.5.2 Komplexität

Die zyklomatische Komplexität und die Halstead-Metrik bewerten die Qualität und Wartbarkeit der Software. Die zyklomatische Komplexität ist definiert als die Anzahl der linear unabhängigen Pfade innerhalb des Kontrollflussgraphen. Je größer der Wert ist, um so mehr Ausführungspfade existieren für eine Methode und umso schwieriger ist diese zu verstehen und zu testen. Mit Hilfe der zyklomatischen Komplexität lässt sich die Komplexität von Software unabhängig von der Programmiersprache beurteilen und eine Aussage über ihre Wartbarkeit ableiten. Um das Ergebnis einordnen zu können, muss dieses mit einem Referenzwert verglichen werden. Das *Software Engineering Institute* (SEI) der *Carnegie Mellon University* definiert die folgenden Wertebereiche [BBF+97]:

[1; 10) Die Methode gilt als simpel sowie leicht zu überschauen und zu testen.

[10; 20) Die Methode beinhaltet deutlich komplexeren Code, der zwar weiterhin verständlich, die Testbarkeit aber deutlich erschwert sein kann.

[20; 50) Die Methode kann eine unüberschaubare Anzahl von Code-Pfaden enthalten und lässt sich nur noch schwer verstehen und testen.

[50; ∞) Die Methode gilt als unwartbar.

Die durchschnittliche zyklomatische Komplexität beträgt 1,51. Es handelt sich um einen Durchschnittswert, so dass vor einer Bewertung die Ausreißer näher betrachtet werden müssen. Ausgehend von einem Gesamtumfang von 9 378 Methoden entfallen auf die 1. Kategorie 99,51 %, auf die 2. Kategorie 0,34 %, auf die 3. Kategorie 0,15 % und auf auf 4. Kategorie gar keine Methoden. Ein weiteres Indiz für die gute Verständlichkeit des Codes liefert die Halstead-Metrik. Für das Volumen gelten folgende Richtwerte: Methoden sollten ein Volumen von 20–1 000 aufweisen und eine Klasse eines von 100–8 000 [CL07]. Im Bezug zum Gesamtumfang ergibt dies einen Durchschnittswert von 280,42 pro Methode und 1 528,93 pro

Artefakt	LOC[1]	NOC[2]	AMC[3]	ALCM[4]	ACC[5]	I[6]	Halstead-Metrik				
							n[7]	N[8]	V[9]	D[10]	E[11]
Managementbaum	**66 864**	**1 584**	**5,53**	**5,39**	**1,52**	**0,461**	**9 066**	**189 050**	**2 485 298,64**	**365,17**	**907 558 970,65**
Core	12 094	226	7,3	5,28	1,419	0,447	1 912	32 532	354 626,99	238,05	84 421 663,17
RESTful-Service	342	10	6,59	3,84	1,014	0,562	151	785	5 682,14	24,2	137 507,97
RESTful-Proxy	999	18	4,33	7,71	1,766	0,554	548	2 972	27 039,35	48,12	1 301 262,34
AMQP-Service	845	18	5,88	5,11	1,284	0,528	364	2 322	19 755,09	39,67	783 846
AMQP-Proxy	1 075	21	7,14	4,78	1,35	0,567	467	2 826	25 058,92	46,39	1 162 689,49
XMPP-Service	653	17	4,35	4,93	1,301	0,556	320	1 639	13 639,64	32,07	437 545,29
XMPP-Proxy	1 216	26	6,38	4,7	1,345	0,607	503	3 020	27 102,73	46,02	1 247 399,03
DMT-Admin-Service	4 000	68	7,22	5,85	2,004	0,597	1 261	13 402	138 045,32	166,74	23 018 010,43
Policy-Manager	**4 510**	**136**	**4,45**	**4,49**	**1,21**	**0,431**	**885**	**10 337**	**101 194,4**	**113,8**	**11 516 747,45**
Regel-Engine	2 403	60	4,84	5,22	1,298	0,613	701	5 686	53 751,29	74,23	3 990 140,58
Ausdrucks-Engine	1 084	23	6,04	4,92	1,182	0,7	391	2 352	20 253,13	45,27	917 025,8
Managementsystem	**71 374**	**1 720**	**5,45**	**5,33**	**1,51**	**0,46**	**9 340**	**199 387**	**2 629 756,38**	**374,62**	**985 177 154,74**

[1] LOC = Lines of Code [2] NOC = Number of Classes [3] AMC = Average Number of Methods per Class [4] ALCM = Average Lines of Code per Method
[5] ACC = Average Cyclomatic Complexity [6] I = Instability [7] n = Vokabulargröße [8] N = Programmlänge [9] V = Volumen [10] D = Schwierigkeitsgrad
[11] E = Implementierungsaufwand

Tabelle 7.1: Kennzahlen des Managementsystems

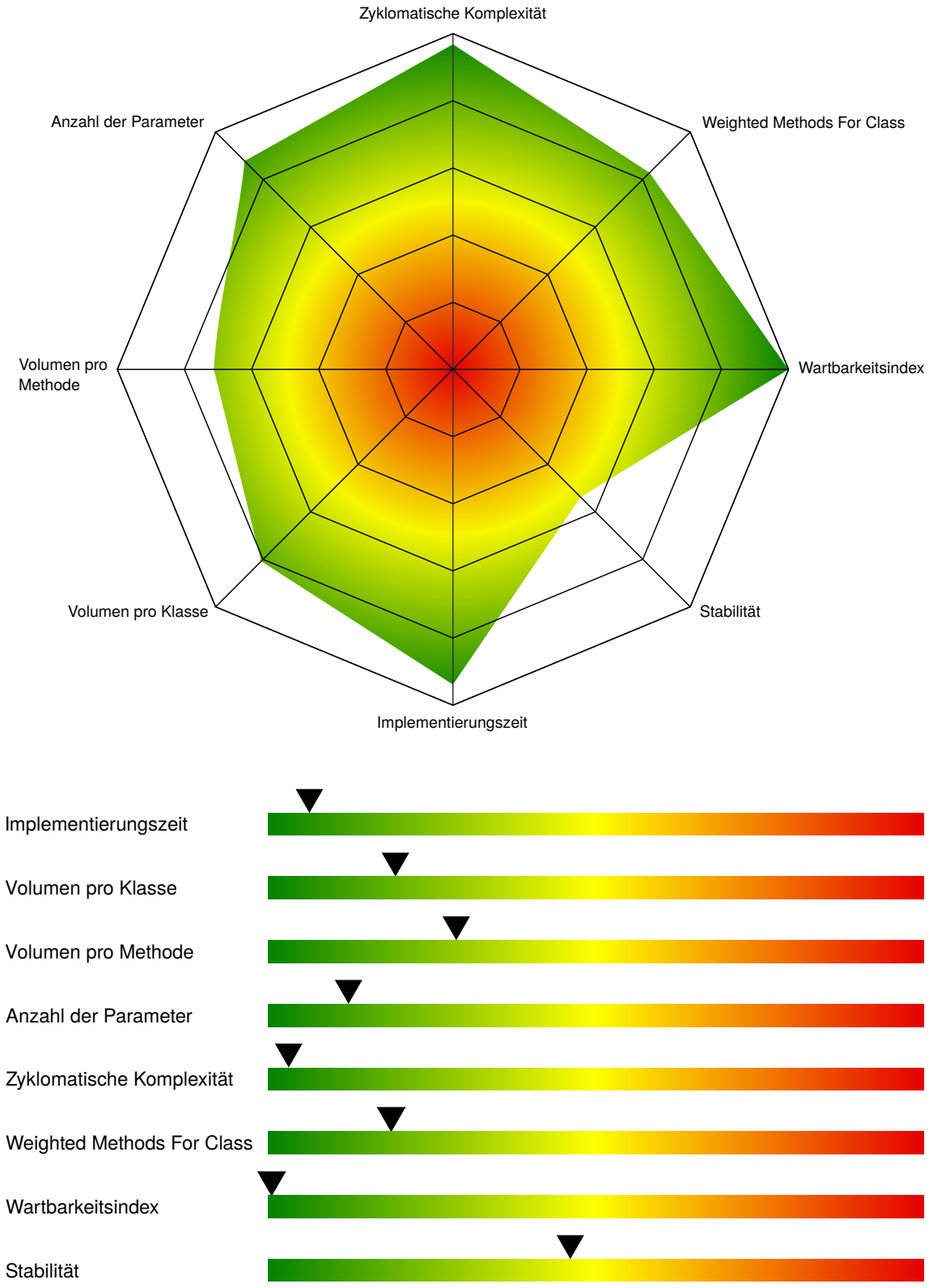


Abbildung 7.8: Kennzahlen des Managementsystems

Klasse. Beide Werte befinden sich jeweils am unteren Ende der Skala. Ähnlich verhält es sich mit der Länge einer Methode. Diese sollte bei einem Wert von 4–40 Programmzeilen liegen. Im Durchschnitt erstreckt sich eine Methode über 5,33 Zeilen, wobei eine Klasse über 5,45 Methoden verfügt. Die Maßzahlen der zyklomatischen Komplexität und der Halstead-Metrik lassen insgesamt den Schluss zu, dass die Implementierung verständlich und leicht zu testen ist. Dies ist insbesondere in Anbetracht des Gesamtumfangs eine wichtige qualitative Eigenschaft. Unterstützt wird diese Aussage durch den *Wartbarkeitsindex* (MI) einer Methode, der aus der Zeilenmetrik, der zyklomatischen Komplexität und dem Volumen berechnet wird. Ein Wert von mindestens 85 zeigt eine gute Wartbarkeit an. Bei einem Wert von 65–85 ist die Wartbarkeit mäßig, unterhalb von 65 ist die Methode kaum mehr wartbar [CLO95]. Der Wartbarkeitsindex beträgt 108,47 und zeigt somit eine sehr gute Wartbarkeit an.

7.5.3 Stabilität

Der Stabilitätswert von 0,46 deutet auf eine gewisse Neigung zur Instabilität hin, d. h. es besteht die Gefahr, dass sich Änderungen an Softwareartefakten negativ auf die Implementierung auswirken. Als Richtwert gilt ein Wert von 0,0–0,3 als stabil, ein Wert von 0,7–1,0 hingegen als instabil [BWL99]. Der leicht erhöhte Wert wird dadurch verursacht, dass gemäß OSGi-Best-Practices [HK07, WHKL08] stets eine Trennung zwischen API und Implementierung erfolgen sollte. Das Softwareartefakt, das die Implementierung bereitstellt, ist daher stets abhängig von demjenigen, das die API enthält. Auch wirkt sich die Auslagerung gemeinsam genutzter Pakete in eigenständige Softwareartefakte negativ auf das Qualitätsmaß aus.

8 Validierung

Das Ziel der Validierung ist eine Überprüfung der Konzepte und Lösungen hinsichtlich Anwendbarkeit und Eignung für das einheitliche Management serviceorientierter Systeme in einer Multi-Provider-Umgebung. Den Ausgangspunkt bildet ein Anwendungsfall aus dem Bereich der Alzheimer- und Demenzforschung, der dem international prämierten Forschungsprojekt EASI-CLOUDS (ITEA Award of Excellence in der Kategorie „Business Impact“, 2015) entstammt, in dessen Umfeld zugleich ein Großteil der eigenen Forschungs- und Entwicklungsarbeiten geleistet wurde [BFL$^+$13, FLT$^+$15, TOR$^+$15]. Das technische Management stützt sich auf das in Kapitel 6 beschriebene Informationsmodell, im Speziellen auf die innerhalb der Abschnitte 6.10 und 6.11 entwickelten Struktur- und Steuerungsmuster. Anhand einer Experimentumgebung und eines Nutzungsszenarios wird nachgewiesen, dass sich mit Hilfe dieser Muster ein gütegesicherter Servicebetrieb in einem Verbund von heterogenen Ausführungsumgebungen gewährleisten lässt.

8.1 Anwendungsfall

Bildgebende Systeme wie die *Magnetresonanztomografie* (MRT) liefern für die medizinische Diagnostik und Forschung zweidimensionale Schichtaufnahmen der menschlichen Anatomie mit immer höherer Auflösung. Diese Daten müssen mit Hilfe rechen- und ressourcenintensiver Verfahren weiterverarbeitet werden. Die zu Analysezwecken erforderlichen Rechenkapazitäten können aufgrund eines zeitlich stark schwankenden Bedarfs von Kliniken und medizinischen Forschungseinrichtungen vielfach nicht selbst vorgehalten werden. Dedizierte Speicher- und Rechnerkapazitäten ließen sich aufgrund einer zu geringen Auslastung nicht wirtschaftlich betreiben. In Phasen intensiver Forschung werden über einen zeitlich beschränkten Zeitraum extrem große Rechenkapazitäten benötigt. In diesem Zusammenhang bietet sich die Bereitstellung der Analysefunktionen nach dem SaaS-Modell an, die im Sinne einer ausgelagerten Dienstleistung von einem externen Anbieter erbracht werden. Dadurch lassen sich die Funktionen nach Bedarf anfordern und nutzungsbasiert abrechnen. Dabei sind neben den funktionalen Anforderungen auch eine Reihe nicht-funktionaler Anforderungen zu erfüllen [CNYM00]. Das Ziel ist die Bereitstellung einer zuverlässigen und leistungsgerechten Dienstleistung auf Basis von

Software-Services. Die bedarfsorientierte Servicebereitstellung sowie der Einsatz cloudbasierter, virtueller und physikalischer Umgebungen würden die Forschung mit bildgebenden Verfahren deutlich erleichtern. Zudem könnten sich auch kleinere Kliniken und Forschungseinrichtungen ohne eigene Rechenzentren daran beteiligen. Allerdings kommen SaaS-Services bisher nur selten zum Einsatz. Das Problem liegt weniger im Medizinbereich, sondern vielmehr darin, dass es noch zu wenig Erfahrungswerte gibt bezüglich der automatisierten Bereitstellung und dem gütegesicherten Betrieb von Services in einem Verbund aus heterogenen Ausführungsumgebungen [DWC10, ZCB10]. Dies gilt insbesondere für Legacy-Anwendungen [SKI13, PXW13]. Dabei handelt es sich um bestehende und teilweise monolithische Anwendungen, die nicht dem SaaS-Modell folgen und eine manuelle Installation und Einrichtung erfordern. Der zu Validierungszwecken verwendete Anwendungsfall entstammt der neuroradiologischen Forschung und dient der computerunterstützten Diagnostik neurodegenerativer Erkrankungen, insbesondere von Morbus Alzheimer sowie anderer Demenzerkrankungen.

8.1.1 Neurodegenerative Erkrankungen

Nach Angaben der Deutschen Alzheimer Gesellschaft gibt es zurzeit in Deutschland etwa 1,7 Millionen Demenzerkrankte und jährlich kommen circa 300 Tausend Menschen neu hinzu [Bic18]. Gemeinsamen Schätzungen von Weltgesundheitsorganisation und Alzheimer's Disease International zufolge waren im Jahr 2018 weltweit etwa 50 Millionen Menschen von der Krankheit betroffen und die Tendenz ist steigend [Alz18]. Es lassen sich unterschiedliche Formen der Demenz unterscheiden. Dabei ist Morbus Alzheimer die häufigste Form. Sie macht vermutlich über 60 % der Krankheitsfälle aus. Die Erkrankung tritt fast ausnahmslos erst jenseits des 60. Lebensjahres auf und zählt ebenso wie die meisten anderen Demenzformen zu den gerontopsychiatrischen Störungen. Dabei handelt es sich um psychische Erkrankungen, die erst im fortgeschrittenen Alter auftreten. Seltenere Formen können auch bei jüngeren Patienten vorkommen. Wie die Abbildung 8.1 zeigt, wird bis zum Jahr 2050 voraussichtlich jeder Dritte in Deutschland über 65 Jahre alt sein. Das Medianalter wird voraussichtlich bei 51,6 liegen [PR15]. Im Jahr 2010 betrug dieses noch 44,1. Die Zahl der altersbedingten Krankheiten insbesondere

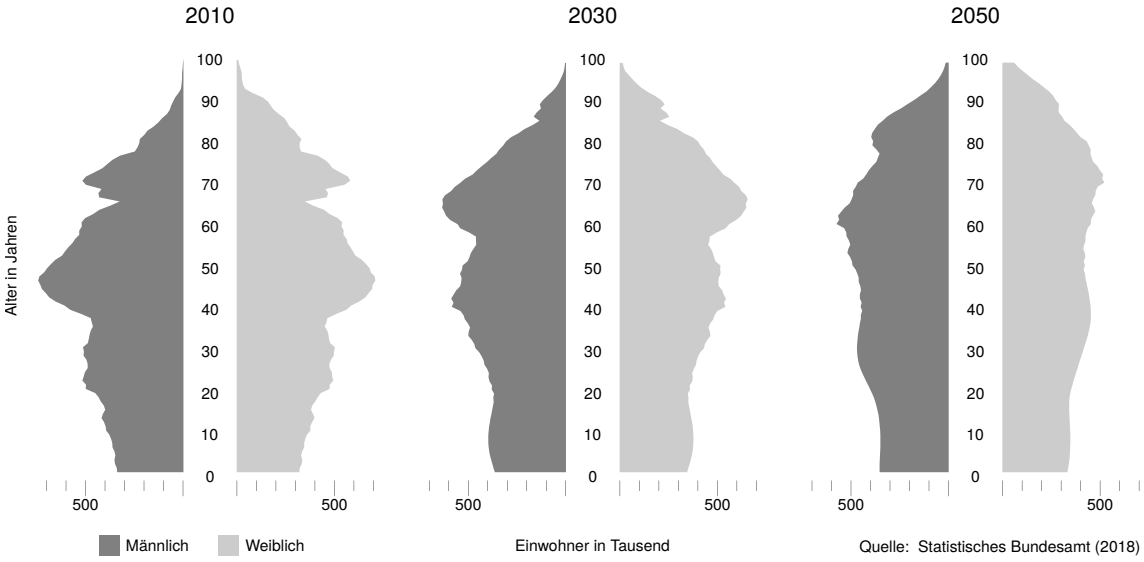


Abbildung 8.1: Prognose der Altersstruktur der deutschen Bevölkerung bis 2050

der Altersdemenz, für die es bisher keine wirksame Therapie und Heilung gibt, wird stark ansteigen. Bei einem so großen Wachstum der älteren Bevölkerung schätzt man die Zahl der Betroffenen auf 152 Millionen weltweit [Alz18]. Derzeitige Behandlungsansätze können lediglich das Fortschreiten der Symptome verlangsamen, der im Gehirn stattfindende krankheitsbedingte Abbauprozess der Großhirnrinde lässt sich nicht verzögern oder gar aufhalten. Die genauen Ursachen der Krankheit ließen sich bisher noch nicht endgültig klären [AKS03, Jel03].

Neuroradiologische Forschung

Angesichts der enorm wachsenden Fallzahlen hat innerhalb der neuroradiologischen Forschung eine intensive Suche nach neuen Therapien und Diagnosemöglichkeiten eingesetzt. Durch Verbesserungen und Fortschritte seitens der bildgebenden Systeme lässt sich inzwischen ein Morbus Alzheimer frühzeitig diagnostizieren. Die Grundlage dafür bilden MRT-Aufnahmen des Gehirns, die computerunterstützt aufbereitet und analysiert werden. Bisherige Forschungsergebnisse zeigen, dass sich bei neurodegenerativen Erkrankungen die Dicke der Großhirnrinde (*kortikale Dicke*) verringert [SBS$^+$04, DBS$^+$09, ACU$^+$10]. Dies lässt sich millimetergenau im MRT feststellen [Är07]. Voraussetzung dafür ist eine entsprechend hochauflösende Bildgebung. Die Abbildung 8.2 zeigt die Bestimmung der kortikalen Dicke aus einer MRT-Aufnahme [FD00]. Der Abstand zwischen gelber und roter Linie entspricht der kortikalen Dicke und ist durch die grünen Pfeile markiert. Der Wert muss für jeden Punkt auf der Hirnoberfläche berechnet werden. Dies erfordert den Einsatz leistungsfähiger Segmentierungsalgorithmen aus der digitalen Bildverarbeitung.

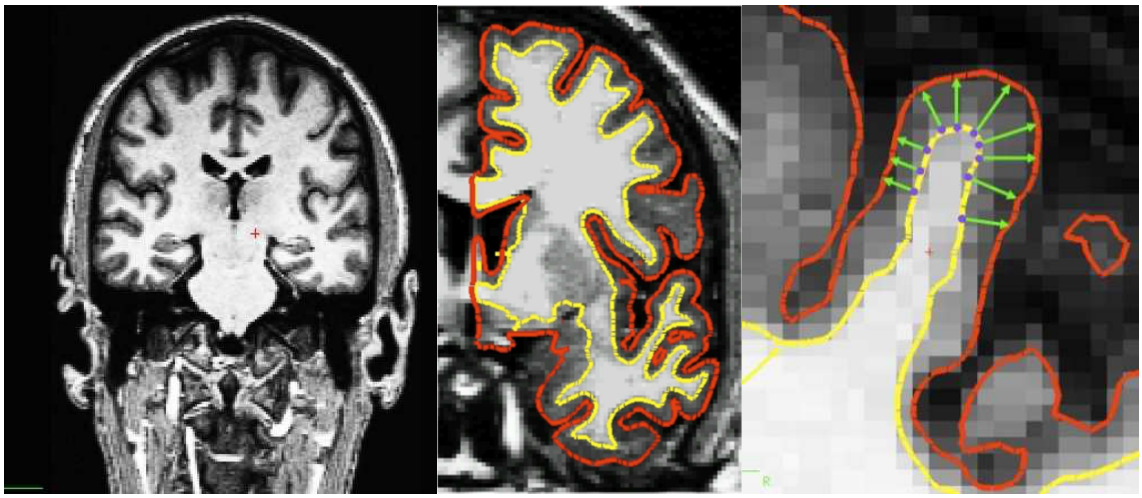


Abbildung 8.2: Bestimmung der kortikalen Dicke aus einer MRT-Aufnahme aus [TOR$^+$15]

8.1.2 Nutzungsszenario

Bisher wurde auf solche Analysemethoden zumeist nur im Forschungsumfeld zurückgegriffen, in Zukunft ist aber davon auszugehen, dass die Diagnoseverfahren auch in der klinischen Versorgung direkt am Patienten zum Einsatz kommen. Das Nutzungsszenario [TOR$^+$15] stellt sich wie folgt dar:

- Der Neurologe erstellt eine Reihe von MRT-Aufnahmen seiner Patienten.

- Die Aufnahmen werden unter Beachtung datenschutzrechtlicher Bestimmungen an einen Speicherdienst übertragen und stehen potentiellen Verarbeitungsdiensten zur Verfügung.
- Über einen Marktplatz wählt der Neurologe einen geeigneten Verarbeitungsdienst aus. Dabei handelt es sich im vorliegenden Fall um einen Service, der die kortikale Dicke aus den MRT-Aufnahmen berechnen kann.
- Der Neurologe legt im Anschluss die nicht-funktionalen Anforderungen fest, insbesondere den spätest möglichen Fertigstellungszeitpunkt sowie eine Obergrenze für die anfallenden Kosten.
- Der Service-Provider erstellt daraufhin ein Angebot, das der Neurologe ablehnen oder verbindlich akzeptieren kann.
- Er beauftragt den Service-Provider zur Durchführung der Analyse und kann jederzeit den Berechnungsfortschritt sowie die bis dato entstandenen Kosten einsehen.
- Ist die Berechnung abgeschlossen, wird der Neurologe darüber in Kenntnis gesetzt.
- Er kann die Ergebnisse online einsehen oder diese zur genaueren Analyse herunterladen und mit lokal installierten Softwarewerkzeugen bearbeiten.

8.1.3 Anforderungen

Die Abschätzung der kortikalen Dicke ist das Teilergebnis einer MRT-Bildanalyse. Diese kann als nicht-interaktiver Service aufgefasst werden, da sich auf einen einmal gestarteten Analysevorgang nicht mehr steuernd einwirken lässt. Bei der Umsetzung des Nutzungsszenarios gilt es, folgende Anforderungen zu erfüllen:

MRT-Bildanalyse als Service

Die MRT-Bildanalyse soll auf Basis der freien und quelloffenen Software *FreeSurfer* [Fis12] erfolgen, die aus MRT-Aufnahmen die kortikale Oberfläche des Gehirns rekonstruieren kann. Entwicklung und Bereitstellung verantwortet das *Athinoula A. Martinos Center for Biomedical Imaging*, ein Forschungszentrum der *Harvard Medical School* und des *Massachusetts Institut of Technology* (MIT). Die Anwendungsfunktion gilt es, gemäß dem SaaS-Modell innerhalb einer Multi-Provider-Umgebung bereitzustellen. Dabei sollen neben physikalischen auch virtuelle und cloudbasierte Umgebungen einsetzbar sein.

Überwachung und Steuerung

Das Informationsmodell muss über Status- und Konfigurationsvariablen verfügen, die eine Überwachung und Steuerung der Anwendungsfunktion ermöglichen. Die Ausführung der MRT-Bildanalyse erfolgt streng sequenziell entlang einer Pipeline, die eine Vielzahl langlaufender rechenintensiver Algorithmen umfasst. Dies setzt voraus, dass sich der Verarbeitungsfortschritt erfassen und durch eine Zustandsgröße beschreiben lässt.

Flexible Node-Bereitstellung

Nodes bilden die Betriebsumgebungen für Services und müssen sich im Falle virtueller und cloudbasierter Umgebungen dynamisch bereitstellen lassen. Die Grundlage dafür bildet eine

Multi-Provider-Umgebung, die unterschiedliche Ausführungsumgebungen und Flavors umfasst. Für den anforderungsgerechten Einsatz müssen Informationen über ihre Leistungsfähigkeit vorliegen. Darauf aufsetzend sind dann die vom Nutzer vorgegebenen leistungs- und kostenbezogenen Zielvorgaben zu erfüllen.

Automatisierte Provisionierung

Neben der Infrastrukturschicht gilt es auch, die Plattformschicht und die Anwendungsschicht kontrollieren zu können. Dementsprechend muss sich der Applikationsstapel automatisiert aufsetzen und anpassen lassen. Dies umfasst die Installation, Deinstallation und Konfiguration von Softwarekomponenten. Dadurch können Services und Applikationen dynamisch in das System eingebracht und auch aus diesem wieder entfernt werden. Die Provisionierung muss es ermöglichen, Nodes unabhängig von Technologie und Anbieter zur Serviceerbringung einzusetzen.

Gütegesicherter Servicebetrieb

Hauptaufgabe des technischen Managements ist die Gewährleistung eines gütegesicherten Servicebetriebs. Zu diesem Zweck müssen sich Leistungsverschlechterungen und Fehlfunktionen zeitnah feststellen lassen. Je nach Einstufung sind gegebenenfalls adaptive Maßnahmen zu ergreifen, um den sich abzeichnenden Zielverfehlungen entgegenzuwirken. Dies setzt im Sinne der Self-X-Eigenschaften grundlegende Fähigkeiten zur Selbstüberwachung und Selbstkonfiguration voraus, so dass das System eigenständig über die Durchführung von Anpassungsmaßnahmen entscheiden kann.

8.2 Anwendungsspezifische Strukturmuster

Um das Nutzungsszenario auf Grundlage eines Managementprozesses umsetzen zu können, ist eine Verfeinerung der in Abschnitt 6.10 beschriebenen Strukturmuster erforderlich. In diesem Zusammenhang gilt es aufzuzeigen, wie sich die bestehenden Managementobjektklassen dazu einsetzen lassen, um die für das Nutzungsszenario relevanten Ressourcen durch Managementobjekte zu erfassen. Die nachfolgenden Abschnitte erläutern die anwendungsspezifischen Strukturmuster zur Repräsentation der Experimentumgebung, der MRT-Bildanalyse und der Managementfunktionen.

8.2.1 Experimentumgebung

Aus den Vertragsbeziehungen zwischen Anbieter und Zulieferer ergeben sich Zusammensetzung und Aufbau der Multi-Provider-Umgebung. Der resultierende Umgebungsverbund kann mit Hilfe eines Strukturmusters beschrieben werden. Die Abbildung 8.3 zeigt die zu Validierungszwecken festgelegte Experimentumgebung. Sie wird durch ein gleichnamiges Managementobjekt repräsentiert und als Ausprägung einer Multi-Provider-Umgebung modelliert. Der Verbund besteht aus Ausführungsumgebungen, die sich hinsichtlich Anbieter, Betriebsmodell, Technologie und Managementschnittstelle unterscheiden. Um im Rahmen der Validierung möglichst repräsentative Ergebnisse erzielen zu können, gilt es, einen breiten Querschnitt aus cloudbasierten, virtuellen und physikalischen Umgebungen zu erstellen. Die Experimentumgebung besteht insgesamt aus 14 Ausführungsumgebungen, darunter elf cloudbasierten, zwei virtuellen und einer physikalischen Umgebung.

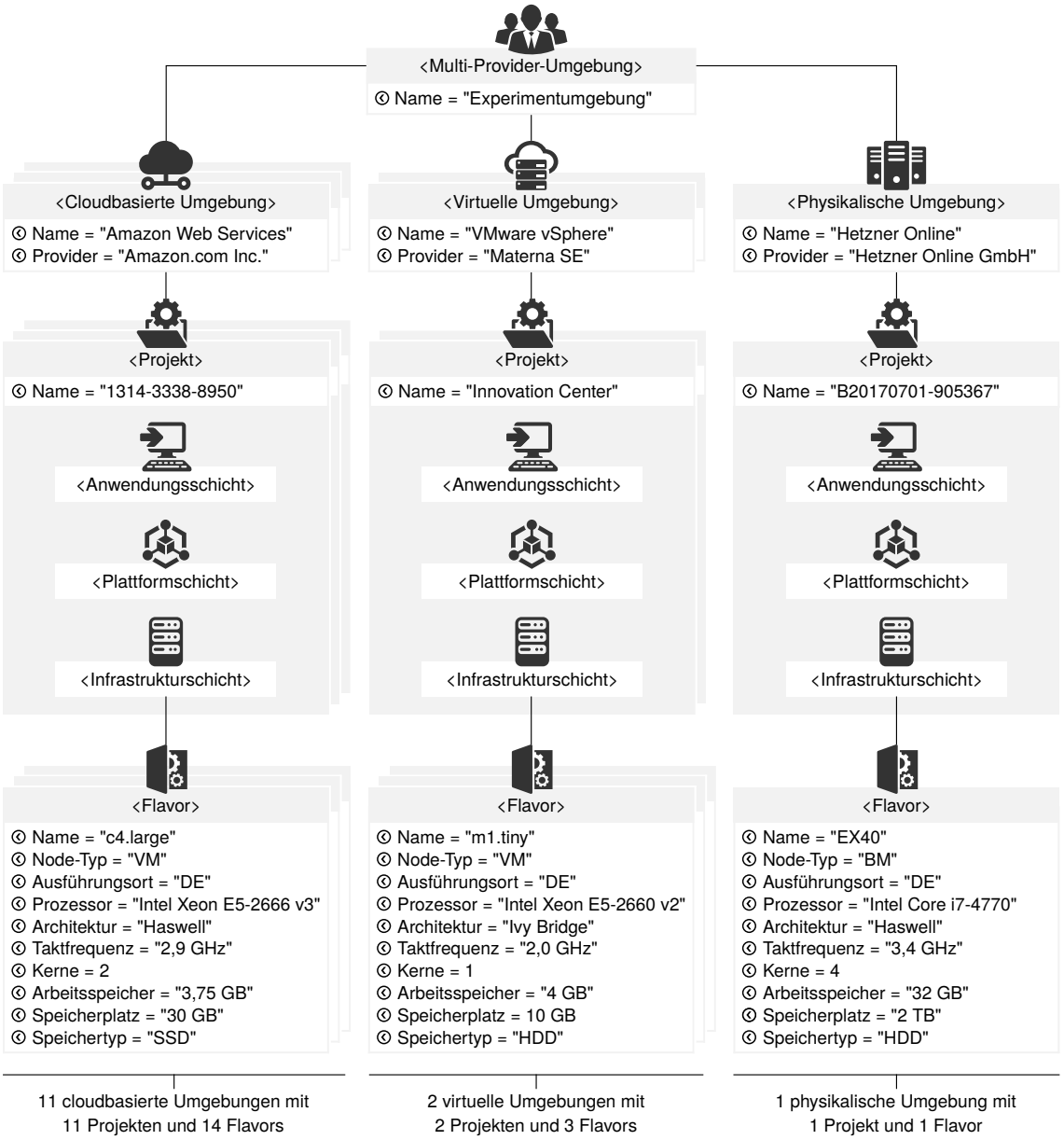


Abbildung 8.3: Managementobjekte der Experimentumgebung

8.2.1.1 Cloudbasierte Umgebungen

Die cloudbasierten Umgebungen lassen sich in öffentliche und private Clouds unterteilen, die entweder als On-Premises- oder Off-Premises-Lösung realisiert sind:

Öffentliche Clouds

Neben den vier führenden US-amerikanischen Unternehmen, sind in der Experimentumgebung fünf deutsche Anbieter öffentlicher Cloud-Umgebungen vertreten. Diese umfasst folgende Ausführungsumgebungen:

- Amazon Web Services

- Microsoft Azure
- IBM Cloud
- Google Cloud
- Open Telekom Cloud
- ProfitBricks
- SysEleven
- ScaleUp
- teutoStack

Amazon Web Services, Microsoft Azure, IBM Cloud, Google Cloud und ProfitBricks verfügen jeweils über eine anbieterspezifische Managementschnittstelle. Die Open Telekom Cloud, ProfitBricks, SysEleven, ScaleUp und teutoStack basieren hingegen auf der freien Open-Source-Software-Plattform *OpenStack* [RB14, SSJ+16]. Der Zugriff auf die Überwachungs- und Steuerungsfunktionen ist über die von OpenStack bereitgestellte Managementschnittstelle möglich. Die einheitliche Verwendung der Ausführungsumgebungen wird jedoch dadurch erschwert, dass sich die Managementschnittstellen hinsichtlich der verwendeten Version unterscheiden. Die genannten Anbieter garantieren den Cloud-Betrieb nach deutschen Datenschutzbestimmungen. Diese Eigenschaft nimmt insbesondere beim Umgang mit medizinischen Bilddaten einen hohen Stellenwert ein. Sie kann die Wahl einer geeigneten Ausführungsumgebung maßgeblich beeinflussen, speziell dann, wenn die MRT-Bilddateien als personenbezogene Daten aufzufassen sind. Je nach Situation kann der Einsatz von Anonymisierungsmaßnahmen nicht ausreichend sein. Entscheidend für die Erkennbarkeit ist stets, ob sich Rückschlüsse auf die Person aufgrund besonderer charakteristischer Merkmale ziehen lassen. Ähnlich wie ein Fingerabdruck unterscheidet sich auch die Hirnanatomie von Mensch zu Mensch [FSS+15, LLXH18, BKC+18].

Private Clouds

Um den Datenschutz- und Datensicherheitsbedenken Rechnung zu tragen, umfasst die Experimentumgebung neben den öffentlichen Cloud-Umgebungen auch zwei private Cloud-Umgebungen. Es handelt sich um:

- Blue Box Cloud
- OpenStack Cloud

Blue Box ist ein kommerzieller Anbieter privater OpenStack-basierter Cloud-Umgebungen, die auf einer IT-Infrastruktur innerhalb eines externen Rechenzentrums betrieben werden. Ergänzend dazu wurde eine OpenStack-basierte Cloud-Umgebung in einem internen Rechenzentrum aufgebaut, die eigenverantwortlich betrieben wird. Sie umfasst zwei Controller-, drei Storage- und vier Compute-Nodes. Es handelt sich um ein Hochverfügbarkeits-Deployment auf Grundlage von Ubuntu 16.04 (Xenial Xerus), Ceph Luminous (v12.2.7) und OpenStack Pike (2017.2).

8.2.1.2 Virtuelle Umgebungen

Die cloudbasierten Umgebungen werden um zwei virtuelle Umgebungen ergänzt, die ebenfalls eigenverantwortlich betrieben werden. Es handelt sich um:

- VMware vSphere
- Proxmox VE

VMware vSphere ist eine Virtualisierungsplattform für die Bereitstellung und den Betrieb virtueller Maschinen. Die Ausführungsumgebung basiert auf vSphere 6.5 und verwendet den Bare-Metal-Hypervisor ESXi. Die Proxmox VE ist eine auf Debian-basierende Open-Source-Virtualisierungsplattform. Die Ausführungsumgebung nutzt den *Quick Emulator* (QEMU) in Verbindung mit der *Kernel-based Virtual Machine* (KVM) und basiert auf der Proxmox VE 5.2. Sie ermöglicht den Einsatz von virtuellen Maschinen und Containern. Diese Fähigkeit stellt eine Besonderheit der Virtualisierungsplattform dar und bedeutet, dass zwei grundlegend verschiedene Virtualisierungstechnologien gleichzeitig auf denselben Wirtssystemen zur Nutzung bereitstehen.

8.2.1.3 Physikalische Umgebungen

Des Weiteren umfasst die Multi-Provider-Umgebung eine physikalische Umgebung. Diese bietet durch den Einsatz von Bare-Metal-Servern als physikalische Single-Tenant-Systeme ein Höchstmaß an Rechenleistung ohne zusätzlichen Verwaltungsaufwand und Hintergrundlast, die durch eine Virtualisierungsschicht und die gemeinsame Ressourcennutzung entstehen. Es handelt sich um den folgenden Anbieter:

- Hetzner Online

Die physikalische Umgebung stellt zugleich eine Referenzumgebung dar, an deren Leistungsfähigkeit sich alle cloudbasierten und virtuellen Umgebungen messen lassen müssen. Innerhalb der Experimentumgebung verfügt sie hinsichtlich der Rechenleistung über das leistungsstärkste Flavor.

8.2.1.4 Projekte

Gemäß der Festlegung in Abschnitt 6.2 umfasst eine Ausführungsumgebung eine beliebige Anzahl von Projekten, die ihrerseits über eine eigene Infrastruktur-, Plattform- und Anwendungsschicht verfügen können. Innerhalb der Experimentumgebung ist jeder Ausführungsumgebung genau ein Projekt zugeordnet, wobei aufgrund der Serviceerbringung in Eigenverantwortung jede der drei Schichten vollständig ausgeprägt ist. Gemäß der vertraglichen Vereinbarungen stellen die Zulieferer dem Anbieter hinreichend leistungsfähige Flavors zur Verfügung. Diese lassen sich den Infrastrukturschichten der dazugehörigen Projekte zuordnen. Flavors besitzen eine projektweit eindeutige Bezeichnung und geben Auskunft über den Ausführungsort in Form eines Länderkürzels nach ISO 3166-2 [ISO13]. Die Entscheidung für ein Flavor bestimmt zugleich auch den Rechtsraum und damit die Datenschutzbestimmungen, denen die Datenverarbeitung auf dem Node untersteht. Des Weiteren legt das Flavor die Betriebstechnologie, den Prozessor, die Architektur, die Taktfrequenz, die Anzahl der Kerne, die Größe des Arbeitsspeichers, die Größe des Speicherplatzes und den Speichertyp fest. Die Tabelle 8.1 bietet einen Überblick über die in der Experimentumgebung verfügbaren Flavors und ordnet sie ihren Anbietern zu.

Provider	Flavor	Node-Typ	Prozessor	Architektur	Taktfrequenz	Kerne	Arbeitsspeicher	Speicherplatz
Amazon Web Services	c4.large	VM	Intel Xeon E5-2666 v3	Haswell	2,9 GHz	2	3,75 GB	30 GB
Microsoft Azure	Standard_D1_v2	VM	Intel Xeon E5-2673 v3	Haswell	2,4 GHz	1	3,5 GB	10 GB
	Standard_D2_v3	VM	Intel Xeon E5-2673 v4	Broadwell	2,3 GHz	2	8 GB	10 GB
IBM Cloud	BL2.1x4x100	VM	Intel Xeon E5-2683 v3	Haswell	2,0 GHz	1	4 GB	100 GB
Google Cloud	n1-standard-1	VM	Intel Xeon E5 v4	Broadwell	2,2 GHz	1	3,75 GB	10 GB
Open Telekom Cloud	s1.medium	VM	Intel Xeon E5-2658A v3	Haswell	2,2 GHz	1	4 GB	10 GB
	h1.large	VM	Intel Xeon E5-2690 v3	Haswell	2,6 GHz	2	4 GB	10 GB
ProfitBricks	AMD	VM	AMD Opteron 62xx	Bulldozer	2,8 GHz	1	4 GB	10 GB
	Intel	VM	Intel Xeon E5-2680 v4	Haswell	2,4 GHz	1	4 GB	10 GB
SysEleven	m1.tiny	VM	Intel Core Processor (no TSX)	Haswell	2,4 GHz	1	4 GB	10 GB
ScaleUp	m1.medium	VM	Intel Xeon E5-2640 v3	Haswell	2,0 GHz	1	4 GB	10 GB
teutoStack	m1.small	VM	Westmere E56xx/L56xx/X56xx	Nehalem	2,0 GHz	1	4 GB	10 GB
Bluebox Cloud	c1.small	VM	Intel Xeon E312xx	Sandy Bridge	2,0 GHz	1	4 GB	10 GB
OpenStack Cloud	m1.small	VM	Intel Xeon E5410	Harpertown	2,33 GHz	1	4 GB	10 GB
VMware vSphere	m1.small	VM	Intel Xeon E5-2660 v2	Ivy Bridge	2,0 GHz	1	4 GB	10 GB
Proxmox VE	Container	CT	Intel Xeon E3-1225 v2	Ivy Bridge	3,2 GHz	1	4 GB	10 GB
	VM	VM	Intel Xeon E3-1225 v2	Ivy Bridge	3,2 GHz	1	4 GB	10 GB
Hetzner Online	EX40	BM	Intel Core i7-4770	Haswell	3,4 GHz	4	32 GB	2 TB

VM = Virtuelle Maschine CT = Container BM = Bare-Metal-Server

Tabelle 8.1: Flavors der Experimentumgebung

8.2.2 MRT-Bildanalyse

Das Strukturmuster aus Abschnitt 6.10.2 bildet den Ausgangspunkt zur Modellierung der Anwendungsfunktion. Die Abbildung 8.4 zeigt die Managementobjekte zur Repräsentation der funktionalen Abhängigkeiten der MRT-Bildanalyse.

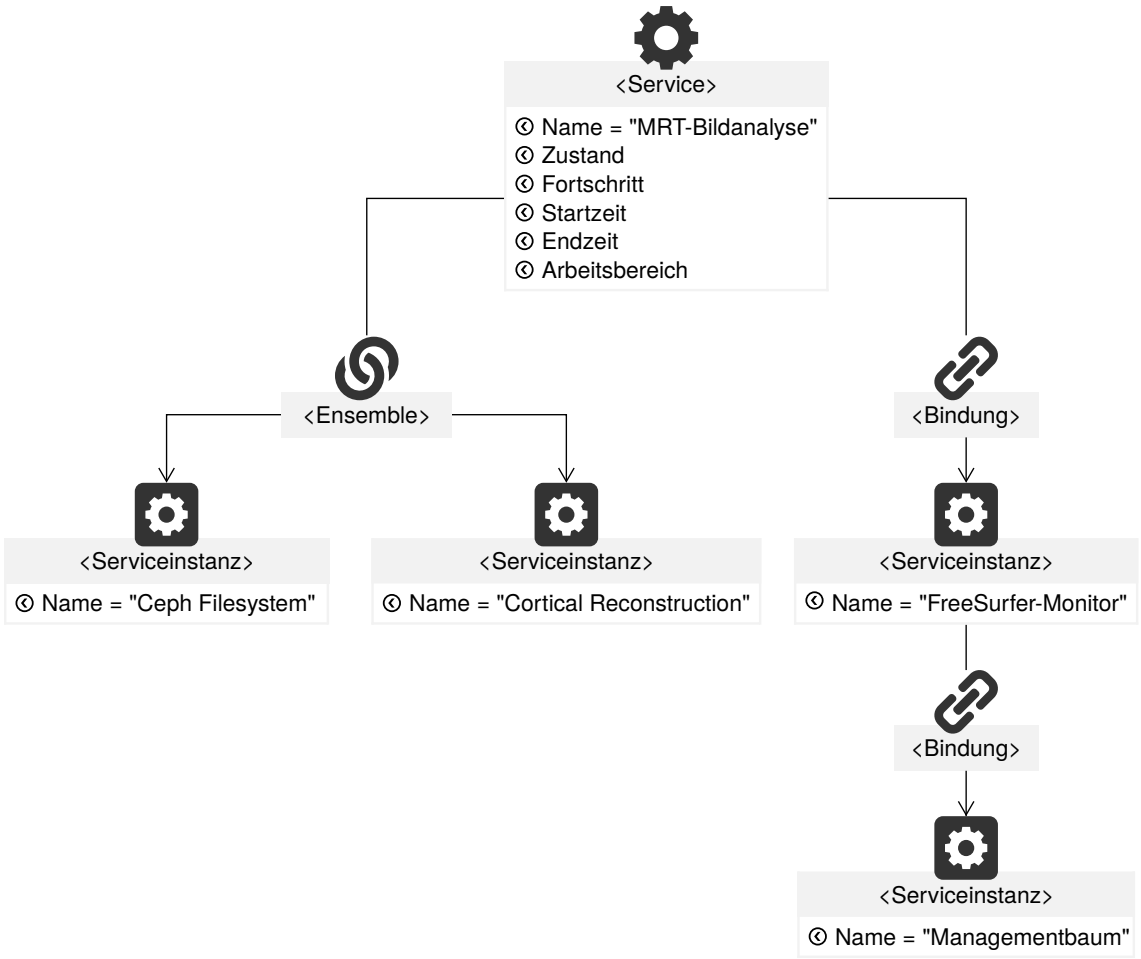


Abbildung 8.4: Funktionale Abhängigkeiten der MRT-Bildanalyse

Der Service wird nicht unmittelbar von einer Softwarekomponente bereitgestellt. Er entspricht vielmehr einer übergeordneten Dienstleistung, die mit Hilfe dreier Services erbracht wird. Diese sind im Objektmodell durch Serviceinstanzen repräsentiert. Zentraler Bestandteil ist ein Service zur oberflächen- und volumenbasierten Rekonstruktion des Gehirns aus zweidimensionalen Schichtaufnahmen. Der Zugriff auf die MRT-Eingabedaten erfolgt über einen Ceph-basierten Dateispeicher, der zugleich auch als Ablageort für Zwischen- und Endergebnisse dient. Beide Services sind Bestandteil eines gemeinsamen Ensembles, das zur Durchführung einer MRT-Bildanalyse aufzubauen ist. Des Weiteren ist ein Überwachungsservice erforderlich, der über den Managementbaum die Zustandsgrößen zugänglich macht. Die Nutzungsbeziehung zwischen der MRT-Bildanalyse und dem Überwachungsservice wird durch eine vom Ensemble unabhängige Bindung modelliert. Der Grund dafür ist, dass der Analysevorgang auch dann noch lauffähig ist, sollte die Überwachungsfunktion ausgefallen sein. Weist aber einer der am Ensemble beteiligten Services eine Fehlfunktion auf, wird dieses im Ganzen ungültig. In einem solchen Fall kann die

Anwendungsfunktion nicht mehr erbracht werden, da der Datenzugriff beeinträchtigt oder der Analyseprozess abgestürzt ist.

8.2.2.1 Ceph-basierter Dateispeicher

Die Abbildung 8.5 zeigt die Managementobjekte, die sich aus dem Einsatz eines Ceph-basierten Shares ergeben. Im Rahmen des Nutzungsszenarios findet ein zentraler Ceph-Speichercluster Anwendung, der ein nutzerspezifisches Share bereitstellt.

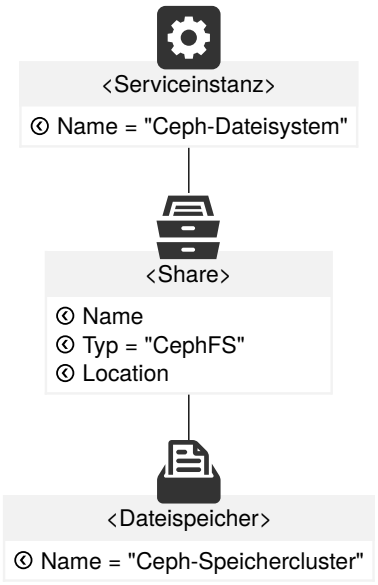


Abbildung 8.5: Managementobjekte des Ceph-basierten Dateispeichers

Dieses nimmt die MRT-Bilddateien auf, die der Anwender zur Verfügung stellt. Die serviceorientierte Sicht erfolgt mit Hilfe einer Serviceinstanz, die den Zugriff auf den Dateispeicher repräsentiert. Das Share ist öffentlich zugänglich und lässt sich daher in jeden Node einhängen, der über eine Internetanbindung sowie über die notwendigen Treiber und Zugriffsberechtigungen verfügt. Die Entkoppelung der Datenhaltung vom Lebenszyklus des Nodes gewährleistet, dass die Zwischen- und Endergebnisse erhalten bleiben und weiterhin zugänglich sind, auch wenn der Ausführungsknoten gestoppt oder beendet wird.

8.2.2.2 Kortikale Rekonstruktion

Die Anwendungsfunktion zur kortikalen Rekonstruktion wird mit Hilfe von FreeSurfer erbracht. Die Abbildung 8.6 zeigt die Managementobjekte, die sich aus der Bereitstellung der Software-Suite und der dazugehörigen Überwachungskomponente ergeben. Die Infrastrukturschicht umfasst einen dedizierten Node eines zuvor festgelegten Flavors, in dessen lokales Dateisystem das nutzerspezifische Share eingehängt wird. Der Node ist über ein Netzwerk-Interface an das Service-Netzwerk angebunden. Die Zuweisung der Rolle eines MRT-Bildanalysierers veranlasst das Konfigurationsmanagement zur Bereitstellung und Konfiguration des Applikationsstapels. Dabei wird unter anderem das FreeSurfer-Softwareartefakt auf den Node kopiert, installiert und gemäß Rollendefinition konfiguriert. Die Plattformschicht umfasst das Betriebssystem sowie ein OSGi-Framework, das die Softwarekomponente ausführt, die den FreeSurfer-basierten Analysevorgang überwacht. Die Bereitstellung des OSGi-Frameworks erfolgt mit Hilfe von *Apache Karaf*,

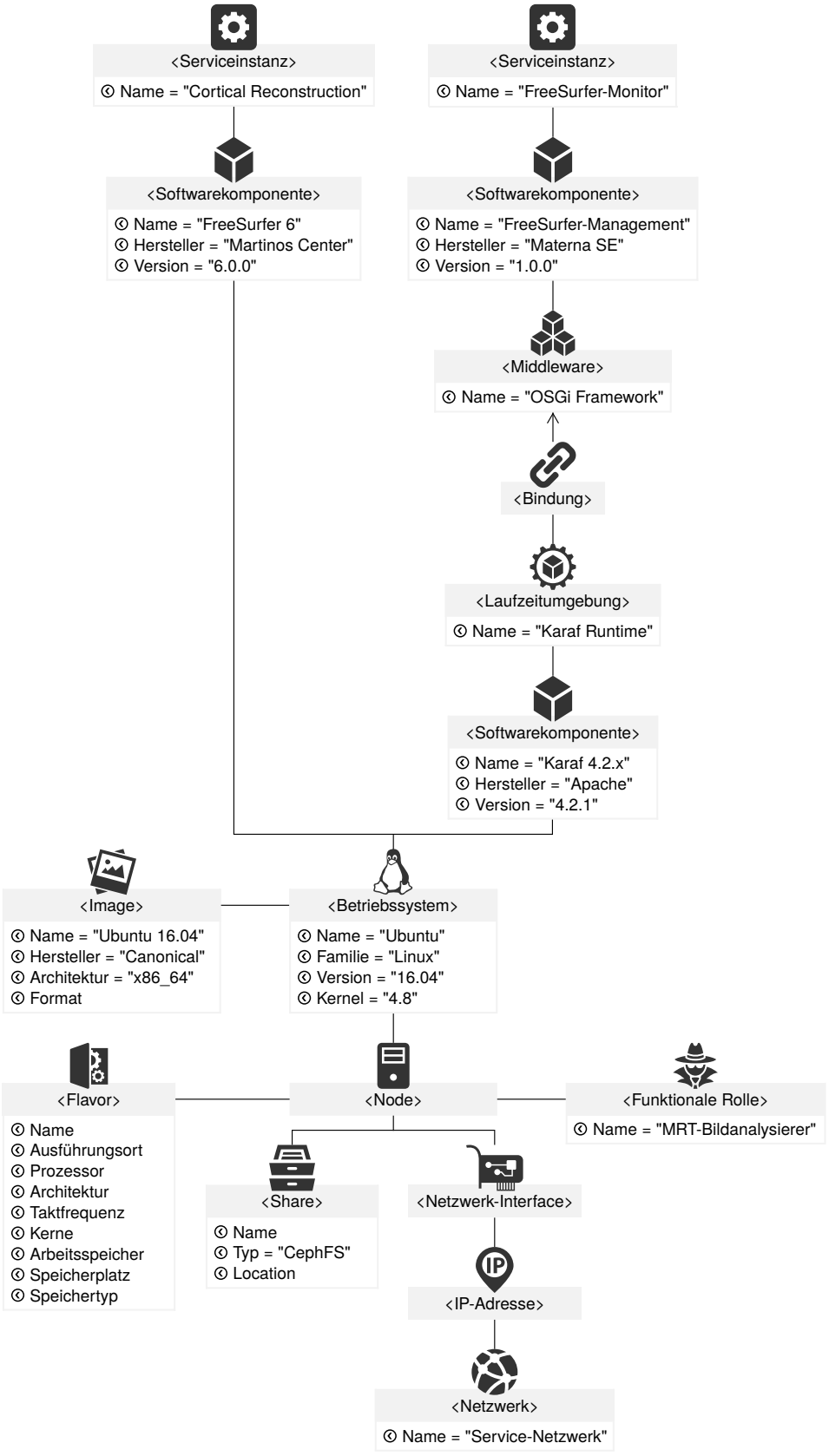


Abbildung 8.6: Managementobjekte der kortikalen Rekonstruktion

einem Applikationsserver, über den sich mehrere voneinander getrennte OSGi-Frameworks erzeugen lassen. Die Abhängigkeitsbeziehung wird mit Hilfe einer Bindung repräsentiert. Das Betriebssystem bildet ein Ubuntu 16.04 mit einem HWE-Kernel 4.8 (engl. Hardware Enablement Stack). Innerhalb der Anwendungsschicht wird jeder Analysevorgang durch zwei Services repräsentiert. Der eine Service bildet den FreeSurfer-basierten Analysevorgang ab, der andere den Überwachungsservice. Innerhalb des OSGi-Frameworks wird dazu das Softwareartefakt für das FreeSurfer-Management installiert. Eine vorgangsspezifische Konfiguration veranlasst die Servicebereitstellung.

8.2.3 Infrastrukturmanagement

Das in Abschnitt 6.10.3 vorgestellte Strukturmuster zur Repräsentation des Infrastrukturmanagements wird an dieser Stelle um die technischen Details der Serviceerbringung erweitert. Die Abbildung 8.7 zeigt den relevanten Teilausschnitt des Strukturmusters. Demnach stützt sich der Provider-Service auf die von *Vagrant* bereitgestellten Funktionen zur Provisionierung und Steuerung virtueller Maschinen und Container. Während des Aufsetzungsprozesses wird dazu das Vagrant-Softwareartefakt auf den Node kopiert, installiert und konfiguriert. Das Ergebnis ist eine Softwarekomponente, deren Funktion durch einen gleichnamigen Service

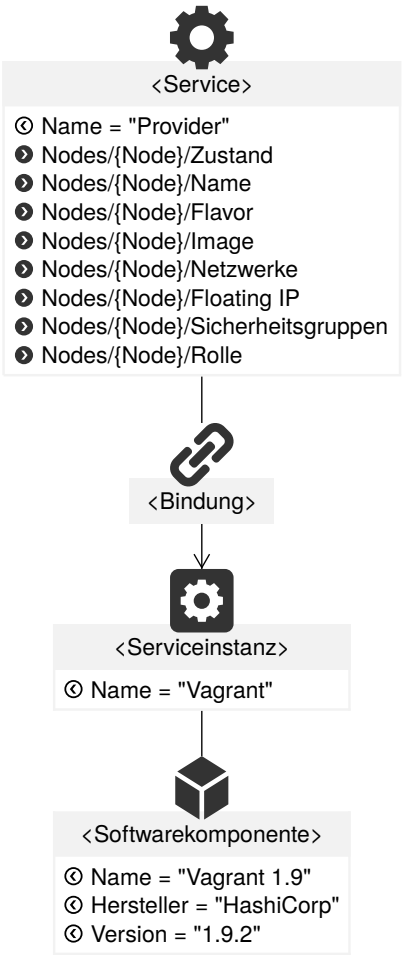


Abbildung 8.7: Managementobjekte des Providers

repräsentiert wird. Zugleich ist die Provider-Rolle auf die technischen Gegebenheiten der Ausführungsumgebung abgestimmt. Dementsprechend wird ein spezielles Provider-Plugin installiert, das die umgebungsspezifische Managementschnittstelle anbindet und damit Vagrant zur Kontrolle der Infrastrukturschicht befähigt. Die Bereitstellungsbeziehung zwischen Provider- und Vagrant-Service ist durch eine Bindung repräsentiert.

8.2.4 Konfigurationsmanagement

Neben dem Infrastrukturmanagement gilt es auch, das in Abschnitt 6.10.4 vorgestellte Strukturmuster zur Repräsentation des Konfigurationsmanagements um die technischen Details der Serviceerbringung zu ergänzen. Die Abbildung 8.8 zeigt den Teilausschnitt aus dem Strukturmuster. Demnach greift der Provisioner auf die vom Konfigurationswerkzeug *Puppet* bereitgestellten Funktionen zurück. Dazu wird während des Aufsetzungsprozesses das Softwareartefakt des Puppet-Servers heruntergeladen und im Zielsystem installiert. Das Ergebnis ist eine Softwarekomponente, die den Puppetmaster als lokalen Systemservice anbietet. Dadurch verfügt der Provisioner über die Fähigkeit, Nodes automatisiert zu provisionieren. Dies beinhaltet die Installation, Deinstallation und Konfiguration von Softwarekomponenten. Die Konfigurationsvariablen des Provisioner-Services sind unterschiedlichen Hierarchieebenen zugeordnet, so dass sich Konfigurationswerte untergeordneter Ebenen in einer übergeordneten Ebene überschreiben lassen. Es handelt sich um Konfigurationsfragmente, die während des Aufsetzungsprozesses

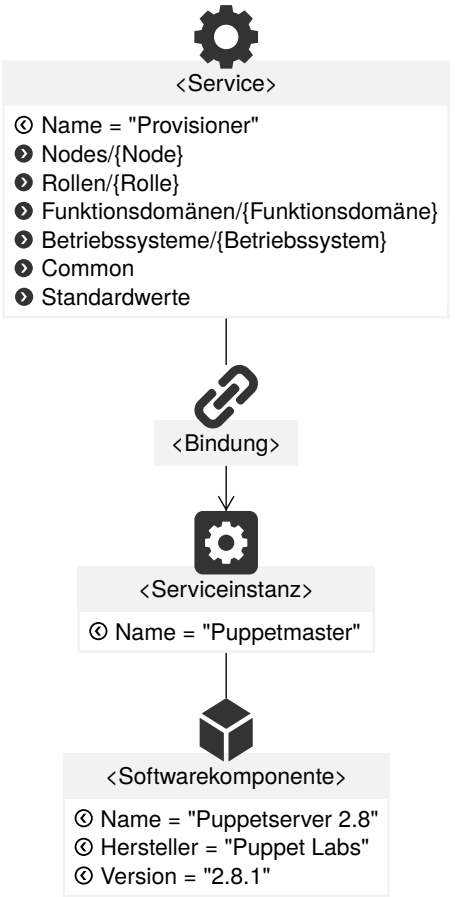


Abbildung 8.8: Managementobjekte des Provisioners

zusammengefügt werden. Dabei wird für jede der sechs Ebenen diejenige Konfiguration ermittelt, die für einen anfragenden Node Gültigkeit hat. Diese werden anschließend gemäß ihrer hierarchischen Anordnung miteinander zu einer Gesamtkonfiguration verbunden und über das Konfigurationsmanagement zur Anwendung gebracht. Die Konfigurationsvariablen werden mit strukturierten Daten im YAML-Format belegt und repräsentieren Konfigurationsdateien für die auf dem Puppetmaster verwendete hierarchische Datenbank *Hiera*.

Anbindung an das Konfigurationsmanagement

Um die Nodes innerhalb eines Projekts über das Konfigurationsmanagement steuern zu können, bedarf es einer Anbindung an den zentralen Verwaltungsserver. Die Abbildung 8.9 zeigt die daraus resultierenden Managementobjekte. Grundsätzlich verfügt jeder Node über einen Agenten, der an den Puppetmaster gebunden ist, der vom Provisioner bereitgestellt wird. Das Konfigurationsmanagement arbeitet nach dem Pull-Prinzip. Dabei fordert der Agent vom Verwaltungsserver

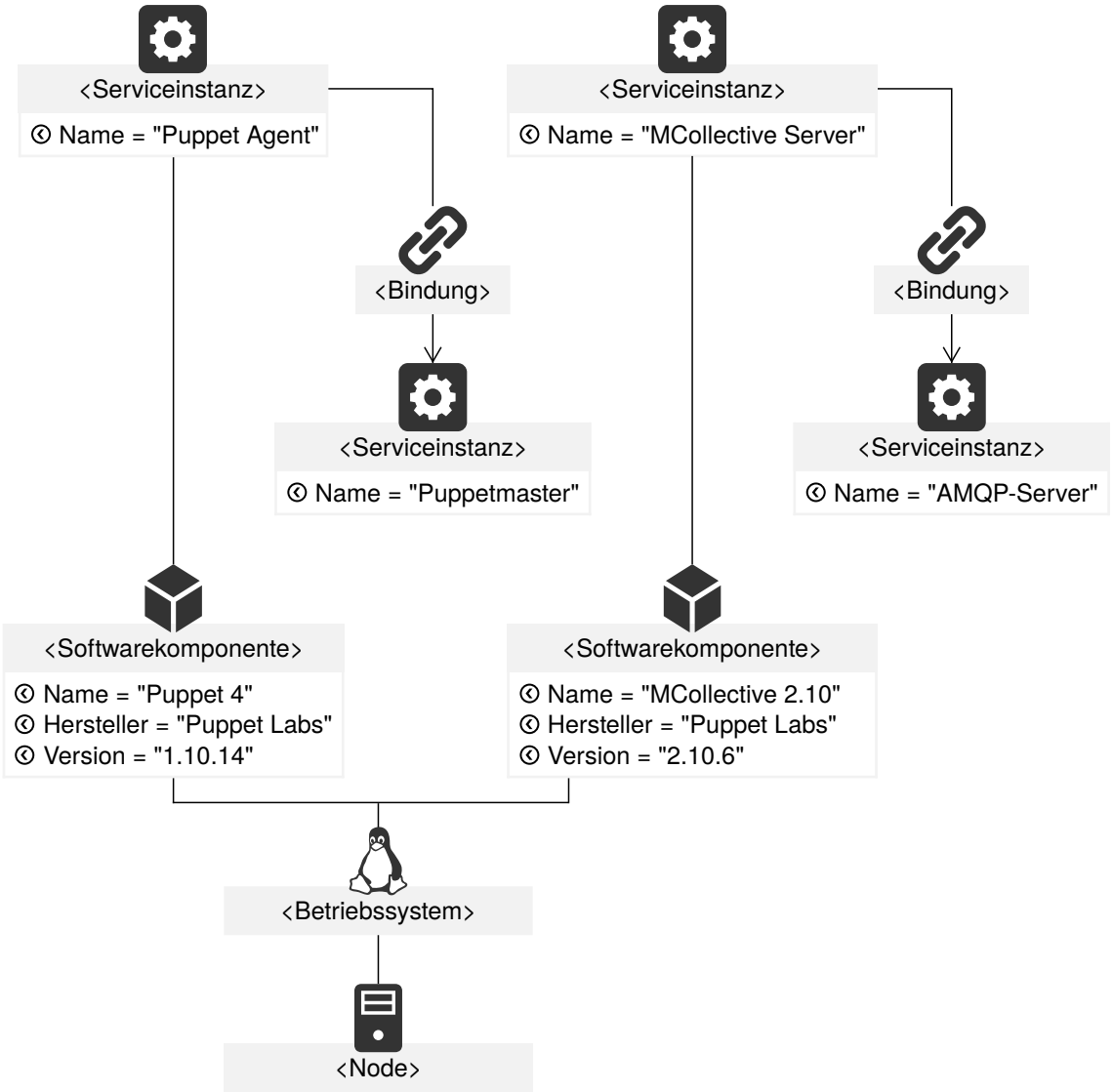


Abbildung 8.9: Managementobjekte zur Anbindung an das Konfigurationsmanagement

in festen Zeitintervallen seine Konfiguration an und bringt diese zur Anwendung. Um einen Node zeitnah rekonfigurieren zu können, verfügt dieser über einen MCollective-Server, der an die vom Messenger bereitgestellte AMQP-basierte Middleware angebunden ist. MCollective ist ein Werkzeug zur Serverorchestrierung, das es ermöglicht, auf einem bestimmten Node oder einer ganzen Gruppe von Nodes einen Konfigurationslauf anzustoßen.

8.2.5 Autonomes Management

Als Organisationsform kommt das zentralisierte Management zum Einsatz. Dazu wird innerhalb des Management-Projekts ein autonomer Manager bereitgestellt, der die Steuerung der Multi-Provider-Umgebung übernimmt. Gemäß dem in Abschnitt 6.5.2 entwickelten funktionalen Rollenmodell erbringt der autonome Manager die Aufgaben des Objekt- und Servicemanagers. Dem Management-Projekt wird ein dedizierter Node hinzugefügt, der mit dem Management-

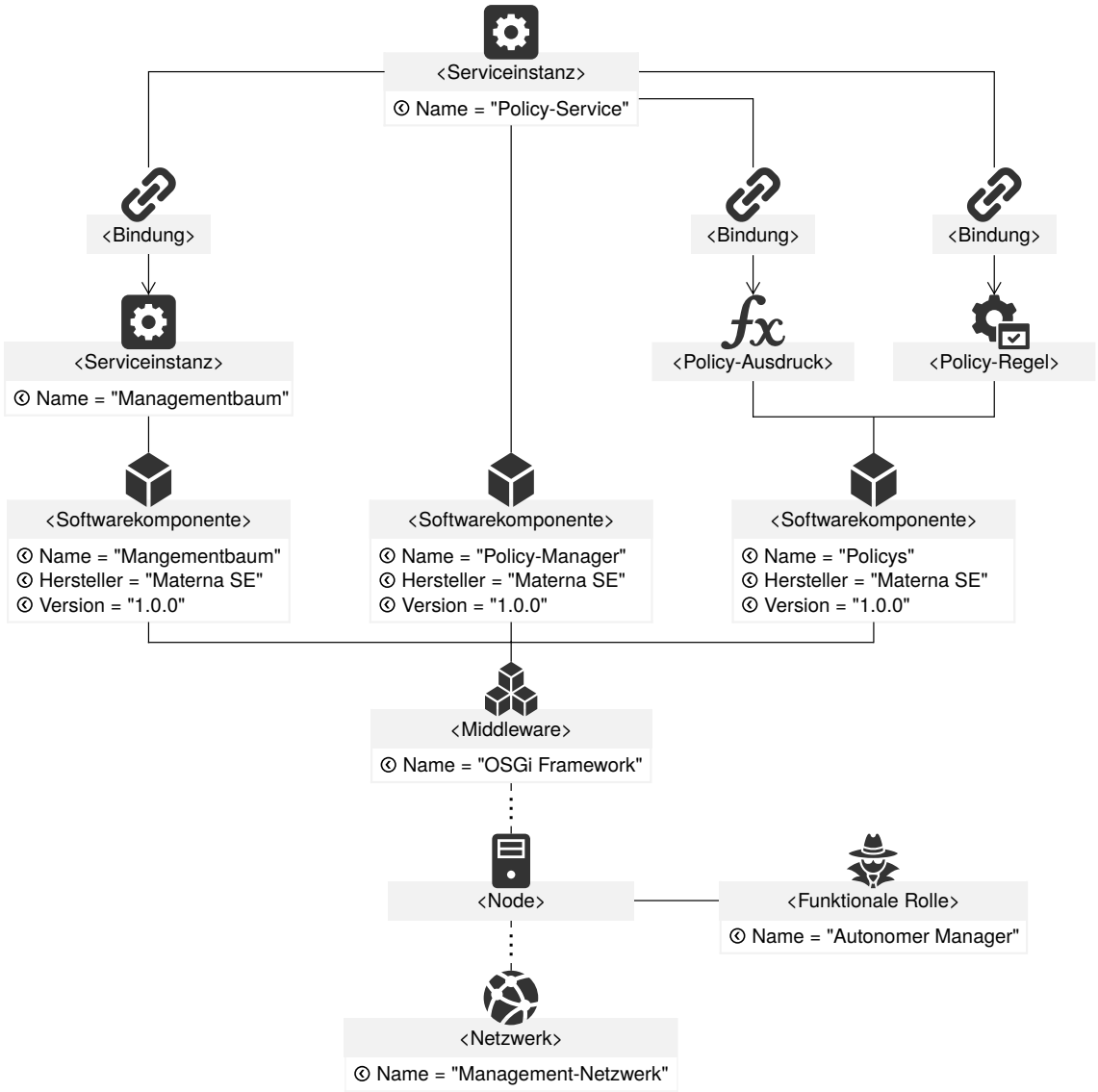


Abbildung 8.10: Managementobjekte des autonomen Managers

Netzwerk verbunden ist und der zugleich die Rolle des autonomen Managers einnimmt. Die Abbildung 8.10 zeigt die sich daraus ergebenden Managementobjekte. Der Managementbaum und der Policy-Manager werden gemeinsam in einem OSGi-Framework bereitgestellt. Dazu werden aus dem Repository die gleichnamigen Softwareartefakte heruntergeladen und auf dem Node installiert. In Verbindung mit einer Konfiguration erfolgt die Servicebereitstellung. Der Managementbaum ermöglicht den Zugriff auf die Managementobjekte der Experimentumgebung. Der Policy-Service verwaltet die zu Steuerungszwecken eingesetzten Policy-Regeln und Policy-Ausdrücke. Sie entstammen einer gemeinsamen Softwarekomponente, die diese als Services innerhalb des OSGi-Frameworks verfügbar macht. Policys operieren ausschließlich auf Managementobjekten. Dementsprechend ist die Nutzungsbeziehung zwischen Policy-Service und Managementbaum durch eine Bindung repräsentiert.

8.3 Parametrisierung der Steuerungsmuster

Die von den Managementobjekten bereitgestellten Status- und Konfigurationsvariablen bilden die Mess- und Stellgrößen des Systems. Sie dienen dem technischen Management zum einen als Grundlage für das Treffen von Steuerungsentscheidungen und zum anderen als Mittel zur Systemanpassung. Im Rahmen des Nutzungsszenarios soll der Managementprozess ausschließlich von Policy-Regeln und Policy-Ausdrücken erbracht werden. Dazu kommen die Steuerungsmuster aus Abschnitt 6.11 zum Einsatz. Diese müssen vorab parametrisiert und auf die Anforderungen des Nutzungsszenarios abgestimmt werden.

8.3.1 Initiale Provisionierung

Aufgabe der Bereitstellungsregel ist die initiale Provisionierung eines Nodes, der mit einem Arbeitsauftrag zur Durchführung einer MRT-Bildanalyse beauftragt wird. Bezugnehmend auf die in Abschnitt 6.11.1.2 eingeführten Platzhalter in Verbindung mit dem in Abschnitt 8.2.2.2 erläuterten Strukturmuster, verwendet die Policy-Regel folgende Werte:

```
${Servicename} = "MRT-Bildanalyse"
${Image} = "Ubuntu 16.04"
${Rolle} = "MRT-Bildanalysierer"
```

Zur Veranschaulichung der in der Regelaktion ermittelten Konfiguration sei die folgende Belegung der Statusvariablen angenommen:

```
Account.Identifikator = 1001
Account.Name = "cfiehe"
Share.Typ = "CephFS"
Share.Location = "/volumes/_nogroup/54bed6e8-4b35-43cf-bc98-888b7106694d"
Service.Arbeitsbereich = "/neuroimaging/subjects/OAS1_0001_MR1"
```

Das nachfolgende Listing zeigt die daraus erstellte Konfiguration, die die Bereitstellung der kortikalen Rekonstruktion auf Basis von FreeSurfer, die Erzeugung eines Überwachungsservices und das Einhängen des nutzerspezifischen Shares in das Host-System veranlasst:

```
# Accounts
accounts::accounts:
  cfiehe:
    ensure: present
    uid: 1001
```

```
   home: /home/cfiehe

# CephFS
cephfs::mounts:
  '/home/cfiehe/media/54bed6e8-4b35-43cf-bc98-888b7106694d':
    ensure: present
    owner: cfiehe
    group: cfiehe
    device: 'conf=/etc/ceph/ceph.conf,id=cephfs,client_mountpoint=/volumes/
        ↪ _nogroup/54bed6e8-4b35-43cf-bc98-888b7106694d'
    options:
      - '_netdev'
      - 'defaults'

# Karaf Runtime
karaf::configs:
  'f1f9ae8c-db05-4784-876b-e17be66c95d0@com.materna.nodes.osgi.service.
      ↪ application.impl.component.ExternalApplicationHandleComponent:root
      ↪ ':
    ensure: present
    properties:
      application.target: '(application.name=FreeSurfer)'
      application.arguments: 'recon-all -no-isrunning -sd #{subjects_dir} -
          ↪ subject #{subject} -make all'
      user: cfiehe
      subjects_dir: '/home/cfiehe/media/54bed6e8-4b35-43cf-bc98-888b7106694d
          ↪ /neuroimaging/subjects'
      subject: OAS1_0001_MR1
  'e4d4f603-97b5-4063-bd74-0911d1a32d00@com.materna.nodes.osgi.service.
      ↪ freesurfer.monitor.component.FreeSurferMonitorComponent:root':
    ensure: present
    properties:
      version: '6.0.0'
      pipeline: autorecon-all
      logfile: '/home/cfiehe/media/54bed6e8-4b35-43cf-bc98-888b7106694d/
          ↪ neuroimaging/subjects/OAS1_0001_MR1/scripts/recon-all.log'
```

8.3.2 Aufrufparametrisierung des Flavorauswahlausdrucks

Die Auswertung des Flavorauswahlausdrucks wird sowohl von der Bereitstellungsregel als auch von der Migrationsregel veranlasst. Neben der Servicereferenz dienen die beiden folgenden Parameter als Eingabe:

```
p = 0.9
f = "(Ausführungsort=DE)"
```

Dadurch wird der Flavorauswahlausdruck dazu veranlasst, ausschließlich diejenigen Flavors in die engere Auswahl zu nehmen, bei denen in mindestens 90 % der Fälle eine Einhaltung des Zeitlimits möglich ist und deren Ausführungsort sich innerhalb Deutschlands befindet.

8.3.3 Bewertung der Rechtzeitigkeit der Ausführung

Der Zielabweichungsausdruck ermittelt gemäß der in Abschnitt 6.11.2.1 vorgestellten Berechnungsvorschrift für eine laufende MRT-Bildanalyse die prozentuale Zielabweichung vom Zeitlimit. Aufgabe der Rechtzeitigkeits- und der Fortschrittsregel ist es, die potentielle Zielverfehlung zu bewerten. Im Rahmen des Nutzungsszenarios verwenden die beiden Policy-Regeln die folgende Abbildungsvorschrift:

$(-\infty; 1)$ Die Ausführung erfolgt plangemäß und erfordert keine Intervention (= Erfüllt).

$[1; 5)$ Abweichungen unter 5 % gelten als Frühwarnung vor Problemen (= Bedenklich).

$[5; 10)$ Dieser Bereich macht das Einleiten von Maßnahmen erforderlich (= Gefährdet).

$[10; \infty)$ Alles über 10 % erfordert eine sofortige Intervention (= Kritisch).

Das dazugehörige Codefragment lautet wie folgt:

```
Float zielabweichung = zielabweichungsausdruck.evaluate(s=service);
List<Servicegüteparameter> servicegüteparameter = service.servicegü
    ↪ teparameter;
Servicegüteparameter rechtzeitigkeit = servicegüteparameter.select(p | p.
    ↪ getValue("Name") == "Rechtzeitigkeit");
if zielabweichung < 1 then
  rechtzeitigkeit.setValue("Erfüllt");
else if zielabweichung < 5 then
  rechtzeitigkeit.setValue("Bedenklich");
else if zielabweichung < 10 then
  rechtzeitigkeit.setValue("Gefährdet");
else
  rechtzeitigkeit.setValue("Kritisch");
end if
```

8.4 Experimentelle Parameterbestimmung

Im Rahmen einer experimentellen Parameterbestimmung wird die Leistungsfähigkeit der Flavors hinsichtlich der Serviceausführung ermittelt. Zudem wird aufgezeigt, wie sich der Verarbeitungsfortschritt eines Arbeitsauftrags näherungsweise bestimmen lässt, auch wenn der Service selbst keine derartigen Informationen bereithält. Abschließend erfolgt die Kostenzuordnung.

8.4.1 Abschätzung der Bearbeitungszeit

Das erste Experiment dient der Abschätzung der Bearbeitungszeit einer MRT-Bildanalyse. Das Ergebnis sind Funktionen, die das Laufzeitverhalten auf den unterschiedlichen Flavors beschreiben und die der Flavorauswahlausdruck zur Quantil-Berechnung verwendet. Die Funktionsbestimmung soll mit Hilfe eines Verfahrens erfolgen, das ohne domänenspezifisches Wissen auskommt und das sich daher auch auf andere Nutzungsszenarien übertragen lässt. Voraussetzung ist, dass sich eine Arbeitseinheit in Form eines Auftrags definieren lässt, der dem Service als Eingabe dient und dessen Bearbeitung nach einer gewissen Zeitspanne abgeschlossen ist. Dabei muss die Bearbeitungszeit unmittelbar von der Leistungsfähigkeit des Flavors abhängen, d. h. der Einsatz eines leistungsstärkeren Flavors führt zu einer Verkürzung der Bearbeitungszeit.

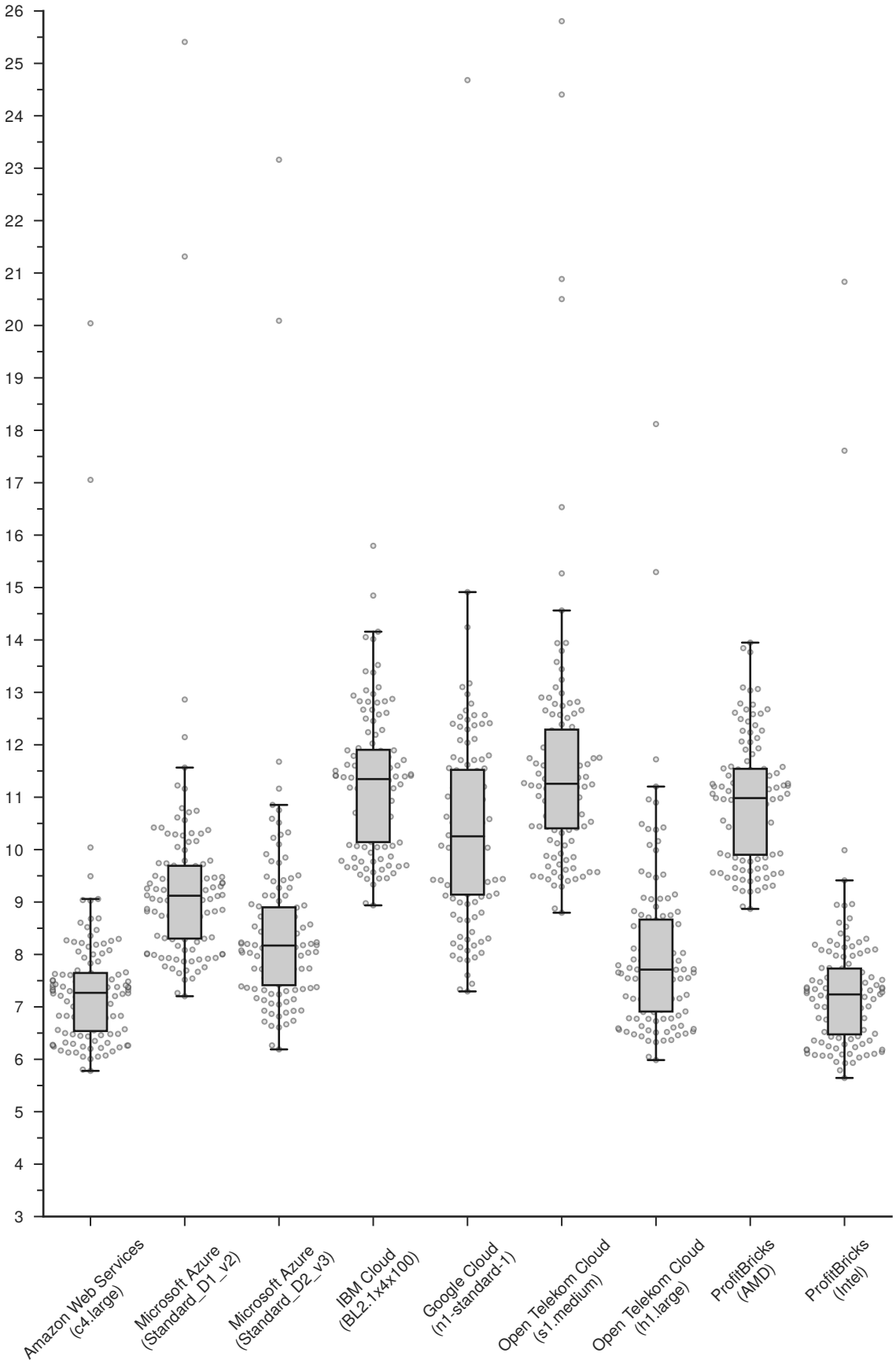
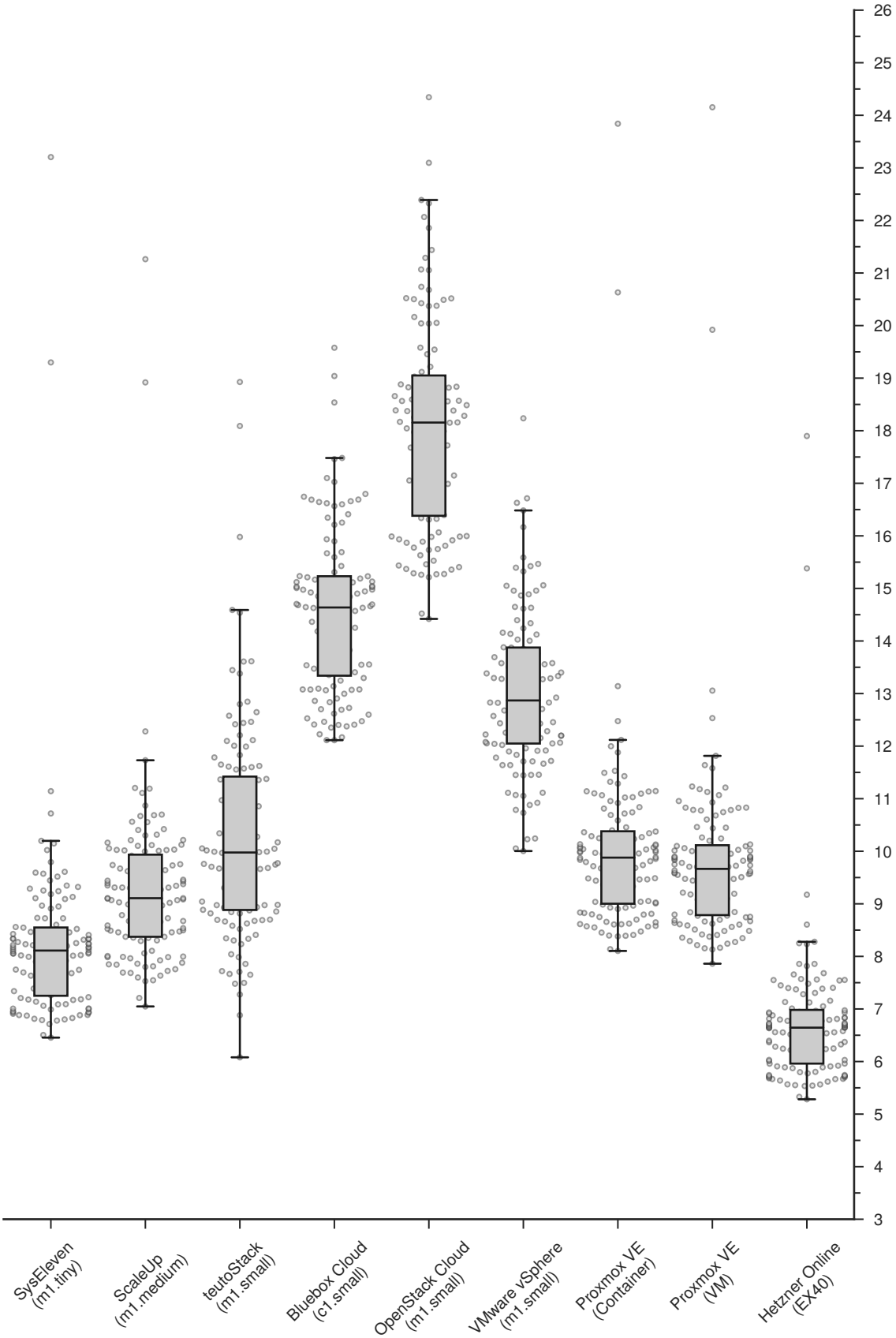


Abbildung 8.11: Bearbeitungszeiten als Box-Whisker-Diagramm


26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
SysEleven (m1.tiny)
ScaleUp (m1.medium)
teutoStack (m1.small)
Bluebox Cloud (c1.small)
OpenStack Cloud (m1.small)
VMware vSphere (m1.small)
Proxmox VE (Container)
Proxmox VE (VM)
Hetzner Online (EX40)

8.4.1.1 Vorgehen

Nicht-interaktiven Services wird das Black-Box-Modell unterstellt, d. h. Arbeits- und Funktionsweise gelten als unbekannt und bleiben unberücksichtigt. Bekannt ist lediglich, dass als Reaktion auf eine Eingabe nach einer gewissen Zeit eine Ausgabe vorliegt. Der Service gilt als fehlerfrei und nicht veränderbar. Nicht-funktionale Anforderungen umfassen ein Zeitlimit, das sich auf das Antwortzeitverhalten der Black Box bezieht. Dazu gilt es, ein zur Ausführung geeignetes Flavor zu ermitteln. Um einen Zusammenhang zwischen Flavor und Bearbeitungszeit herzustellen, sind auf der Basis von Testdaten Berechnungen auf den Flavors durchzuführen. Mit Hilfe einer statistischen Analyse lässt sich dann eine Aussage über die zu erwartende Bearbeitungszeit ableiten. Da sich insbesondere beim Einsatz von Virtualisierung Störgrößen wie die aktuelle Hintergrundbelastung und Verzögerungen, verursacht durch die gemeinsame Ressourcennutzung, negativ auf den Arbeitsfortschritt auswirken, entspricht das Ergebnis einer Wahrscheinlichkeitsverteilung über die Bearbeitungszeit.

8.4.1.2 Versuchsaufbau

Die Rolle des nicht-interaktiven Services nimmt der Service zur kortikalen Rekonstruktion des Gehirns ein, dessen Bereitstellung gemäß Abschnitt 8.2.2.2 auf der Basis von FreeSurfer erfolgt. Dabei bildet die in Abschnitt 8.2.1 festgelegte Experimentumgebung, im Speziellen die in der Tabelle 8.1 aufgeführten Flavors, die Versuchsgrundlage. Die Eingabesequenz umfasst 120 unterschiedliche Subjekte, wobei jedem Subjekt ein MRT-Datensatz zugeordnet ist, der eine Aufnahme des Gehirns enthält. Die MRT-Datensätze entstammen dem Projekt *OASIS Brains*, das im Zuge der *Open Access Series of Imaging Studies* die Bilddateien zu wissenschaftlichen Zwecken zur Verfügung stellt. Die Eingabedaten entsprechen den ersten 120 Subjekten, die dem Release OASIS-3 entstammen.

8.4.1.3 Ergebnisse

Zur Datenerhebung wurde von allen in der Experimentumgebung verfügbaren Flavors Nodes erstellt und diese als MRT-Bildanalysierer klassifiziert. Anschließend wurde ein Prozess gestartet, der die 120 Analysevorgänge sequenziell ausführte. Nach ihrem Abschluss erfolgte die Extraktion der Laufzeiten aus den Logdateien. Dadurch ergaben sich insgesamt 18 Messreihen, wobei jede Messreihe 120 Beobachtungswerte umfasste. Die Abbildung 8.11 stellt die Ergebnisse in Form eines Box-Whisker-Diagramms dar. Auf der Abszisse sind die Flavors, auf der Ordinate die Bearbeitungszeiten abgetragen. Das Diagramm ermöglicht einen visuellen Vergleich zwischen der Leistungsfähigkeit der Flavors. Die Box entspricht dem Bereich, in dem sich die mittleren 50 % der Beobachtungswerte befinden. Sie wird begrenzt vom ersten und dritten Quantil. Der sich innerhalb der Box befindende durchgehende Strich repräsentiert das zweite Quantil, den Median. Die Längen der Whisker folgen der Regel von John W. Tukey [Tuk77]. Die um Ausreißer bereinigten Messreihen bilden die Ausgangsbasis zur Abschätzung der Wahrscheinlichkeitsverteilungen. Untersucht wurden insgesamt 98 kontinuierliche Wahrscheinlichkeitsverteilungen. Dafür kam die Python-basierte Softwarebibliothek für wissenschaftliche Berechnungen *SciPy* [JOP+18] zum Einsatz. Ausgewählt wurde stets diejenige Verteilung, die in Bezug zur Messreihe die minimale *Summe der quadratischen Abweichungen* (engl. *Sum of Squared Errors*, SSE) aufwies. Die Tabelle 8.2 zeigt die Ergebnisse und ordnet jedem Flavor eine Wahrscheinlichkeitsverteilung zu. Die Funktionsvorschrift lässt sich aus der Verteilung, dem Lageparameter (*Location*), dem Skalierungsparameter (*Scale*) sowie den Formgebungsparame-

			Parameterschätzung						KS-Test	
Provider	**Flavor**	**Verteilung**	**Lage**	**Skalierung**	**Formgebung**		**Q(0,9)**	**SSE**	**D**	**p**
Amazon Web Services	c4.large	Johnson SB	5,609	3,935	a = 0,488	b = 1,034	8,30	0,047	0,082	0,397
Microsoft Azure	Standard_D1_v2	Generalized Gamma	7,132	2,874	a = 0,516	c = 3,302	10,305	0,069	0,074	0,549
	Standard_D2_v3	Exponentiated Weibull	5,946	2,358	a = 1,230	c = 2,108	9,587	0,018	0,045	0,972
IBM Cloud	BL2.1x4x100	Generalized Gamma	8,883	3,740	a = 0,426	c = 3,532	12,813	0,021	0,066	0,704
Google Cloud	n1-standard-1	Exponential Power	7,225	4,643	b = 1,447		12,475	0,002	0,044	0,978
Open Telekom Cloud	s1.medium	Exponentiated Weibull	8,689	3,256	a = 0,663	c = 2,655	12,851	0,004	0,049	0,971
	h1.large	Chi	5,954	1,826	df = 1,497		9,466	0,013	0,058	0,827
ProfitBricks	AMD	Folded Normal	8,869	1,268	c = 1,576		12,492	0,026	0,068	0,644
	Intel	Folded Normal	5,644	0,871	c = 1,705		8,247	0,069	0,052	0,904
SysEleven	m1.tiny	Folded Normal	6,455	0,956	c = 1,598		9,208	0,076	0,081	0,414
ScaleUp	m1.medium	Johnson SB	6,383	7,593	a = 0,980	b = 1,588	10,539	0,007	0,058	0,817
teutoStack	m1.small	Johnson SB	3,765	18,760	a = 1,689	b = 2,414	12,354	0,001	0,059	0,813
Bluebox Cloud	c1.small	Folded Normal	12,113	1,526	c = 1,454		16,288	0,034	0,086	0,342
OpenStack Cloud	m1.small	Exponentiated Weibull	14,340	5,297	a = 0,470	c = 3,263	20,464	0,007	0,064	0,749
VMware vSphere	m1.small	Generalized Extreme Value	12,389	1,283	c = 0,212		14,687	0,015	0,039	0,995
Proxmox VE	Container	Generalized Gamma	8,085	3,001	a = 0,333	c = 3,968	11,062	0,046	0,056	0,857
	VM	Beta	7,790	4,416	a = 1,984	b = 2,888	10,839	0,075	0,082	0,405
Hetzner Online	EX40	Johnson SB	5,124	3,637	a = 0,545	b = 1,072	7,544	0,104	0,076	0,505

Tabelle 8.2: Abschätzung der Bearbeitungszeit zur Flavorauswahl

tern (*Shape*) bestimmen. Des Weiteren ist das 0,9-Quantil aufgeführt, das im Zusammenhang mit der Bereitstellungs- und der Migrationsregel von Bedeutung ist. Die Tabelle enthält neben der SSE auch die Ergebnisse des *Kolmogorov-Smirnov-Tests* (*KS-Tests*) in Form der *D*-Statistik und des *p*-Werts. Über alle Messreihen hinweg betrachtet, bewegt sich der SSE zwischen 0,001 und 0,104. Die Werte der *D*-Statistik liegen zwischen 0,039 und 0,082. Die sehr kleinen Werte der SSE und der *D*-Statistik lassen den Schluss zu, dass die Verteilung von den Messwerten nur geringfügig abweicht. Zudem bewegt sich der *p*-Wert zwischen 0,342 und 0,972. Bei einem Signifikanzniveau von $\alpha = 0{,}05$ gilt für alle *p*-Werte:

$$p > \alpha$$

Basierend auf diesem Ergebnis lässt sich keine der angenommenen Verteilungsfunktionen ablehnen, ihre Annahme ist damit gerechtfertigt.

8.4.1.4 Einordnung

Dieses Experiment zeigt auf, wie sich die Bearbeitungszeit nicht-interaktiver Services auf unterschiedlichen Flavors abschätzen lässt. Innerhalb des Nutzungsszenarios dienen die Wahrscheinlichkeitsverteilungen dem Flavorauswahlausdruck als Berechnungsgrundlage. Dadurch lässt sich ein Flavor ermitteln, das mit hoher Wahrscheinlichkeit dazu in der Lage ist, die Auftragsausführung unter Einhaltung einer zeitlichen Zielvorgabe abzuschließen. Auf Basis der Funktionsvorschriften kann letztlich für einen Auftrag erst entschieden werden, ob sich Zeitvorgaben überhaupt einhalten lassen und welche Flavors für den gütegesicherten Servicebetrieb in Betracht kommen. Die Black-Box-Annahme gestattet es, den Lösungsansatz auf andere nicht-interaktive Services zu übertragen.

8.4.2 Abschätzung des Verarbeitungsfortschritts

Das zweite Experiment soll aufzeigen, wie sich der Fortschritt einer Auftragsverarbeitung näherungsweise bestimmen lässt, auch wenn der Service keine derartige Zustandsgröße zur Verfügung stellt. Durch den Einsatz von Heuristiken und zufallsgesteuerten Programmabläufen ist selbst aus der Anwendungsfunktion heraus eine exakte Berechnung häufig nicht möglich. Im Umfeld eines gütegesicherten Servicebetriebs wird aber die Berechenbarkeit des Fortschritts vorausgesetzt. Die Wertermittlung ist insbesondere für langlaufende Services, wie bei der MRT-Bildanalyse, von zentraler Bedeutung, da sich anhand des Ergebnisses abschätzen lässt, ob die Auftragsausführung planmäßig erfolgt oder eine Zielverfehlung zu erwarten ist.

8.4.2.1 Vorgehen

Im Zuge einer Black-Box-Betrachtung gilt es, eine Zustandsgröße zu identifizieren, die sich von außen beobachten lässt, keinerlei domänenspezifisches Wissen erfordert und gleichzeitig kumulativ ist. Zudem sollte sie sich dadurch auszeichnen, dass sie für jeden Verarbeitungsvorgang um einen konstanten Wert wächst. Diese Voraussetzung dürfte vielfach nicht erfüllt sein. In der Regel wird der Wertzuwachs Schwankungen unterworfen sein. Dementsprechend lässt sich das Merkmal lediglich näherungsweise mit Hilfe einer Wahrscheinlichkeitsverteilung beschreiben. Ziel ist die Ermittlung einer oberen Schranke für die kumulative Zustandsgröße, die bei einer Auftragsverarbeitung mit hoher Wahrscheinlichkeit nicht überschritten, aber zugleich auch nur selten deutlich unterschritten wird.

8.4.2.2 Versuchsaufbau

Die Rolle des nicht-interaktiven Services nimmt wiederum der Service zur kortikalen Rekonstruktion des Gehirns auf der Basis von FreeSurfer ein, der keinerlei prozentuale Fortschrittsinformationen bereitstellt. In diesem speziellen Fall erfüllt die Größe der Logdatei das Kriterium einer kumulativen Zustandsgröße. Jedem Subjekt ist eine eigene Logdatei zugeordnet, die pro Analysevorgang neu erstellt wird. Sie enthält domänenspezifische Informationen über die derzeit ausgeführte Pipeline-Stufe. Basierend auf der Black-Box-Annahme ist eine anwendungsspezifische Interpretation der Daten ausgeschlossen. Dementsprechend gilt es, eine vom Nutzungsszenario unabhängige Metrik festzulegen. Die Größe der Logdatei lässt sich mit Hilfe der beiden folgenden Metriken messen:

- Dateigröße
- Zeilenanzahl

Die Aufgabe besteht nun darin, die zur Merkmalsbeschreibung geeigneten Verteilungsfunktionen zu ermitteln. Die Abschätzung erfolgt wiederum auf der Basis von Messreihen, für deren Erhebung auf die bereits zuvor verwendenden 120 Subjekte aus dem OASIS-Brains-Projekt zurückgegriffen wird.

8.4.2.3 Ergebnisse

Die nachfolgende Tabelle 8.3 enthält die Ergebnisse und ordnet jedem der beiden Merkmale eine Wahrscheinlichkeitsverteilung zu:

		Parameterschätzung					KS-Test	
Merkmal	**Verteilung**	**Lage**	**Skalierung**	**Formgebung**		**SSE**	**D**	**p**
Dateigröße	Johnson SU	449,394	6,865	$a = -1{,}766$	$b = 1{,}295$	0,000	0,046	0,969
Zeilenanzahl	Johnson SU	10 215,002	107,442	$a = -1{,}701$	$b = 1{,}094$	0,000	0,056	0,861

Tabelle 8.3: Wahrscheinlichkeitsverteilungen für die Logdateigröße

Die Standardabweichung der Dateigröße beträgt 16,942 kB, die der Zeilenanzahl 450,262. Die Werte sind sehr klein und belegen die Eignung beider Merkmale zur Fortschrittsabschätzung. Mit Hilfe der obigen Wahrscheinlichkeitsverteilungen lässt sich die Logdateigröße nach oben abschätzen. Die Grundlage bildet das 0,99-Quantil. Demnach werden in 99 % der Fälle die folgenden Werte nicht überschritten:

$$\text{Dateigröße} \leq 530{,}16 \text{ kB}$$
$$\text{Zeilenanzahl} \leq 12346$$

Unter der Annahme einer linearen Wachstumsrate entspricht ein Fortschritt um 1 % einer Zunahme der Dateigröße um 5,3016 kB bzw. einem Anstieg der Zeilenanzahl um 123,46.

8.4.2.4 Einordnung

Um nicht-interaktive Services gütegesichert erbringen zu können, muss der prozentuale Verarbeitungsfortschritt in Form einer Zustandsgröße repräsentiert werden. Problematisch ist,

wenn der Service selbst keine Fortschrittsinformationen bereitstellt. In diesem Fall gilt es, eine kumulative Größe zu identifizieren, die von außen beobachtbar ist und die zugleich mit dem Fortschritt wächst. Das Experiment hat aufgezeigt, wie sich beispielsweise anhand der Logdateigröße der Fortschritt eines Verarbeitungsvorgangs näherungsweise bestimmen lässt. Diese Größe dient dazu, die Rechtzeitigkeit der Ausführung zu beurteilen und bildet gemeinsam mit der bereits verstrichenen Bearbeitungszeit und dem Zeitlimit die Eingabeparameter des Zielabweichungsausdrucks.

8.4.3 Abschätzung der Kosten

Neben dem Zeitlimit muss der Flavorauswahlausdruck auch das Kostenlimit berücksichtigen. Dies setzt voraus, dass für jedes Flavor die zu erwartenden Kosten für eine darauf ausgeführte MRT-Bildanalyse vorliegen. Die Erwartungswerte der Wahrscheinlichkeitsverteilungen aus der Tabelle 8.2 bilden den Ausgangspunkt der Kostenabschätzung. Die nachfolgende Abbildung 8.12 zeigt die Ergebnisse:

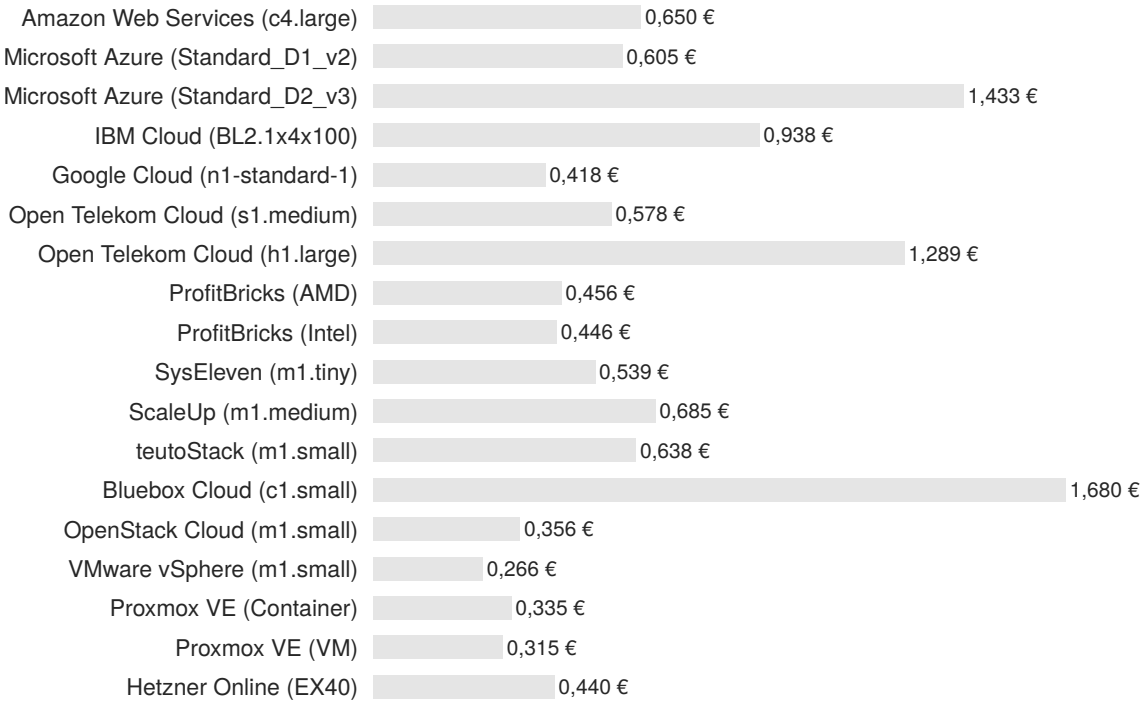


Abbildung 8.12: Erwartete Kosten einer MRT-Bildanalyse

8.5 Validierungsexperimente

Anhand von zwei Experimenten soll der Nachweis erbracht werden, dass sich nicht-interaktive Services mit Hilfe des Informationsmodells und der Steuerungsmuster in einer Multi-Provider-Umgebung gütegesichert erbringen lassen. Das Experiment zur Vereinheitlichung der Node-Bereitstellung dient als Beleg dafür, dass die am Verbund beteiligten cloudbasierten und virtuellen Umgebungen gleich behandelbar sind. Den Abschluss bildet ein Experiment zur Gewährleistung von Servicegüteanforderungen.

8.5.1 Vereinheitlichung der Node-Bereitstellung

Das erste Validierungsexperiment zeigt auf, wie sich Nodes in einer Multi-Provider-Umgebung einheitlich bereitstellen lassen. Zwar bietet das Managementobjekt des Provider-Services eine umgebungsübergreifende einheitliche Schnittstelle, dennoch gilt es, im Zuge der Verarbeitung die Konfigurationsvariablen auf die Managementschnittstelle der Ausführungsumgebung abzubilden. Ziel des Experiments ist es nachzuweisen, dass die technische und die semantische Heterogenität überwindbar sind.

8.5.1.1 Vorgehen

Den Ausgangspunkt bilden die Konfigurationen zweier Provider-Services, die die Node-Bereitstellung im jeweiligen Projekt bewirken. An dieser Stelle soll der Zusammenhang zwischen den über das Managementobjekt bereitgestellten Konfigurationsvariablen und der konkreten technischen Konfiguration der zugrundeliegenden Managementfunktion erläutert werden. Sie übernimmt die Anbindung der Managementschnittstelle und verantwortet die Befehlsausführung zur Anpassung der Projektumgebung.

8.5.1.2 Versuchsaufbau

Die Abbildung der Konfigurationen wird exemplarisch anhand zweier Ausführungsumgebungen gezeigt, die über unterschiedliche Managementschnittstellen verfügen. Es kommen die folgenden cloudbasierten Umgebungen zum Einsatz:

- OpenStack Cloud
- Amazon Web Services

Das nachfolgende Listing zeigt die Belegung der Konfigurationsvariablen der für das projektseitige Infrastrukturmanagement verantwortlichen Provider-Services:

```
# OpenStack Cloud (OSC)
Nodes/Host-OSC/Zustand = "Aktiv"
Nodes/Host-OSC/Name = "Host-OSC"
Nodes/Host-OSC/Flavor = "m1.small"
Nodes/Host-OSC/Image = "Ubuntu 16.04"
Nodes/Host-OSC/Netzwerke = "Service-Netzwerk"

# Amazon Web Services (AWS)
Nodes/Host-AWS/Zustand = "Aktiv"
Nodes/Host-AWS/Name = "Host-AWS"
Nodes/Host-AWS/Flavor = "c4.large"
Nodes/Host-AWS/Image = "Ubuntu 16.04"
Nodes/Host-AWS/Netzwerke = "Service-Netzwerk"
```

Das Infrastrukturmanagement nutzt das von Charlie Sharpsteen entwickelte Vagrant-Plugin `oscar-stack/vagrant-config_builder`. Es wurde im Rahmen dieser Arbeit aktualisiert und um die Fähigkeiten zur Anbindung weiterer cloudbasierter und virtueller Umgebungen ergänzt, unter anderem von OpenStack. Ein Teil der Erweiterungen ist im Zuge des Release 1.0 veröffentlicht und der Community zur freien Nutzung überlassen worden. Auf jedem der beiden Provider-Nodes wurde zudem das zur Anbindung der Managementschnittstelle erforderliche Provider-Plugin bereitgestellt. Dabei kamen folgende Implementierungen zum Einsatz:

- `cloudbau/vagrant-openstack-plugin`
- `mitchellh/vagrant-aws`

8.5.1.3 Ergebnisse

Unter diesen technischen Voraussetzungen lassen sich die Projektumgebungen vollständig deklarativ beschreiben. Die Abbildung der obigen Konfigurationsvariablen führt somit zu zwei ähnlich aussehenden Konfigurationen. Das folgende Listing zeigt die Konfiguration für die OpenStack Cloud:

```
---
vms:
- name: Host-OSC
  box: dummy
  providers:
  - server_name: Host-OSC
    type: openstack_plugin
    endpoint: 'https://os-cloud:5000/v3/auth/tokens'
    tenant: neuroimaging
    project_name: neuroimaging
    username: vagrant
    api_key: ********
    project_domain: default
    user_domain: default
    keypair_name: vagrant
    ssh_username: ubuntu
    flavor: m1.small
    # Ubuntu 16.04 LTS (Xenial Xerus)
    image: b78bbc04-6dbe-4f13-af4a-62f6c3699699
    networks:
    - service-net
    overrides:
      ssh:
        private_key_path = '/home/vagrant/.ssh/id_rsa'
```

Das nachfolgende Listing zeigt die Konfiguration für Amazon Web Services:

```
---
vms:
- name: Host-AWS
  box: dummy
  providers:
  - type: aws
    access_key_id: ********
    secret_access_key: ********
    keypair_name: vagrant
    # Ubuntu 16.04 LTS (Xenial Xerus)
    ami: ami-0dd0be70cc0d493b7
    # Deutschland (Frankfurt)
```

```
    region: eu-central-1
    instance_type: c4.large
    # Service-Netzwerk
    subnet_id: subnet-2f09a348
    associate_public_ip: false
    monitoring: true
    ssh_host_attribute: !ruby/sym private_ip_address
    tags:
      Name: Host-AWS
    overrides:
      ssh:
        username: ubuntu
        private_key_path = '/home/vagrant/.ssh/id_rsa'
```

Anschließend wurde Vagrant dazu aufgefordert, einen Abgleich des projektseitigen Istzustands mit dem durch die Konfigurationen festgelegten Sollzustand durchzuführen. Die obigen Konfigurationen führten zur Erzeugung zweier virtueller Maschinen in den jeweiligen Projekten der Ausführungsumgebungen.

8.5.1.4 Einordnung

Das Experiment zeigt am Beispiel der Node-Erstellung, wie sich die technische und die semantische Heterogenität der Ausführungsumgebungen überwinden lassen. Der Lösungsansatz sieht vor, die Zustände der Projektumgebungen deklarativ zu beschreiben. Dadurch kann von den Managementschnittstellen der Ausführungsumgebungen abstrahiert, eine einheitliche Ansteuerung cloudbasierter und virtueller Umgebungen erreicht und die technische Heterogenität überwunden werden. Die Grundlage dafür bilden Vagrant-Plugins, die die deklarativen Konfigurationen auf die funktionalen Managementschnittstellen abbilden. Mit Hilfe der Managementobjekte wird eine Abstraktionsschicht oberhalb der Managementfunktion aufgebaut. Die vom Provider bereitgestellten Konfigurationsvariablen leisten zur Überwindung der semantischen Heterogenität einen wichtigen Beitrag. Dies lässt sich unmittelbar an den beiden obigen Konfigurationen für die OpenStack Cloud und Amazon Web Services erkennen. Während bei OpenStack von Flavors und Images gesprochen wird, sind in Amazon Web Services Instanztypen und Amazon Machine Images (AMIs) äquivalente Begriffe. Im Informationsmodell sind diese Elemente einheitlich als Flavors und Images repräsentiert. Dementsprechend macht auch das Managementobjekt des Providers ausschließlich Gebrauch von diesen Begrifflichkeiten. Die Abbildung auf die jeweilige Managementschnittstelle erfolgt in der Implementierung des Providers und damit außerhalb des Überwachungs- und Steuerungsprozesses. Dieser operiert ausschließlich auf der von Managementobjekten aufgebauten Abstraktionsschicht.

8.5.2 Gewährleistung von Servicegüteanforderungen

Das zweite Validierungsexperiment soll den Nachweis erbringen, dass die in Abschnitt 6.11 vorgestellten Steuerungsmuster in der Lage sind, Services in einer Multi-Provider-Umgebung anforderungsgerecht bereitzustellen und gütegesichert zu betreiben. Aufgabe der Policy-Regeln ist es trotz des Eintritts von Fehlern und Leistungsverschlechterungen, den Servicebetrieb weitestgehend aufrechtzuerhalten. Dabei gilt es nachzuweisen, dass die Servicebereitstellung nicht nur auf eine einzelne Ausführungsumgebung beschränkt bleibt. Vielmehr müssen sich die am Verbund beteiligten Ausführungsumgebungen flexibel einsetzen lassen.

8.5.2.1 Vorgehen

Zu diesem Zweck wird in das System ein Arbeitsauftrag eingebracht, den es von einem nicht-interaktiven Service zu verarbeiten gilt. Die Servicegütevereinbarung beinhaltet ein Zeit- und ein Kostenlimit, beide Zielvorgaben sind bei der Auftragsverarbeitung einzuhalten. Während der Ausführung wird der Service durch eine künstlich erzeugte Hintergrundlast mehrmals so stark verlangsamt, dass die Rechtzeitigkeit der Ausführung immer wieder einen kritischen Wert annimmt. Die Eingriffe erfolgen stets dann, wenn die Berechnung einen Fortschritt von 20 % gemacht hat.

8.5.2.2 Versuchsaufbau

Die Rolle des nicht-interaktiven Services nimmt wiederum der Service zur kortikalen Rekonstruktion des Gehirns ein, der von FreeSurfer bereitgestellt wird. Als Eingabedaten dienen die bereits verwendeten 120 Subjekte des OASIS-Brains-Projekts. Das Zeitlimit zur Ausführung einer MRT-Bildanalyse beträgt 12,5 h, als Kostenlimit dient ein Wert von 1,50 € . Die automatisierte Provisionierung von FreeSurfer erfordert zunächst eine Optimierung des Softwareartefakts. Dieses liegt derzeit in Version 6 vor und hat eine Größe von 4,6 GB, was den Aufsetzungsprozess deutlich verlangsamt. Es hat sich gezeigt, dass ein Großteil der ausgelieferten Programmdateien zur Ausführung der Pipeline nicht erforderlich ist. Dementsprechend wurde für das Experiment ein optimiertes Softwareartefakt erstellt. Dabei ließen sich folgende Verbesserungen erzielen:

- Verringerung der Archivgröße um 93,35 % auf 312 MB
- Reduzierung der Dateianzahl um 98,77 % von 22 474 auf 277
- Verkürzung der Entpackzeit um 95,2 %

8.5.2.3 Ergebnisse

Das Managementsystem war in 95 % der Fälle in der Lage, den gütegesicherten Servicebetrieb aufrechtzuerhalten. Lediglich in 5 % der Fälle fand eine Überschreitung des Zeitlimits statt. Die Arbeitsweise der Policy-Regeln soll anhand eines ausgewählten Analysevorgangs veranschaulicht werden. Das Hinzufügen eines Arbeitsauftrags zur Durchführung einer MRT-Bildanalyse löste die Ausführung der Bereitstellungsregel aus. Sie veranlasste mit Hilfe des Flavorauswahlausdrucks die Erzeugung eines Containers innerhalb der Proxmox VE sowie die Erstellung eines Services zur kortikalen Rekonstruktion auf dem Zielsystem. Dieser Vorgang war nach 6,967 min abgeschlossen und der Analysevorgang begann. Durch eine künstliche Verlangsamung wurden Leistungsbeeinträchtigungen simuliert. Es ergaben sich insgesamt vier Eingriffszeitpunkte, nämlich bei einem Fortschritt von 20 %, 40 %, 60 % und 80 %. Dies führte dazu, dass die Zielabweichung mehrmals einen als kritisch einzustufenden Wert von 10 % und mehr annahm. Der Wert wurde jeweils nach 44,4 min, 53,28 min, 66,36 min bzw. 75,9 min erreicht. Der Verlauf der Zielabweichung ist in der Abbildung 8.13 dargestellt. Die Rechtzeitigkeits- und die Fortschrittsregel waren aufgrund der Schwellenwertüberschreitungen dazu veranlasst, die Rechtzeitigkeit der Ausführung als kritisch einzustufen, was sich in der Belegung des Servicegüteparameters widerspiegelte. Die Wertänderung löste ihrerseits die Ausführung der Migrationsregel aus, die als adaptive Maßnahme die Servicemigration auf ein leistungsstärkeres Flavor veranlasste. Der Vorgang wiederholte sich insgesamt vier Mal, und zwar alle 2,75 h. Dies führte zu folgender Migrationssequenz:

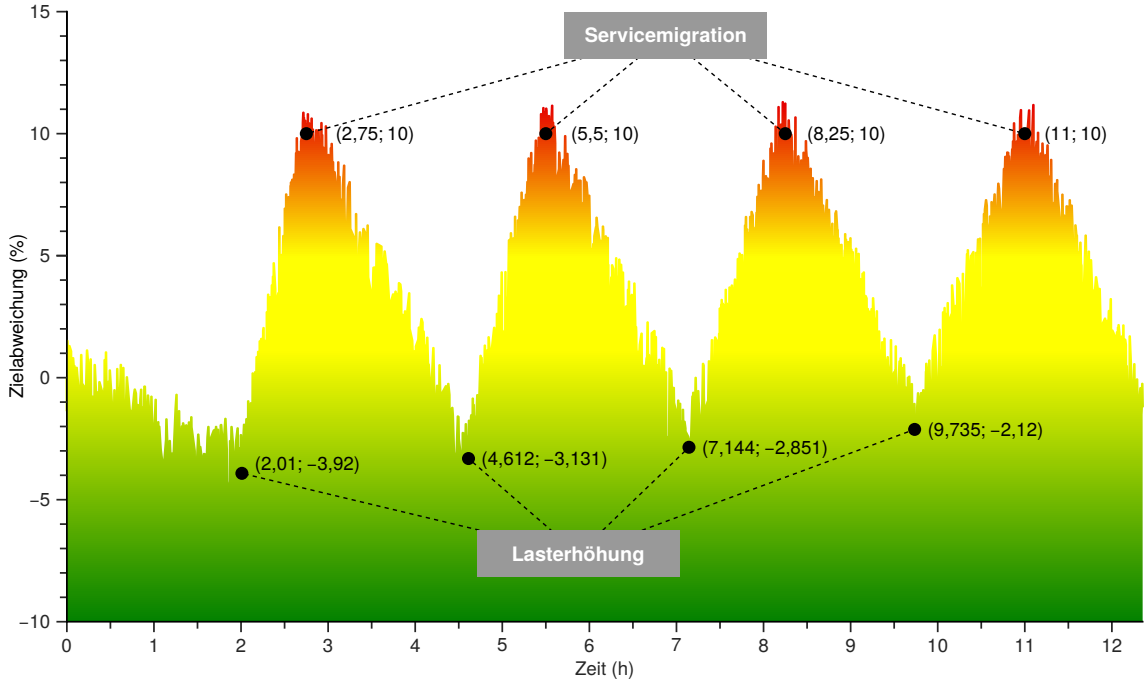


Abbildung 8.13: Verlauf der Zielabweichung

Proxmox VE ► ScaleUp ► Open Telekom Cloud ► Amazon Web Services ► Hetzner Online

Die Migrationssequenz enthielt folgende Übergänge:

- Wechsel des Anbieters bei gleicher Managementschnittstelle:
  ScaleUp ► Open Telekom Cloud
- Wechsel des Anbieters und der Managementschnittstelle:
  Open Telekom Cloud ► Amazon Web Services
- Wechsel der Ausführungsumgebung:
  - Migration von einer virtuellen Umgebung in eine cloudbasierte Umgebung:
    Proxmox VE ► ScaleUp
  - Migration von einer cloudbasierten Umgebung in eine physikalische Umgebung:
    Amazon Web Services ► Hetzner Online
- Wechsel der Virtualisierungstechnologie:
  Proxmox VE ► ScaleUp

Die Tabelle 8.4 enthält die Zeitdauern der Servicemigrationen. Die Angaben schließen die Node-Erstellung, gegebenenfalls den Bootvorgang sowie das Aufsetzen und Einrichten des Applikationsstapels mit ein. Gemessen wurde die Zeitspanne zwischen dem Absetzen der Anfrage durch die Migrationsregel bis zum Start der MRT-Bildanalyse auf dem Zielsystem. Durch die Servicemigration ließ sich mehrfach eine deutliche Verbesserung der Zielabweichung erreichen. Die Berechnung war nach 85,081 min, 76,757 min, 73,515 min bzw. 72,651 min wieder planmäßig und terminierte nach 12,353 h, 8,82 min vor der Zeitvorgabe.

Start		► Ziel		
Provider	**Flavor**	**Provider**	**Flavor**	**Zeitdauer**
Proxmox VE	Container	ScaleUp	m1.medium	3,52 min
ScaleUp	m1.medium	Open Telekom Cloud	h1.large	3,15 min
Open Telekom Cloud	h1.large	Amazon Web Services	c4.large	2,88 min
Amazon Web Services	c4.large	Hetzner Online	EX40	2,31 min

Tabelle 8.4: Zeitdauern der Servicemigrationen

8.5.2.4 Einordnung

Das Experiment führt die zuvor erzielten Teilergebnisse zusammen und dient als Nachweis dafür, dass sich die an einer Multi-Provider-Umgebung beteiligten Ausführungsumgebungen zur Serviceerbringung flexibel und anforderungsgerecht einsetzen lassen. Zudem zeigt es, dass die Policys in der Lage sind, den gütegesicherten Servicebetrieb auch bei sich abzeichnenden Zielverfehlungen zu gewährleisten. Eine dieser Maßnahmen kann die Servicemigration auf ein leistungsstärkeres Flavor sein. Im Rahmen des Experiments gelang es in 95 % der Fälle, die Zeitvorgabe trotz mehrmaliger Leistungsverschlechterungen einzuhalten. Dieses Ergebnis zeigt sehr deutlich den Mehrwert des Managementsystems und belegt gleichzeitig, dass die Lösungsansätze zur Überwachung und Steuerung nicht-interaktiver Services tragfähig sind. Es ist insbesondere gelungen, die vormals scharfen Grenzen zwischen den Ausführungsumgebungen zu überwinden. Die Migrationssequenz dient dafür als eindeutiger Beleg. Sie umfasst alle innerhalb eines Verbunds von heterogenen Ausführungsumgebungen möglichen Übergänge. Dabei erfolgte die Serviceausführung neben cloudbasierten auch auf virtuellen und physikalischen Umgebungen. Zudem dienten Container, virtuelle Maschinen und Bare-Metal-Server als gleichberechtigte Host-Systeme. Die Betriebstechnologie eines Nodes tritt vollständig in den Hintergrund. Letztlich stellen die Leistungsfähigkeit und die Kosten die alleinigen Entscheidungskriterien für den Einsatz eines Flavors dar.

9 Schluss

Die im Rahmen der Arbeit erzielten Ergebnisse wurden über mehrere Forschungsprojekte hinweg kontinuierlich weiterentwickelt und erprobt. Dabei kam das Managementsystem in unterschiedlichsten Anwendungsdomänen zum Einsatz. Es ermöglichte neben dem Servicemanagement in medizinischen Umgebungen (OSAmI, Medolution) auch das Management cloudbasierter Umgebungen (EASI-CLOUDS) sowie das Management von Gebäudeautomatisierungssystemen (BaaS). Den Ausgangspunkt bildete stets das in Kapitel 6 beschriebene Informationsmodell, das jeweils um domänenspezifische Managementobjekte erweitert wurde. Die beiden folgenden Abschnitte schließen die Arbeit mit einer Zusammenfassung und einem Ausblick ab. Es werden offene Punkte und Erweiterungsmöglichkeiten für zukünftige Forschungsaktivitäten aufgeführt.

9.1 Zusammenfassung

Die Arbeit leistet einen Beitrag zum einheitlichen Management serviceorientierter Systeme in einer Multi-Provider-Umgebung. Das zentrale Ergebnis ist ein Informationsmodell für eine Multi-Provider-Umgebung, das managementrelevante Ressourcen identifiziert und durch Managementobjekte beschreibt. Hierbei wird die Multi-Provider-Umgebung selbst als Managementobjekt repräsentiert. Indem physikalische, virtuelle und cloudbasierte Umgebungen durch dieselbe Managementobjektklasse beschrieben werden, lässt sich die semantische Heterogenität der Managementinformationen überwinden. Das Informationsmodell ordnet der Multi-Provider-Umgebung ihre Ausführungsumgebungen zu und unterteilt diese in Projekte. Sie stellen vollständig voneinander isolierte Verwaltungseinheiten dar und können über eine Infrastruktur-, Plattform- und Anwendungsschicht verfügen. Die Infrastrukturschicht umfasst Ressourcen zur Datenverarbeitung, -kommunikation und -speicherung, die Plattformschicht erfasst Middleware und Laufzeitumgebungen und die Anwendungsschicht beinhaltet Applikationen und Services. Sie werden von Hardware-, Software- und Speicherkomponenten erbracht. Gleichwohl lassen sich auch fremdbezogene Services im Sinne des serverlosen Computings erfassen, bei denen keinerlei Informationen über Bereitstellung und Betrieb vorliegen. Neben den Vertragsbeziehungen zwischen Nutzer, Kunde, Anbieter und Zulieferer werden auch die Service-

güteanforderungen abgebildet. Sie enthalten Zielvorgaben in Form von Service-Level-Objectives, die sich auf Servicegüteparameter beziehen. Im Zusammenhang mit den Managementfunktionen wurde ein funktionales Rollenmodell entwickelt, das die für das Management erforderlichen Funktionseinheiten klassifiziert und diese durch Managementobjekte beschreibt. Die MESEA-Kontrollschleife basiert auf dem von IBM entwickelten MAPE-K-Regelkreis und ergänzt die Ausführungsphase um eine zusätzliche Anwendungsphase, die eine effiziente Anpassung der Multi-Provider-Umgebung erlaubt. Im Rahmen einer organisatorischen Projektstruktur wurden Interaktions- und Kommunikationsbeziehungen zwischen den Funktionseinheiten festgelegt, so dass sich Projekte eigenverantwortlich überwachen und steuern lassen. Der Managementprozess wird auf Grundlage von Policy-Regeln erbracht. Es handelt sich um ECA-Regeln, die auf Änderungen von Managementobjekten reagieren und bei Vorliegen einer positiven Bedingungsüberprüfung eine Aktion ausführen. Dazu können sie die Auswertung von Policy-Ausdrücken anfordern. Mit Hilfe von Managementdomänen lassen sich Managementobjekte nach organisatorischen Gesichtspunkten gruppieren. Die rollenbasierte Zugangskontrolle gewährleistet, dass keine unautorisierte Informationsgewinnung möglich ist. Das Managementsystem verdeckt die technische Heterogenität und bietet eine vereinheitlichte Sicht auf die Managementinformationen. Zur Informationsintegration lassen sich unterschiedlichste Datenquellen anbinden. Der Managementbaum stellt eine übergeordnete Zugriffsstruktur zur Verfügung und beschreibt Managementobjekte in Form einer hierarchischen Baumstruktur, die sich auslesen und verändern lässt. Managementbäume bilden eigenständige Verwaltungsdomänen, die sich untereinander hierarchisch verschachteln lassen. Auf diese Weise lässt sich die Multi-Provider-Umgebung durch ein zusammenhängendes Managementobjekt beschreiben. Im Rahmen eines Anwendungsfalls ließ sich experimentell nachweisen, dass das Informationsmodell und das Managementsystem eine tragfähige Lösung für das einheitliche Management serviceorientierter Systeme in einer Multi-Provider-Umgebung bilden. Das adaptive Management stützt sich ausschließlich auf Policy-Regeln und Policy-Ausdrücke, die sich mit geringem Änderungsaufwand in anderen Nutzungsszenarien einsetzen lassen. In diesem Zusammenhang wurde aufgezeigt, wie sich nicht-interaktive Services auf Basis von Steuerungsmustern anforderungsgerecht bereitstellen und gütegesichert betreiben lassen. Die vereinheitlichten Managementobjekte bilden die Grundvoraussetzung für den flexiblen Einsatz der an einer Multi-Provider-Umgebung beteiligten Ausführungsumgebungen. Mit Hilfe dieses Ansatzes ist es gelungen, die vormals scharfen Grenzen zwischen den Ausführungsumgebungen zu überwinden.

9.2 Ausblick

Das Management serviceorientierter Systeme in einer Multi-Provider-Umgebung ist ein noch wenig erforschtes Gebiet und bedarf weiterer Forschungsaktivitäten. Im Folgenden sollen einige offene Punkte für zukünftige Arbeiten aufgeführt werden:

Modellbasiertes Management

Das policybasierte Management bietet vielfältige Erweiterungsmöglichkeiten. Insbesondere die Einbettung in das modellbasierte Management stellt eine nützliche Ergänzung dar. Die manuelle Implementierung der Steuerungsmuster ist zeitaufwendig und birgt ein hohes Fehlerpotenzial. Beim modellbasierten Management wird der Managementprozess um eine zusätzliche Planungsphase erweitert. Hierbei wird das reale System auf unterschiedlichen Abstraktionsniveaus modelliert und mit Bedingungen versehen. Die deklarativen Policys werden im Rahmen

eines automatisierten Policy-Refinements zu imperativen Policys abgeleitet. Der Vorgang stützt sich auf sogenannte *Verfeinerungs-* und *Kontrollmuster*, die den Ableitungsprozess lenken. Das Ergebnis der Codegenerierung ist eine Menge operationaler Policys, die das System so steuern, dass die Zielvorgaben eingehalten werden. Die Lösung basiert auf einer strikten Trennung zwischen der modellbasierten Systemplanung zur Entwicklungszeit und der effizienten Systemsteuerung mittels ausführbarer Policys zur Laufzeit. Im Zentrum der Betrachtung sollte die Identifikation von Verfeinerungs- und Kontrollmustern stehen, die das adaptive Management einer Multi-Provider-Umgebung ermöglichen.

Self-X-Eigenschaften

Selbstüberwachung und Selbstkonfiguration sind Grundvoraussetzung für weitere Self-X-Eigenschaften. Das im Rahmen der Arbeit entwickelte Managementsystem verfügt über diese beiden Merkmale. Darauf aufbauend lassen sich Eigenschaften wie die Selbstheilung und die Selbstoptimierung realisieren. Dazu gilt es, Metriken zu definieren, anhand derer sich entscheiden lässt, ob ein Systemzustand den Einsatz von Korrektur- und Optimierungsmaßnahmen erfordert. Zudem müssen Zielfunktionen und Nebenbedingungen festgelegt werden. Interessant dürfte zudem sein, Adaptionsmechanismen näher zu betrachten, bei denen die Managementinfrastruktur das Adaptionsobjekt darstellt. Dementsprechend lassen sich Situationen untersuchen, in denen das Management sich selbst optimiert, also beispielsweise Skalierungsmaßnahmen ergreift, um den Managementprozess oder die Managementfunktionen bei Leistungsengpässen zu beschleunigen.

Interaktive Services

Im Rahmen der Validierung wurde aufgezeigt, wie sich mit Hilfe adaptiver Maßnahmen Servicegüteanforderungen an nicht-interaktive Services gewährleisten lassen. Das Vorgehen kann jedoch nicht ohne Weiteres auch auf interaktive Services übertragen werden. Während für nicht-interaktive Services die Bearbeitungszeit das zentrale Qualitätsmaß darstellt, bleibt zunächst offen, ob sich ein solches auch für interaktive Services finden lässt. Im Gegensatz zu nicht-interaktiven Services entspricht die Kommunikation einer Interaktionssequenz, wobei eine Interaktion aus einer Eingabe und einer Ausgabe besteht, die nach einer gewissen Bearbeitungszeit vorliegt. Zwischen den einzelnen Interaktionen vergeht eine variable Zeitspanne. Einer Interaktion kann zwar das Prinzip der Jobverarbeitung unterstellt werden, inwiefern dieser Ansatz aber auch auf eine Sequenz übertragbar ist, lässt sich nicht ohne weitere Analysen beantworten.

Literaturverzeichnis

[ACU+10] AVANTS, Brian B. ; COOK, Philip A. ; UNGAR, Lyle ; GEE, James C. ; GROSSMAN, Murray: Dementia Induces Correlated Reductions in White Matter Integrity and Cortical Thickness: A Multivariate Neuroimaging Study with Sparse Canonical Correlation Analysis. In: *NeuroImage* 50 (2010), April, Nr. 3, S. 1004–1016

[ADB+99] ABOWD, Gregory D. ; DEY, Anind K. ; BROWN, Peter J. ; DAVIES, Nigel ; SMITH, Mark ; STEGGLES, Pete: Towards a Better Understanding of Context and Context-Awareness. In: *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, Springer-Verlag, 1999 (HUC '99), S. 304–307

[ADOB98] ABOWD, D. ; DEY, A. K. ; ORR, R. ; BROTHERTON, J.: Context-Awareness in Wearable and Ubiquitous Computing. In: *Virtual Reality* 3 (1998), September, Nr. 3, S. 200–211

[AGAF16] AMELLER, David ; GALSTER, Matthias ; AVGERIOU, Paris ; FRANCH, Xavier: A Survey on Quality Attributes in Service-based Systems. In: *Software Quality Journal* 24 (2016), Juni, Nr. 2, S. 271–299

[AK15] ABOLHASSAN, Ferri (Hrsg.) ; KELLERMANN, Jörn (Hrsg.): *Effizienz durch Automatisierung: Das „Zero Touch"-Prinzip im IT-Betrieb*. Springer Gabler, 2015

[AKS03] ALEMAN, André ; KAHN, René S. ; SELTEN, Jean-Paul: Sex differences in the risk of schizophrenia: Evidence from meta-analysis. In: *Archives of General Psychiatry* 60 (2003), Nr. 6, S. 565–571

[ALL05] AGRAWAL, D. ; LEE, Kang-Won ; LOBO, J.: Policy-based Management of Networked Computing Systems. In: *IEEE Communications Magazine* 43 (2005), Oktober, Nr. 10, S. 69–75

[Alz18] ALZHEIMER'S DISEASE INTERNATIONAL: *World Alzheimer Report*. September 2018

[AML89] ALBUS, James S. ; MCCAIN, Harry G. ; LUMIA, Roland: *NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)*. April 1989

[AP17] ALPAR, Paul ; POLYVIOU, Ariana: Management of Multi-cloud Computing. In: OSHRI, Ilan (Hrsg.) ; KOTLARSKY, Julia (Hrsg.) ; WILLCOCKS, Leslie P. (Hrsg.): *Global Sourcing of Digital Services: Micro and Macro Perspectives*, Springer International Publishing, 2017, S. 124–137

[APB09] ALLMAN, M. ; PAXSON, V. ; BLANTON, E.: *TCP Congestion Control*. RFC 5681 (Draft Standard), September 2009 (Request for Comments)

[APDM18] AL-DHURAIBI, Y. ; PARAISO, F. ; DJARALLAH, N. ; MERLE, P.: Elasticity in Cloud Computing: State of the Art and Research Challenges. In: *IEEE Transactions on Services Computing* 11 (2018), März, Nr. 2, S. 430–447

[BBF+97] BRAY, Michael ; BRUNE, Kimberly ; FISHER, David A. ; FOREMAN, John ; GERKEN, Mark ; GROSS, Jon ; HAINES, Gary ; KEAN, Elizabeth ; LUGINBUHL, Maj D. ; MILLS, William ; ROSENSTEIN, Robert ; SADOSKI, Darleen ; SHIMP, James ; DOREN, Edmond V. ; VONDRAK, Cory: *C4 Software Technology Reference Guide – A Prototype*. Januar 1997

[BCC+17] *Kapitel* Serverless Computing: Current Trends and Open Problems. In: BALDINI, Ioana ; CASTRO, Paul ; CHANG, Kerry ; CHENG, Perry ; FINK, Stephen ; ISHAKIAN, Vatche ; MITCHELL, Nick ; MUTHUSAMY, Vinod ; RABBAH, Rodric ; SLOMINSKI, Aleksander ; SUTER, Philippe: *Research Advances in Cloud Computing*. Springer Singapore, 2017, S. 1–20

[BCD+07] BRITTENHAM, P. ; CUTLIP, R. R. ; DRAPER, C. ; MILLER, B.A. ; CHOUDHARY, S. ; PERAZOLO, M.: IT Service Management Architecture and Autonomic Computing. In: *IBM Systems Journal* 46 (2007), Nr. 3, S. 565–581

[BDR+12] BUGNION, Edouard ; DEVINE, Scott ; ROSENBLUM, Mendel ; SUGERMAN, Jeremy ; WANG, Edward Y.: Bringing Virtualization to the x86 Architecture with the Original VMware Workstation. In: *ACM Transactions on Computer Systems (TOCS)* 30 (2012), November, Nr. 4, S. 12:1–12:51

[Bel66] BELADY, Laszlo A.: A Study of Replacement Algorithms for a Virtual-Storage Computer. In: *IBM Systems Journal* 5 (1966), Nr. 2, S. 78–101

[BFH84] BARBERA, Anthony J. ; FITZGERALD, M. L. ; HAYNES, Leonard S.: RCS: The NBS Real-Time Control System. In: *Robots 8: Conference Proceedings* Bd. 2. Detroit, Michigan, USA : Society of Manufacturing Engineers, June 1984, S. 19–1–19–33

[BFL+13] BRINKMANN, André ; FIEHE, Christoph ; LITVINA, Anna ; LÜCK, Ingo ; NAGEL, Lars ; NARAYANAN, Krishnaprasad ; OSTERMAIR, Florian ; THRONICKE, Wolfgang: Scalable Monitoring System for Clouds. In: *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*. Washington, DC, USA : IEEE Computer Society, 2013 (UCC '13), S. 351–356

[BG73] BUZEN, J. P. ; GAGLIARDI, U. O.: The Evolution of Virtual Machine Architecture. In: *Proceedings of the International Workshop on Managing Requirements Knowledge*, ACM, 1973 (AFIPS '73), S. 291–299

[BGP06] BARESI, Luciano ; GUINEA, Sam ; PLEBANI, Pierluigi: WS-Policy for Service Monitoring. In: BUSSLER, Christoph (Hrsg.) ; SHAN, Ming-Chien (Hrsg.): *Technologies for E-Services*, Springer Berlin Heidelberg, 2006, S. 72–83

[BHRL10] BÖHM, Markus ; HERZOG, Andreas ; RIEDL, Christoph ; LEIMEISTER, Helmut Stefanie und K. Stefanie und Krcmar: Cloud Computing als Treiber der IT-Industrialisierung? Ein Vergleich mit der Automobilbranche. In: *Information Management & Consulting* 25 (2010), Nr. 4, S. 46–54

[Bic18] BICKEL, Horst: *Informationsblatt 1: Die Häufigkeit von Demenzerkrankungen*. Juni 2018

[BKC+18] BEATY, Roger E. ; KENETT, Yoed N. ; CHRISTENSEN, Alexander P. ; ROSENBERG, Monica D. ; BENEDEK, Mathias ; CHEN, Qunlin ; FINK, Andreas ; QIU, Jiang ; KWAPIL, Thomas R. ; KANE, Michael J. u. a.: Robust prediction of individual creative ability from brain functional connectivity. In: *Proceedings of the National Academy of Sciences* 115 (2018), Nr. 5, S. 1087–1092

[BKL09] BAUN, Christian ; KUNZE, Marcel ; LUDWIG, Thomas: Servervirtualisierung. In: *Informatik-Spektrum* 32 (2009), Juni, Nr. 3, S. 197–205

[Bla94] BLACK, Uyless D.: *Network Management Standards: SNMP, CMIP, TMN, MIBs and Object Libraries*. 2nd. McGraw-Hill, Inc., 1994

[BLMR04] BANDARA, A. K. ; LUPU, E. C. ; MOFFETT, J. ; RUSSO, A.: A Goal-based Approach to Policy Refinement. In: *IEEE Fifth International Workshop on Policies for Distributed Systems and Networks (POLICY 2004)*, 2004, S. 229–239

[BM05] BAUKE, H. ; MERTENS, S.: *Cluster Computing: Praktische Einführung in das Hochleistungsrechnen auf Linux-Clustern*. Springer Berlin Heidelberg, 2005 (X. systems. press Series)

[Boe88] BOEHM, Barry W.: A Spiral Model of Software Development and Enhancement. In: *Computer* 21 (1988), May, Nr. 5, S. 61–72

[BPG17] BUCEK, Pavel ; PERICAS-GEERTSEN, Santiago: *JSR-370: Java API for RESTful Web Services (JAX-RS), Version 2.1*. Juli 2017

[BSI08] BSI: *SOA-Security-Kompendium: Sicherheit in Service-orientierten Architekturen*. 2008

[BSI13] BSI: *Hochverfügbarkeitskompendium (Band B, Kapitel 10: Überwachung)*. 2013

[BSI14] BSI: *Zertifizierung nach ISO 27001 auf der Basis von IT-Grundschutz*. 2014

[BSK+15] BURKERT, Malte ; STEWING, Franz-Josef ; KRUMM, Heiko ; FIEHE, Christoph ; LÜCK, Ingo: Networked Devices Meeting Dependability while Supporting the Medical Supervision of Cardiac Patients under Rehabilitation. In: *IFAC-PapersOnLine* 48 (2015), Nr. 4, S. 424–429. – 13th IFAC and IEEE Conference on Programmable Devices and Embedded Systems

[BVKF17] BURKERT, Malte ; VOLMER, Joe ; KRUMM, Heiko ; FIEHE, Christoph: Rule-based technical management for the dependable operation of networked building automation systems. In: *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2017, S. 612–615

[BWL99] BRIAND, L. C. ; WUST, J. ; LOUNIS, H.: Using Coupling Measurement for Impact Analysis in Object-oriented Systems. In: *Proceedings IEEE International Conference on Software Maintenance (ICSM)*, 1999, S. 475–482

[CDKB11] COULOURIS, George ; DOLLIMORE, Jean ; KINDBERG, Tim ; BLAIR, Gordon: *Distributed Systems: Concepts and Design*. 5. Addison-Wesley Publishing Company, 2011

[CFSD89] CASE, J. ; FEDOR, M. ; SCHOFFSTALL, M. ; DAVIN, C.: *RFC 1067: A Simple Network Management Protocol (SNMP)*. April 1989

[CG17] CHANDGUDE, C. P. ; GADEKAR, G. B.: Core of Cloud Computing. In: *International Journal of Engineering Research and Application* 7 (2017), April, Nr. 4, S. 76–81

[CGK+11] CALINESCU, Radu ; GRUNSKE, Lars ; KWIATKOWSKA, Marta ; MIRANDOLA, Raffaela ; TAMBURRELLI, Giordano: Dynamic QoS management and optimization in service-based systems. In: *IEEE Transactions on Software Engineering* 37 (2011), Nr. 3, S. 387–409

[CL07] CULLMAN, Xavier-Noël ; LAMBERTZ, Klaus: *Komplexität und Qualität von Software*. 2007

[CLG+09] CHENG, Betty H. ; LEMOS, Rogério ; GIESE, Holger ; INVERARDI, Paola ; MAGEE, Jeff ; ANDERSSON, Jesper ; BECKER, Basil ; BENCOMO, Nelly ; BRUN, Yuriy ; CUKIC, Bojan ; MARZO SERUGENDO, Giovanna ; DUSTDAR, Schahram ; FINKELSTEIN, Anthony ; GACEK, Cristina ; GEIHS, Kurt ; GRASSI, Vincenzo ; KARSAI, Gabor ; KIENLE, Holger M. ; KRAMER, Jeff ; LITOIU, Marin ; MALEK, Sam ; MIRANDOLA, Raffaela ; MÜLLER, Hausi A. ; PARK, Sooyong ; SHAW, Mary ; TICHY, Matthias ; TIVOLI, Massimo ; WEYNS, Danny ; WHITTLE, Jon: Software Engineering for Self-Adaptive Systems: A Research Roadmap. In: CHENG, Betty H. (Hrsg.) ; LEMOS, Rogério (Hrsg.) ; GIESE, Holger (Hrsg.) ; INVERARDI, Paola (Hrsg.) ; MAGEE, Jeff (Hrsg.): *Software Engineering for Self-Adaptive Systems* Bd. 5525. Springer-Verlag, 2009, S. 1–26

[CLO95] COLEMAN, Don ; LOWTHER, Bruce ; OMAN, Paul: The Application of Software Maintainability Models in Industrial Software Systems. In: *Journal of Systems and Software* 29 (1995), Nr. 1, S. 3–16

[CNYM00] CHUNG, Lawrence ; NIXON, Brian A. ; YU, Eric ; MYLOPOULOS, John: *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000

[CTVP10] CELESTI, A. ; TUSA, F. ; VILLARI, M. ; PULIAFITO, A.: How to Enhance Cloud Architectures to Enable Cross-Federation. In: *IEEE 3rd International Conference on Cloud Computing*, 2010, S. 337–345

[Cus08] CUSUMANO, M. A.: The Changing Software Business: Moving from Products to Services. In: *Computer* 41 (2008), Januar, Nr. 1, S. 20–27

[Dam02] DAMIANOU, Nicodemos C.: *A Policy Framework for Management of Distributed Systems*, Universität London, Diss., Februar 2002

[DBC+00] DURHAM, David ; BOYLE, Jim ; COHEN, Ron ; HERZOG, Shai ; RAJAN, Raju ; SASTRY, Arun: *RFC 2748: The COPS (Common Open Policy Service) Protocol*. 2000

[DBS+09] DICKERSON, Bradford C. ; BAKKOUR, Akram ; SALAT, David H. ; FECZKO, Eric ; PACHECO, Jenni ; GREVE, Douglas N. ; GRODSTEIN, Fran ; WRIGHT, Christopher I. ; BLACKER, Deborah ; ROSAS, H. D. ; SPERLING, Reisa A. ; ATRI, Alireza ; GROWDON, John H. ; HYMAN, Bradley T. ; MORRIS, John C. ; FISCHL, Bruce ; BUCKNER, Randy L.: The Cortical Signature of Alzheimer's Disease: Regionally Specific Cortical Thinning Relates to Symptom Severity in Very Mild to Mild AD Dementia

and is Detectable in Asymptomatic Amyloid-Positive Individuals. In: *Cerebral Cortex* 19 (2009), Nr. 3, S. 497–510

[DC01] DOWLING, Jim ; CAHILL, Vinny: The K-Component Architecture Meta-model for Self-Adaptive Software. In: *Proceedings of the Third International Conference on Metalevel Architectures and Separation of Crosscutting Concerns*. London, UK : Springer-Verlag, 2001 (Reflection 2001). – ISBN 3–540–42618–3, S. 81–88

[DDF+06] DOBSON, Simon ; DENAZIS, Spyros ; FERNÁNDEZ, Antonio ; GAÏTI, Dominique ; GELENBE, Erol ; MASSACCI, Fabio ; NIXON, Paddy ; SAFFRE, Fabrice ; SCHMIDT, Nikita ; ZAMBONELLI, Franco: A Survey of Autonomic Communications. In: *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 1 (2006), Dezember, Nr. 2, S. 223–259

[Deb05] DEBUSMANN, Markus: *Modellbasiertes Service Level Management verteilter Anwendungssysteme*, Universität Kassel, Diss., June 2005

[DFZ+15] DUAN, Y. ; FU, G. ; ZHOU, N. ; SUN, X. ; NARENDRA, N. C. ; HU, B.: Everything as a Service (XaaS) on the Cloud: Origins, Current and Future Trends. In: *IEEE 8th International Conference on Cloud Computing (CLOUD)*, 2015, S. 621–628

[DGG95] DITTRICH, Klaus R. ; GATZIU, Stella ; GEPPERT, Andreas: The Active Database Management System Manifesto: A Rulebase of ADBMS Features. In: SELLIS, Timos (Hrsg.): *Rules in Database Systems* Bd. 985. Springer Berlin Heidelberg, 1995, S. 1–17

[DH08] DINGER, J. ; HARTENSTEIN, H.: *Netzwerk- und IT-Sicherheitsmanagement: eine Einführung*. Univ.-Verlag Karlsruhe, 2008

[DKK+10a] DOHNDORF, O. ; KRUGER, J. ; KRUMM, H. ; FIEHE, C. ; LITVINA, A. ; LUCK, I. ; STEWING, F.: Policy-Based Management for Resource-Constrained Devices and Systems. In: *IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY)*, 2010, S. 61–64

[DKK+10b] DOHNDORF, Oliver ; KRÜGER, Jan ; KRUMM, Heiko ; FIEHE, Christoph ; LITVINA, Anna ; LÜCK, Ingo ; STEWING, Franz-Josef: Lightweight Policy-Based Management of Quality-Assured, Device-Based Service Systems. In: *IEEE 24th International Conference on Advanced Information Networking and Applications (AINA)*, 2010, S. 526–531

[DKK+11a] DOHNDORF, O. ; KRÜGER, J. ; KRUMM, H. ; FIEHE, C. ; LITVINA, A. ; LÜCK, I. ; STEWING, F.: Adaptive and Reliable Binding in Ambient Service Systems. In: *IEEE 16th Conference on Emerging Technologies & Factory Automation (ETFA)*, 2011, S. 1–8

[DKK+11b] DOHNDORF, O. ; KRUGER, J. ; KRUMM, H. ; FIEHE, C. ; LITVINA, A. ; LÜCK, I. ; STEWING, F.: Tool-Supported Refinement of High-Level Requirements and Constraints Into Low-Level Policies. In: *IEEE International Symposium on Policies for Distributed Systems and Networks*, 2011, S. 97–104

[DL03] DAVID, Pierre-Charles ; LEDOUX, Thomas: Towards a Framework for Self-adaptive Component-Based Applications. In: STEFANI, Jean-Bernard (Hrsg.) ; DEMEURE, Isabelle (Hrsg.) ; HAGIMONT, Daniel (Hrsg.): *Distributed Applications and Interoperable Systems* Bd. 2893. Springer Berlin Heidelberg, 2003, S. 1–14

[DLSD01] DULAY, N. ; LUPU, E. ; SLOMAN, M. ; DAMIANOU, N.: A Policy Deployment Model for the Ponder Language. In: *IEEE/IFIP International Symposium on Integrated Network Management Proceedings. Integrated Network Management VII. Integrated Management Strategies for the New Millennium*, 2001, S. 529–543

[DMT07] DMTF: *CIM Query Language Specification, Version: 1.0.0*. August 2007

[DMT10] DMTF: *Web Services for Management (WS-Management) Specification, Version: 1.1.0*. März 2010

[DMT13] DMTF: *CIM Operations over HTTP, Version: 1.4.0*. August 2013

[DMT14a] DMTF: *Open Virtualization Format*. White Paper, April 2014

[DMT14b] DMTF: *Representation of CIM in XML, Version: 2.4.0*. Januar 2014

[DMT14c] DMTF: *WBEM Discovery Using the Service Location Protocol (SLP), Version: 1.0.1*. Januar 2014

[DMT16] DMTF: *Cloud Infrastructure Management Interface (CIMI) Model and RESTful HTTP-based Protocol: An Interface for Managing Cloud Infrastructure*. Juli 2016

[DMT17] DMTF: *Common Information Model (CIM) Specification, Version: 2.50.0*. Januar 2017

[DS14] DENECKEN, Sven ; SCHULZE, Bert: Von der Unternehmenskonsolidierung zu Innovation und Flexibilität. In: *HMD Praxis der Wirtschaftsinformatik* 51 (2014), Dezember, Nr. 6, S. 775–781

[DTSL08] DULAY, N. ; TWIDLE, K. ; SLOMAN, M. ; LUPU, E.: Ponder2 – A Policy Environment for Autonomous Pervasive Systems. In: *IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, 2008, S. 245–246

[DWC10] DILLON, T. ; WU, C. ; CHANG, E.: Cloud Computing: Issues and Challenges. In: *24th IEEE International Conference on Advanced Information Networking and Applications*, 2010, S. 27–33

[Ech90] ECHTLE, K.: *Fehlertoleranzverfahren*. Springer Berlin Heidelberg, 1990 (Studienreihe Informatik)

[Eck09] ECKERT, Claudia: *IT-Sicherheit - Konzepte, Verfahren, Protokolle (6. Aufl.)*. Oldenbourg, 2009

[ED96] EBERT, C. ; DUMKE, R.: *Software-Metriken in der Praxis: Einführung und Anwendung von Software-Metriken in der industriellen Praxis*. Springer Berlin Heidelberg, 1996

[EDM12] ESSOH, Alex D. ; DOUBRAVA, Clemens ; MÜNCH, Isabel: *Sicherheitsempfehlungen für Cloud-Computing-Anbieter*. 2012

[EHS+11] EVERS, Christoph ; HOFFMANN, Axel ; SAUR, Daniel ; GEIHS, Kurt ; LEIMEISTER, Jan M.: Ableitung von Anforderungen zum Adaptionsverhalten in ubiquitären adaptiven Anwendungen. In: *Electronic Communications of the EASST* 37 (2011)

[EMPR12] EDMONDS, A. ; METSCH, T. ; PAPASPYROU, A. ; RICHARDSON, A.: Toward an Open Cloud Standard. In: *IEEE Internet Computing* 16 (2012), Juli, Nr. 4, S. 15–25

[End94] ENDLER, Markus: A Language for Implementing Generic Dynamic Reconfigurations of Distributed Programs. In: *12th Brazilian Symposium on Computer Networks (SBRC 12)*, 1994, S. 175–187

[Erl05] ERL, Thomas: *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall Professional Technical Reference, 2005

[FD00] FISCHL, Bruce ; DALE, Anders M.: Measuring the Thickness of the Human Cerebral Cortex from Magnetic Resonance Images. In: *Proceedings of the National Academy of Sciences* 97 (2000), Nr. 20, S. 11050–11055

[FGJ+78] FLYNN, Michael J. (Hrsg.) ; GRAY, Jim (Hrsg.) ; JONES, Anita K. (Hrsg.) ; LAGALLY, Klaus (Hrsg.) ; OPDERBECK, Holger (Hrsg.) ; POPEK, Gerald J. (Hrsg.) ; RANDELL, Brian (Hrsg.) ; SALTZER, Jerome H. (Hrsg.) ; WIEHLE, Hans-Rüdiger (Hrsg.): *Operating Systems – An Advanced Course*. Springer-Verlag, 1978

[FHS+06] FLOCH, Jacqueline ; HALLSTEINSEN, Svein ; STAV, Erlend ; ELIASSEN, Frank ; LUND, Ketil ; GJORVEN, Eli: Using Architecture Models for Runtime Adaptability. In: *IEEE Software* 23 (2006), March, Nr. 2, S. 62–70

[Fie00] FIELDING, Roy T.: *Architectural Styles and the Design of Network-Based Software Architectures*, University of California, Diss., 2000

[Fis12] FISCHL, Bruce: FreeSurfer. In: *Neuroimage* 62 (2012), Nr. 2, S. 774–781

[FK92] FERRAIOLO, David ; KUHN, Richard: Role-Based Access Control. In: *15th NIST-NCSC National Computer Security Conference*, 1992, S. 554–563

[FK03] FOSTER, Ian ; KESSELMAN, Carl: *The Grid 2: Blueprint for a New Computing Infrastructure*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2003

[FLL+09] FIEHE, Christoph ; LITVINA, Anna ; LÜCK, Ingo ; STEWING, Franz-Josef ; DOHNDORF, Oliver ; KRÜGER, Jan ; KRUMM, Heiko: Policy-gesteuertes Management adaptiver und gütegesicherter Dienstesysteme im Projekt OSAMI. In: *Informatik 2009: Im Focus das Leben, Beiträge der 39. Jahrestagung der Gesellschaft für Informatik e.V. (GI)* (2009), S. 970–983

[FLT+15] FIEHE, C. ; LITVINA, A. ; TONN, J. ; WU, J. ; SCHEEL, M. ; BRINKMANN, A. ; NAGEL, L. ; NARAYANAN, K. ; ZOTH, C. ; GOLTZ, H. ; UNGER, S. ; THRONICKE, W. ; PURSCHE, F.: Building a Medical Research Cloud in the EASI-CLOUDS Project. In: *Concurrency and Computation: Practice and Experience* 27 (2015), November, Nr. 16, S. 4465–4477

[FSS+15] FINN, Emily S. ; SHEN, Xilin ; SCHEINOST, Dustin ; ROSENBERG, Monica D. ; HUANG, Jessica S. ; CHUN, Marvin M. ; PAPADEMETRIS, Xenophon ; CONSTABLE,

R. T.: Functional connectome fingerprinting: Identifying individuals based on patterns of brain connectivity. In: *Nature Neuroscience* 18 (2015), November, Nr. 11, S. 1664—-1671

[GBE+09] GEIHS, Kurt ; BARONE, Paolo ; ELIASSEN, Frank ; FLOCH, Jacqueline ; FRICKE, Rolf ; GJORVEN, Eli ; HALLSTEINSEN, Svein ; HORN, Geir ; KHAN, Mohammad U. ; MAMELLI, Alessandro u. a.: A comprehensive solution for application-level adaptation. In: *Software: Practice and Experience* 39 (2009), Nr. 4, S. 385–422

[Gei08] GEIHS, Kurt: Selbst-Adaptive Software. In: *Informatik-Spektrum* 31 (2008), April, Nr. 2, S. 133–145

[GHGL12] GHIJSEN, M. ; HAM, J. van d. ; GROSSO, P. ; LAAT, C. de: Towards an Infrastructure Description Language for Modeling Computing Infrastructures. In: *IEEE 10th International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, 2012, S. 207–214

[GHH+01] GARSCHHAMMER, M. ; HAUCK, R. ; HEGERING, H. G. ; KEMPTER, B. ; RADISIC, I. ; ROLLE, H. ; SCHMIDT, H. ; HEGERING, H. G. ; LANGER, M. ; NERB, M.: Towards generic Service Management Concepts – A Service Model Based Approach. In: *IEEE/IFIP Seventh International Symposium on Integrated Management (IM 2001)*, 2001, S. 719–732

[GHJV94] GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John M.: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994

[GHK+01] GARSCHHAMMER, Markus ; HAUCK, Rainer ; KEMPTER, Bernhard ; RADISIC, Igor ; ROELLE, Harald ; SCHMIDT, Holger ; SCHMIDT, H.: The MNM Service Model – Refined Views on Generic Service Management. In: *Journal of Communications and Networks* 3 (2001)

[GLP75] GRAY, J. N. ; LORIE, R. A. ; PUTZOLU, G. R.: Granularity of Locks in a Shared Data Base. In: *Proceedings of the 1st International Conference on Very Large Data Bases*. New York, NY, USA : ACM, 1975 (VLDB '75), S. 428–451

[GLPT94] GRAY, J. N. ; LORIE, R. A. ; PUTZOLU, G. R. ; TRAIGER, I. L.: Granularity of Locks and Degrees of Consistency in a Shared Data Base. In: STONEBRAKER, Michael (Hrsg.): *Readings in Database Systems*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1994, S. 181–208

[GM99] GUPTA, Ashish ; MUMICK, Inderpal S.: Maintenance of Materialized Views: Problems, Techniques, and Applications. In: *Materialized Views: Techniques, Implementations, and Applications*. MIT Press, 1999, S. 145–157

[GMW00] GARLAN, David ; MONROE, Robert T. ; WILE, David: Acme: Architectural Description of Component-Based Systems. In: LEAVENS, Gary T. (Hrsg.) ; SITARAMAN, Murali (Hrsg.): *Foundations of Component-Based Systems*. Cambridge University Press, 2000, S. 47–67

[Gol73] GOLDBERG, R. P.: Architecture of Virtual Machines. In: *Proceedings of the International Workshop on Managing Requirements Knowledge*, ACM, 1973 (AFIPS '73), S. 309–318

[GS90] GORA, Walter ; SPEYERER, Reinhard: *Abstract Syntax Notation One (ASN.1)*. 2nd. DATACOM, 1990

[Ham05] HAMMERSCHALL, Ulrike: *Verteilte Systeme und Anwendungen - Architekturkonzepte, Standards und Middleware-Technologien*. Pearson Education, 2005. – ISBN 978–3–8273–7096–9

[HAN99] HEGERING, Heinz-Gerd ; ABECK, Sebastian ; NEUMAIR, Bernhard: *Integriertes Management vernetzter Systeme: Konzepte, Architekturen und deren betrieblicher Einsatz*. dpunkt-Verlag, 1999

[HB12] HILLBRECHT, R. ; BONA, L. C. E.: A SNMP-Based Virtual Machines Management Interface. In: *IEEE Fifth International Conference on Utility and Cloud Computing*, 2012, S. 279–286

[HBV17] HEILIG, Leonard ; BUYYA, Rajkumar ; VOSS, Stefan: Location-Aware Brokering for Consumers in Multi-Cloud Computing Environments. In: *Journal of Network and Computer Applications* 95 (2017), S. 79–93

[HGB10] HEBIG, Regina ; GIESE, Holger ; BECKER, Basil: Making Control Loops Explicit when Architecting Self-adaptive Systems. In: *Proceedings of the Second International Workshop on Self-organizing Architectures*, ACM, 2010 (SOAR '10), S. 21–28

[HJS17] HILLE, Maximilian ; JANATA, Steve ; SCHWALM, Anna-Lena: *Hybrid- & Multi-Cloud-Services im deutschen Mittelstand*. August 2017. – Empirische Studie im Auftrag der QSC AG

[HK07] HARGRAVE, B. J. ; KRIENS, Peter: OSGi Best Practices! In: *OSGi Alliance Community Event* (2007), S. 26–27

[HN15] HAYNES, Tom ; NOVECK, Dave: *RFC 7530 - Network File System (NFS) Version 4 Protocol*. 2015

[Hob04] HOBBS, C.: *A Practical Approach to WBEM/CIM Management*. CRC Press, 2004

[How96] HOWES, T.: *RFC 1960: A String Representation of LDAP Search Filters*. Juni 1996

[HS05] HUHNS, Michael N. ; SINGH, Munindar P.: Service-Oriented Computing: Key Concepts and Principles. In: *IEEE Internet Computing* 9 (2005), Nr. 1, S. 75–81

[HTMP02] HANSMANN, Uwe ; THOMPSON, Peter ; METTALA, Riku M. ; PURAKAYASTHA, Apratim: *SyncML: Synchronizing Your Mobile Data*. Prentice Hall Professional Technical Reference, 2002

[IBM06] IBM CORPORATION: *An Architectural Blueprint for Autonomic Computing*. White Paper, June 2006

[ISO89] ISO: *ISO/IEC 7498-4: Information Processing Systems – Open Systems Interconnection – Basic Reference Model – Part 4: Management Framework*. 1989

[ISO13] ISO: *ISO 3166-2: Codes for the representation of names of countries and their subdivisions – Part 2: Country subdivision code*. 2013

[ISO15] ISO: *Qualitätsmanagementsysteme – Anforderungen (ISO 9001:2015)*. 2015

[Jel03] JELLINGER, Kurt: Demenzen, Morbus Alzheimer und Morbus Parkinson beim alternden Mann. In: *Blickpunkt der Mann* 1 (2003), Nr. 3, S. 34–42

[JOP+18] JONES, Eric ; OLIPHANT, Travis ; PETERSON, Pearu u. a.: *SciPy: Open source scientific tools for Python*. `http://www.scipy.org/`, 2018

[Kö18] KÖHLER, Helmut: *Bürgerliches Gesetzbuch - mit BGB-Informationspflichten-Verordnung, Allgemeinem Gleichbehandlungsgesetz, Produkthaftungsgesetz, Unterlassungsklagengesetz, Wohnungseigentumsgesetz, Beurkundungsgesetz, Lebenspartnerschaftsgesetz und Erbbaurechtsgesetz*. dtv Verlagsgesellschaft, 2018

[Kan14] KAN, Stephen H.: *Metrics and Models in Software Quality Engineering*. 2nd. Addison-Wesley Professional, 2014

[KBC02] KETFI, Abdelmadjid ; BELKHATIR, Noureddine ; CUNIN, Pierre-Yves: Automatic Adaptation of Component-based Software: Issues and Experiences. In: ARABNIA, Hamid R. (Hrsg.): *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*. Las Vegas, Nevada, USA : CSREA Press, 2002 (PDPTA), S. 1365–1371

[Kir06] KIRCHER, Herbert (Hrsg.): *IT: Technologien, Lösungen, Innovationen*. Berlin : Springer-Verlag, 2006

[KKK96] KOCH, Thomas ; KRELL, Christoph ; KRAEMER, Bernd: Policy Definition Language for Automated Management of Distributed Systems. In: *Proceedings of the 2nd IEEE International Workshop on Systems Management (SMW'96)*, IEEE Computer Society, 1996 (SMW '96), S. 55–64

[KKS01] KELLER, Alexander ; KREGER, Heather ; SCHOPMEYER, Karl: Towards a CIM Schema for RunTime Application Management. In: *12th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM)*, 2001, S. 217–230

[Kla04] KLAMAR, Sebastian: *Adaptive Architekturen für verteilte Softwaresysteme*, Technische Universität Dresden, Fakultät Informatik,Institut für Systemarchitektur, Lehrstuhl Rechnernetze, Diplomarbeit, September 2004

[Koc96] KOCH, Thomas: *Automated Management of Distributed Systems*, Fernuniversität Hagen, Diss., Dezember 1996

[KTMF09] KEAHEY, K. ; TSUGAWA, M. ; MATSUNAGA, A. ; FORTES, J.: Sky Computing. In: *IEEE Internet Computing* 13 (2009), September, Nr. 5, S. 43–51

[Lad01] LADDAGA, Robert: Active Software. In: ROBERTSON, Paul (Hrsg.) ; SHROBE, Howie (Hrsg.) ; LADDAGA, Robert (Hrsg.): *Self-Adaptive Software* Bd. 1936. Springer Berlin Heidelberg, 2001, S. 11–26

[Lan01] LANGER, Michael: *Konzeption und Anwendung einer Customer Service Management Architektur*, LMU München: Fakultät für Mathematik, Informatik und Statistik, Diss., 2001

[Lew13] LEWIS, G. A.: Role of Standards in Cloud-Computing Interoperability. In: *46th Hawaii International Conference on System Sciences*, IEEE Computer Society, Januar 2013, S. 1652–1661

[Lil08] LILIENTHAL, Carola: *Komplexität von Softwarearchitekturen – Stile und Strategien*, Universität Hamburg , Fachbereich Informatik, Arbeitsbereich Softwaretechnik, Diss., Juli 2008

[Liu11] LIU, Lei: *Organic Service-Level Management in Service-Oriented Environments*. Karlsruhe, KIT, Fakultät für Wirtschaftswissenschaften, Diss., 2011

[LKBT11] LOUTAS, N. ; KAMATERI, E. ; BOSI, F. ; TARABANIS, K.: Cloud Computing Interoperability: The State of Play. In: *IEEE Third International Conference on Cloud Computing Technology and Science*, 2011, S. 752–757

[LKN+09] LENK, Alexander ; KLEMS, Markus ; NIMIS, Jens ; TAI, Stefan ; SANDHOLM, Thomas: What's Inside the Cloud? An Architectural Map of the Cloud Landscape. In: *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, IEEE Computer Society, 2009 (CLOUD '09), S. 23–31

[LLXH18] LIU, Jin ; LIAO, Xuhong ; XIA, Mingrui ; HE, Yong: Chronnectome fingerprinting: identifying individuals and predicting higher cognitive functions using dynamic brain connectivity patterns. In: *Human brain mapping* 39 (2018), Nr. 2, S. 902–915

[Lüc06] LÜCK, Ingo: *Modellbasierte Konfiguration von Sicherheitsdiensten*, Universität Dortmund, Diss., 2006

[LYMZ11] LEE, B. S. ; YAN, S. ; MA, D. ; ZHAO, G.: Aggregating IaaS Service. In: *2011 Annual SRII Global Conference*, 2011, S. 335–338

[Maj10] MAJER, Frederic: *Semantisches Informationsmodell für die Betriebsunterstützung dienstorientierter Systeme*. Karlsruhe, KIT, Fakultät für Informatik, Diss., 2010

[Mar97] MARRIOTT, Damian A.: *Policy Service for Distributed Systems*, Department of Computing, Imperial College, Diss., Juni 1997

[MB17] MCGRATH, G. ; BRENNER, P. R.: Serverless Computing: Design, Implementation, and Performance. In: *IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, 2017, S. 405–410

[MC93] MAULLO, M. J. ; CALO, S. B.: Policy Management: An Architecture and Approach. In: *IEEE 1st International Workshop on Systems Management*, 1993, S. 13–26

[McC04] MCCONNELL, Steve: *Code Complete: A Practical Handbook of Software Construction*. 2. Microsoft Press, 2004

[Mel10] MELZER, Ingo: *Service-orientierte Architekturen mit Web Services: Konzepte – Standards – Praxis*. Spektrum Akademischer Verlag, 2010

[Men00] MENG, Alex C.: On Evaluating Self-Adaptive Software. In: *Proceedings of the First International Workshop on Self-Adaptive software*, Springer-Verlag New York, Inc., 2000 (IWSAS' 2000). – ISBN 3–540–41655–2, S. 65–74

[MESW01] MOORE, Bob ; ELLESSON, Ed ; STRASSNER, John ; WESTERINEN, Andrea: *Policy Core Information Model – Version 1 Specification*. 2001

[MG11] MELL, Peter ; GRANCE, Timothy: *The NIST Definition of Cloud Computing*. 26. January 2011

[Mic17] MICROSOFT CORPORATION: *Common Internet File System (CIFS) Protocol*. Juni 2017

[MLM+06] MACKENZIE, C. M. ; LASKEY, Ken ; MCCABE, Francis ; BROWN, Peter F. ; METZ, Rebekah: *Reference Model for Service Oriented Architecture 1.0*. August 2006

[MLO86] MOHAN, C. ; LINDSAY, B. ; OBERMARCK, R.: Transaction Management in the R* Distributed Database Management System. In: *ACM Transactions on Database Systems (TODS)* 11 (1986), Dezember, Nr. 4, S. 378–396

[Mot03] MOTUZENKO, Pavel: Adaptive domain model: dealing with multiple attributes of self-managing distributed object systems. In: *Proceedings of the 1st international symposium on Information and communication technologies* Trinity College Dublin, 2003, S. 549–554

[MPS08] MÜLLER, Hausi ; PEZZÈ, Mauro ; SHAW, Mary: Visibility of Control in Adaptive Systems. In: *Proceedings of the 2nd international workshop on Ultra-large-scale software-intensive systems*, ACM, 2008 (ULSSIS '08), S. 23–26

[MS93] MOFFETT, J. D. ; SLOMAN, M. S.: Policy Hierarchies for Distributed Systems Management. In: *IEEE Journal on Selected Areas in Communications* 11 (1993), Dezember, Nr. 9, S. 1404–1414

[NA12] NARENDRA, Kumpati S. ; ANNASWAMY, Anuradha M.: *Stable Adaptive Systems*. Dover Publications, 2012 (Dover Books on Electrical Engineering)

[NCR+18] NETTO, Marco A. S. ; CALHEIROS, Rodrigo N. ; RODRIGUES, Eduardo R. ; CUNHA, Renato L. F. ; BUYYA, Rajkumar: HPC Cloud for Scientific and Business Applications: Taxonomy, Vision, and Research Challenges. In: *ACM Computing Surveys* 51 (2018), Januar, Nr. 1, S. 8:1–8:29

[NSH+17] NEMETH, Evi ; SNYDER, Garth ; HEIN, Trent R. ; WHALEY, Ben ; MACKIN, Dan: *UNIX and Linux System Administration Handbook*. 5. Addison-Wesley Professional, 2017

[OAS06] OASIS: *WSDM 1.1 OASIS Standard Specifications*. August 2006

[OAS13] OASIS: *Topology and Orchestration Specification for Cloud Applications*. November 2013

[OAS14] OASIS: *Cloud Application Management for Platforms*. November 2014

[OGT+99] OREIZY, Peyman ; GORLICK, Michael M. ; TAYLOR, Richard N. ; HEIMBIGNER, Dennis ; JOHNSON, Gregory ; MEDVIDOVIC, Nenad ; QUILICI, Alex ; ROSENBLUM, David S. ; WOLF, Alexander L.: An Architecture-Based Approach to Self-Adaptive Software. In: *IEEE Intelligent Systems* 14 (1999), May, Nr. 3, S. 54–62

[OMB07] O'BRIEN, Liam ; MERSON, Paulo ; BASS, Len: Quality Attributes for Service-Oriented Architectures. In: *Proceedings of the International Workshop on Systems Development in SOA Environments*, IEEE Computer Society, 2007 (SDSOA '07), S. 1–7

[Ope16a] OPEN GRID FORUM: *Open Cloud Computing Interface*. September 2016

[Ope16b] OPEN MOBILE ALLIANCE: *OMA Device Management Tree and Description Serialization, Version 1.3*. Mai 2016

[Ope16c] OPEN MOBILE ALLIANCE: *OMA Device Management Tree and Description, Version 1.3*. Mai 2016

[OSG14] OSGI ALLIANCE: *OSGi Core Release 6 Specification*. `https://osgi.org/download/r6/osgi.core-6.0.0.pdf`, Juni 2014

[OSG15] OSGI ALLIANCE: *OSGi Compendium Release 6 Specification*. `https://osgi.org/download/r6/osgi.cmpn-6.0.0.pdf`, Juli 2015

[PCCA13] PINHEIRO, B. ; CHAVES, R. ; CERQUEIRA, E. ; ABELEM, A.: CIM-SDN: A Common Information Model extension for Software-Defined Networking. In: *IEEE Globecom Workshops (GC Wkshps)*, 2013, S. 836–841

[Pet11] PETCU, Dana: Portability and Interoperability between Clouds: Challenges and Case Study. In: ABRAMOWICZ, Witold (Hrsg.) ; LLORENTE, Ignacio M. (Hrsg.) ; SURRIDGE, Mike (Hrsg.) ; ZISMAN, Andrea (Hrsg.) ; VAYSSIÈRE, Julien (Hrsg.): *Towards a Service-Based Internet*, Springer Berlin Heidelberg, 2011, S. 62–74

[PKH+08] POHL, A. ; KRUMM, H. ; HOLLAND, F. ; STEWING, F. ; LÜCK, I.: Service-Orientation and Flexible Service Binding in Distributed Automation and Control Systems. In: *22nd International Conference on Advanced Information Networking and Applications - Workshops (aina workshops 2008)*, 2008, S. 1393–1398

[Poh09] POHLAND, Sven: *Flexibilisierung von Geschäftsprozessen: Konzepte und Praxisbeispiele*. Walter de Gruyter GmbH & Co KG, 2009

[PR98] POSTEL, J. ; REYNOLDS, J. K.: *RFC 2400: Internet Official Protocol Standards*. September 1998

[PR15] PÖTZSCH lga ; RÖSSGER, Felix: *Bevölkerung Deutschlands bis 2060 – 13. koordinierte Bevölkerungsvorausberechnung*. April 2015

[PRS17] PELLEGRINI, R. ; ROTTMANN, P. ; STRIEDER, G.: Preventing vendor lock-ins via an interoperable multi-cloud deployment approach. In: *12th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2017, S. 382–387

[PTDL07] PAPAZOGLOU, Michael P. ; TRAVERSO, Paolo ; DUSTDAR, Schahram ; LEYMANN, Frank: Service-Oriented Computing: State of the Art and Research Challenges. In: *Computer* 40 (2007), November, Nr. 11, S. 38–45. – ISSN 0018–9162

[PW92] PERRY, Dewayne E. ; WOLF, Alexander L.: Foundations for the Study of Software Architecture. In: *SIGSOFT Software Engineering Notes* 17 (1992), October, Nr. 4, S. 40–52

[PXW13] PAHL, Claus ; XIONG, Huanhuan ; WALSHE, Ray: A Comparison of On-Premise to Cloud Migration Approaches. In: LAU, Kung-Kiu (Hrsg.) ; LAMERSDORF, Winfried (Hrsg.) ; PIMENTEL, Ernesto (Hrsg.): *Service-Oriented and Cloud Computing*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2013, S. 212–226

[Är07] ÄRZTE ZEITUNG: *Kognitive Defizite sind leicht auf M. Alzheimer abzuklären – MRT und Liquordiagnostik liefern deutliche Hinweise*. Oktober 2007

[Rap04] RAPPA, M. A.: The Utility Business Model and the Future of Computing Services. In: *IBM Systems Journal* 43 (2004), Nr. 1, S. 32–42

[RB14] ROSADO, Tiago ; BERNARDINO, Jorge: An Overview of Openstack Architecture. In: *Proceedings of the 18th International Database Engineering & Applications Symposium*, ACM, 2014 (IDEAS '14), S. 366–367

[RC07] RANGANATHAN, Anand ; CAMPBELL, Roy H.: What is the complexity of a distributed computing system? In: *Complexity* 12 (2007), Nr. 6, S. 37–45

[Red12] REDDY, C. H. N.: A CIM (Common Information Model) Based Management Model for Clouds. In: *IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, 2012, S. 1–5

[RJB04] RUMBAUGH, James ; JACOBSON, Ivar ; BOOCH, Grady: *The Unified Modeling Language Reference Manual*. 2nd Edition. Pearson Higher Education, 2004

[RMVG+10] RODERO-MERINO, Luis ; VAQUERO, Luis M. ; GIL, Victor ; GALÁN, Fermín ; FONTÁN, Javier ; MONTERO, Rubén S. ; LLORENTE, Ignacio M.: From Infrastructure Delivery to Dervice Management in Clouds. In: *Future Generation Computer Systems* 26 (2010), Nr. 8, S. 1226–1240

[Ros90] ROSE, M.: *RFC 1158: Management Information Base for Network Management of TCP/IP-based internets: MIB-II*. Mai 1990

[Ros99] ROSENBLUM, Mendel: VMware's Virtual Platform. In: *Proceedings of Hot Chips* Bd. 1999, 1999, S. 185–196

[Rot16] ROTH, Armin (Hrsg.): *Einführung und Umsetzung von Industrie 4.0: Grundlagen, Vorgehensmodell und Use Cases aus der Praxis*. Springer Gabler, 2016

[RR18] *Kapitel* Multi-cloud Brokerage Solutions and Services. In: RAJ, Pethuru ; RAMAN, Anupama: *Software-Defined Cloud Centers: Operational and Management Technologies and Tools*. Springer International Publishing, 2018, S. 155–184

[RRB15] RANI, B. K. ; RANI, B. P. ; BABU, A. V.: Cloud Computing and Inter-Clouds – Types, Topologies and Research Issues. In: *Procedia Computer Science* 50 (2015), S. 24–29

[Sai07] SAILER, Martin: *Konzeption einer Service-MIB – Analyse und Spezifikation dienstorientierter Managementinformation*, LMU München: Fakultät für Mathematik, Informatik und Statistik, Diss., 2007

[Sam02] SAMULOWITZ, Michael: *Kontextadaptive Dienstnutzung in Ubiquitous Computing Umgebungen*, LMU München: Fakultät für Mathematik, Informatik und Statistik, Diss., Juni 2002

[Sat96] SATYANARAYANAN, Mahadev: Mobile Information Access. In: *IEEE Personal Communications* 3 (1996), S. 26–33

[SBD+10] SCHATTEN, Alexander ; BIFFL, Stefan ; DEMOLSKY, Markus ; GOSTISCHA-FRANTA, Erik ; OESTREICHER, Thomas ; WINKLER, Dietmar: *Best Practice Software-Engineering Eine praxiserprobte Zusammenstellung von komponentenorientierten Konzepten, Methoden und Werkzeugen*. Spektrum Akademischer Verlag, 2010

[SBS+04] SALAT, David H. ; BUCKNER, Randy L. ; SNYDER, Abraham Z. ; GREVE, Douglas N. ; DESIKAN, Rahul S. ; BUSA, Evelina ; MORRIS, John C. ; DALE, Anders M. ; FISCHL, Bruce: Thinning of the Cerebral Cortex in Aging. In: *Cerebral Cortex* 14 (2004), Nr. 7, S. 721–730

[Sch07] SCHIFFERS, Michael: *Management dynamischer Virtueller Organisationen in Grids*, LMU München: Fakultät für Mathematik, Informatik und Statistik, Diss., 2007

[Sch08] SCHAAF, Thomas: *IT-gestütztes Service-Level-Management – Anforderungen und Spezifikation einer Managementarchitektur*, LMU München: Fakultät für Mathematik, Informatik und Statistik, Diss., Dezember 2008

[SFK00] SANDHU, Ravi ; FERRAIOLO, David ; KUHN, Richard: The NIST Model for Role-based Access Control: Towards a Unified Standard. In: *Proceedings of the Fifth ACM Workshop on Role-based Access Control*, ACM, 2000 (RBAC '00), S. 47–63

[SK98] SCHILDT, Gerhard-Helge ; KASTNER, Wolfgang: *Prozeßautomatisierung*. Wien : Springer, 1998

[SKI13] STAVRU, Stavros ; KRASTEVA, Iva ; ILIEVA, Sylvia: Challenges for Migrating to the Service Cloud Paradigm: An Agile Perspective. In: HALLER, Armin (Hrsg.) ; HUANG, Guangyan (Hrsg.) ; HUANG, Zhisheng (Hrsg.) ; PAIK, Hye-young (Hrsg.) ; SHENG, Quan Z. (Hrsg.): *Web Information Systems Engineering – WISE 2011 and 2012 Workshops*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2013, S. 77–91

[SL90] SHETH, Amit P. ; LARSON, James A.: Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. In: *ACM Computing Surveys* 22 (1990), September, Nr. 3, S. 183–236

[Slo94] SLOMAN, Morris: Policy Driven Management for Distributed Systems. In: *Journal of Network and Systems Management* 2 (1994), Dezember, Nr. 4, S. 333–360

[Slo95] SLOMAN, Morris: Management Issues for Distributed Services. In: *Proceedings of the 2nd International Workshop on Services in Distributed and Networked Environments* IEEE, 1995 (SDNE '95), S. 52–59

[SP92] SHE, J. ; PECHT, M. G.: Reliability of a k-out-of-n warm-standby system. In: *IEEE Transactions on Reliability* 41 (1992), März, Nr. 1, S. 72–75

[Spr04] SPRINGER, Thomas: *Ein komponentenbasiertes Meta-Modell kontextabhängiger Adaptionsgraphen für mobile und ubiquitäre Anwendungen*, Technischen Universität Dresden, Fakultät Informatik, Diss., Oktober 2004

[SSB10] SNEED, Harry M. ; SEIDL, Richard ; BAUMGARTNER, Manfred: *Software in Zahlen die Vermessung von Applikationen*. München : Hanser, 2010

[SSJ+16] SHRIVASTWA, Alok ; SARAT, Sunil ; JACKSON, Kevin ; BUNCH, Cody ; SIGLER, Egle ; CAMPBELL, Tony: *OpenStack: Building a Cloud Environment*. Packt Publishing, 2016

[ST07] STARKE, Gernot (Hrsg.) ; TILKOV, Stefan (Hrsg.): *SOA-Expertenwissen: Methoden, Konzepte und Praxis serviceorientierter Architekturen*. dpunkt-Verlag, 2007

[ST09] SALEHIE, Mazeiar ; TAHVILDARI, Ladan: Self-Adaptive Software: Landscape and Research Challenges. In: *ACM Transactions on Autonomous and Adaptive Systems* 4 (2009), May, Nr. 2, S. 1–42

[Sta98] STALLINGS, William: *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. 3rd. Addison-Wesley Longman Publishing Co., Inc., 1998

[Swa76] SWANSON, E. B.: The Dimensions of Maintenance. In: *Proceedings of the 2nd International Conference on Software Engineering*. Los Alamitos, CA, USA : IEEE Computer Society Press, 1976 (ICSE '76), S. 492–497

[SWC+07] STEINDER, M. ; WHALLEY, I. ; CARRERA, D. ; GAWEDA, I. ; CHESS, D.: Server virtualization in autonomic management of heterogeneous workloads. In: *10th IFIP/IEEE International Symposium on Integrated Network Management*, 2007, S. 139–148

[TOR+15] THRONICKE, W. ; OSTERMAIR, F. ; RÖHR, F. ; STEWING, F.J. ; LÜCK, I. ; FIEHE, C.: *Abschlussbericht des ITEA 2/BMBF Projektes: EASI-CLOUDS: Erweiterbare Architektur und Service Infrastruktur für Cloud-Computing Software (Laufzeit: 1.12.2011–30.3.2015)*. Atos IT Solutions and Services GmbH, 2015

[TS08] TANENBAUM, Andrew S. ; STEEN, Maarten v.: *Verteilte Systeme, Prinzipien und Paradigmen*. 2., aktualisierte Aufl. Pearson Studium, 2008

[Tuk77] TUKEY, John W.: *Exploratory Data Analysis*. Addison-Wesley, 1977 (Addison-Wesley Series in Behavioral Science)

[VB17] VARGHESE, Blesson ; BUYYA, Rajkumar: Next Generation Cloud Computing: New Trends and Research Directions. In: *CoRR* (2017)

[VBR+12] VILLEGAS, David ; BOBROFF, Norman ; RODERO, Ivan ; DELGADO, Javier ; LIU, Yanbin ; DEVARAKONDA, Aditya ; FONG, Liana ; SADJADI, S. M. ; PARASHAR, Manish: Cloud Federation in a Layered Service Model. In: *Journal of Computer and System Sciences* 78 (2012), Nr. 5, S. 1330–1344

[VCD+13] VERSTEEGEN, G. ; CHUGHTAI, A. ; DÖRNEMANN, H. ; HEINOLD, R. ; HUBERT, R. ; SALOMON, K. ; VOGEL, O.: *Software Management: Beherrschung des Lifecycles*. Springer Berlin Heidelberg, 2013 (Xpert.press)

[Ver00] VERMA, Dinesh C.: *Policy-Based Networking: Architecture and Algorithms*. New Riders Publishing, 2000

[VRMCL08] VAQUERO, Luis M. ; RODERO-MERINO, Luis ; CACERES, Juan ; LINDNER, Maik: A Break in the Clouds: Towards a Cloud Definition. In: *SIGCOMM Computer Communication Review* 39 (2008), December, Nr. 1, S. 50–55

[WF02] WERMELINGER, Michel ; FIADEIRO, José Luiz: A Graph Transformation Approach to Run-Time Software Architecture Reconfiguration. In: *Science of Computer Programming* 44 (2002), August, Nr. 2, S. 133–155

[WHKL08] WÜTHERICH, Gerd ; HARTMANN, Nils ; KOLB, Bernd J. ; LÜBKEN, Matthias: *Die OSGi Service Platform: Eine Einführung mit Eclipse Equinox*. dpunkt.verlag, 2008

[Wie95] WIES, René Félix Jacques: *Policies in Integrated Network and Systems Management: Methodologies for the Definition, Transformation, and Application of Management Policies*, Ludwig-Maximilians-Universität München, Diss., Juni 1995

[Wor07] WORLD WIDE WEB CONSORTIUM (W3C): *Web Services Policy 1.5 – Framework*. September 2007

[WSS+01] WESTERINEN, Andrea ; SCHNIZLEIN, J ; STRASSNER, J ; SCHERLING, M ; QUINN, B ; HERZOG, S ; HUYNH, A ; CARLSON, M ; PERRY, J ; WALDBUSSER, S: *RFC 3198: Terminology for Policy-Based Management*. Januar 2001

[YVCB10] YEO, Chee S. ; VENUGOPAL, Srikumar ; CHU, Xingchen ; BUYYA, Rajkumar: Autonomic Metered Pricing for a Utility Computing Service. In: *Future Generation Computer Systems* 26 (2010), Nr. 8, S. 1368 –1380

[ZCB10] ZHANG, Qi ; CHENG, Lu ; BOUTABA, Raouf: Cloud computing: state-of-the-art and research challenges. In: *Journal of Internet Services and Applications* 1 (2010), Mai, Nr. 1, S. 7–18

Publikationen

2009

FIEHE, Christoph ; LITVINA, Anna ; LÜCK, Ingo ; DOHNDORF, Oliver ; KATTWINKEL, Jens ; STEWING, Franz-Josef ; KRÜGER, Jan ; KRUMM, Heiko: Location-Transparent Integration of Distributed OSGi Frameworks and Web Services. In: *IEEE 23nd International Conference on Advanced Information Networking and Applications (AINA)*, 2009, S. 464–469

FIEHE, Christoph ; LITVINA, Anna ; LÜCK, Ingo ; STEWING, Franz-Josef ; DOHNDORF, Oliver ; KRÜGER, Jan ; KRUMM, Heiko: Policy-gesteuertes Management adaptiver und gütegesicherter Dienstesysteme im Projekt OSAMI. In: *Informatik 2009: Im Focus das Leben, Beiträge der 39. Jahrestagung der Gesellschaft für Informatik e.V. (GI)*, 2009, S. 970-–983

2010

DOHNDORF, Oliver ; KRÜGER, Jan ; KRUMM, Heiko ; FIEHE, Christoph ; LITVINA, Anna ; LÜCK, Ingo ; STEWING, Franz-Josef: Policy-Based Management for Resource-Constrained Devices and Systems. In: *IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY)*, 2010, S. 61—64

DOHNDORF, Oliver ; KRÜGER, Jan ; KRUMM, Heiko ; FIEHE, Christoph ; LITVINA, Anna ; LÜCK, Ingo ; STEWING, Franz-Josef: Towards the Web of Things: Using DPWS to bridge isolated OSGi platforms. In: *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2010, S. 720–725

DOHNDORF, Oliver ; KRÜGER, Jan ; KRUMM, Heiko ; FIEHE, Christoph ; LITVINA, Anna ; LÜCK, Ingo ; STEWING, Franz-Josef: Lightweight Policy-Based Management of Quality-Assured, Device-Based Service Systems. In: *IEEE International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, 2010, S. 526–531

2011

Naci Dai ; Wolfgang Thronicke ; Alejandra Ruiz López ; Félix Cuadrado Latasa ; Elmar Zeeb ; Christoph Fiehe ; Anna Litvina ; Jan Krüger ; Oliver Dohndorf ; Isaac Agudo ; Jesús Bermejo Muñoz: OSAMI Commons - An open dynamic services platform for ambient intelligence: In: IEEE 16th Conference on Emerging Technologies & Factory Automation (ETFA), 2011, S. 1–10

DOHNDORF, Oliver ; KRÜGER, Jan ; KRUMM, Heiko ; FIEHE, Christoph ; LITVINA, Anna ; LÜCK, Ingo ; STEWING, Franz-Josef: Adaptive and Reliable Binding in Ambient Service Systems. In: *IEEE 16th Conference on Emerging Technologies & Factory Automation (ETFA)*, 2011, S. 1–8

DOHNDORF, Oliver ; KRÜGER, Jan ; KRUMM, Heiko ; FIEHE, Christoph ; LITVINA, Anna ; LÜCK, Ingo ; STEWING, Franz-Josef: Tool-Supported Refinement of High-Level Requirements and Constraints Into Low-Level Policies. In: *IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY)*, 2011, S. 97–104

2013

BRINKMANN, André ; FIEHE, Christoph ; LITVINA, Anna ; LÜCK, Ingo ; NAGEL, Lars ; NARAYANAN, Krishnaprasad ; OSTERMAIER, Florian ; THRONICKE, Wolfgang: Scalable Monitoring System for Clouds. In: *IEEE/ACM 6th International Conference on Utility and Cloud Computing (UCC)*, 2013, S. 351–356

2014

FIEHE, Christoph ; LITVINA, Anna ; TONN, Jakob ; WU, Jie ; SCHEEL, Michael ; BRINKMANN, André ; NAGEL, Lars ; NARAYANAN, Krishnaprasad ; ZOTH, Carsten ; GOLTZ, Hans-Joachim ; UNGER, Steffen ; THRONICKE, Wolfgang ; PURSCHE, Fabian: Building a Medical Research Cloud in the EASI-CLOUDS Project. In: *IEEE 6th International Workshop on Science Gateways (IWSG)*, 2014, S. 36–41

2015

THRONICKE, Wolfgang ; OSTERMAIR, Florian ; RÖHR, Florian ; STEWING, Franz-Josef ; LÜCK, Ingo ; FIEHE, Christoph: *Abschlussbericht des ITEA 2/BMBF Projektes: EASI-CLOUDS: Erweiterbare Architektur und Service Infrastruktur für Cloud-Computing Software (Laufzeit: 1.12.2011–30.3.2015)*. Atos IT Solutions and Services GmbH, 2015

BURKERT, Malte ; KRUMM, Heiko ; FIEHE, Christoph: Technical management system for dependable Building Automation Systems, In: *IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, 2015, S. 1–8

WU, Jie ; NARAYANAN, Krishnaprasad ; NAGEL, Lars ; FIEHE, Christoph ; LITVINA, Anna ; TONN, Jakob ; ZOTH, Carsten ; GOLTZ, Hans-Joachim ; UNGER, Steffen ; PURSCHE, Fabian ; SCHEEL, Michael ; BRINKMANN, André ; THRONICKE, Wolfgang: Building a medical research cloud in the EASI-CLOUDS project. In: *Concurrency and Computation: Practice and Experience* 27 (2015), November, Nr. 16, S. 4465–4477

2016

BUTZIN, Björn ; KONIECZEK, Björn ; GOLATOWSKI; Frank ; TIMMERMANN, Dirk ; FIEHE, Christoph: Applying the BaaS reference architecture on different classes of devices. In: *IEEE 2nd International Workshop on Modelling, Analysis, and Control of Complex CPS (CPS Data)*, 2016, S. 1–6

2017

BURKERT, Malte ; VOLMER, Joe ; KRUMM, Heiko ; FIEHE, Christoph: Rule-based technical management for the dependable operation of networked building automation systems. In: *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2017, S. 612–615

Betreute Arbeiten

2013

HOLTERBORK, Katrin: *Ortstransparenter Zugriff auf Managementvariablen mittels einer verteilten Baumstruktur nach OMA DM*, Technische Universität Dortmund, Fakultät für Informatik, Masterarbeit, September 2013

SCHULTE-HOBEIN, Nina: *Konzeption und Implementierung eines XMPP-Event-Service für einen verteilten Managementbaum nach OMA DM auf Basis der OSGi-Software-Plattform*, Hochschule Emden/Leer, Fachbereich Technik, Abteilung Elektrotechnik und Informatik, Projektarbeit, Oktober 2013

BURKERT, Malte: *Modellbasiertes vereinheitlichtes Management heterogener IaaS-Ressourcen*, Technische Universität Dortmund, Fakultät für Informatik, Masterarbeit, Oktober 2013

2014

JORDAN, Annika: *Adaptable Ensembles Supporting Dynamic and Reliable Service Bindings in Ambient OSGi Systems*, Technische Universität Dortmund, Fakultät für Informatik, Masterarbeit, Februar 2014

STICKAN, Mirco: *Generische Handlerarchitektur zum offenen Zugriff auf Management-, Konfigurations- und Statusgrößen und Spezialisierung zur Anbindung von SNMP-Agenten*, Technische Universität Dortmund, Fakultät für Informatik, Masterarbeit, März 2014

KONOPKA, Marcel: *3D-Visualisierung der automatisierten Überwachung und Konfiguration verteilter Clouds nach SLA-Elementen und ECA-Regelsätzen*, Technische Universität Dortmund, Fakultät für Informatik, Bachelorarbeit, Mai 2014

2015

KLASSEN, Mathias: *Icinga-Kommunikationshandler für das Management-Tree-basierte Monitoring virtueller Server-Infrastrukturen*, Technische Universität Dortmund, Fakultät für Informatik, Bachelorarbeit, Juni 2015

ALFES, Fabian: *Verknüpfung und gemeinsame Darstellung der Konfigurations- und Statusdaten eines Gebäudeautomatisierungssystems mit einem 3D-Gebäudemodell*, Technische Universität Dortmund, Fakultät für Informatik, Bachelorarbeit, August 2015

2016

TÖNGES, Patrick: *Spheres-Server für den Multi-Administrator-Zugriff auf die Managementdaten verteilter Systeme unter Verwendung domänenspezifischer Sichten*, Technische Universität Dortmund, Fakultät für Informatik, Bachelorarbeit, Dezember 2016

2017

BERGMANN, Jim-Marvin: *Layout zur Visualisierung von Status- und Konfigurationsdaten des technischen Systemmanagements*, Technische Universität Dortmund, Fakultät für Informatik, Bachelorarbeit, Juli 2017

2018

POPOVIC, Dragana: *Ein Docker- und Puppet-basiertes Konfigurationssystem für sporadisch vernetzte IoT-Geräte*, Technische Universität Dortmund, Fakultät für Informatik, Masterarbeit, August 2018