

## **Modellierung von Qualitätsmerkmalen für Services**

**Dissertation  
zur Erlangung des akademischen Grades  
"doctor rerum naturalium"  
(Dr. rer. nat.)  
in der Wissenschaftsdisziplin "Business Process Technology"**

**eingereicht an der  
Mathematisch-Naturwissenschaftlichen Fakultät  
der Universität Potsdam**

**von  
Jens Hündling**

**Potsdam, den 10. März 2008**

# Modellierung von Qualitätsmerkmalen für Services

Jens Hündling

23. November 2008



# Danksagung

Die vorliegende Dissertation entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Fachgebiet für Business Process Technology von Prof. Dr. Mathias Weske am Hasso-Plattner-Institut für Softwaresystemtechnik an der Universität Potsdam. Mein Dank gilt allen denjenigen, die mich bei der Erstellung der Arbeit unterstützt haben. Ich danke meinem Doktorvater Mathias Weske für die Unterstützung und Begleitung meines Werdegangs in Forschung, Lehre und wissenschaftliches Arbeiten, die bereits während meines Studium der Wirtschaftsinformatik an der Westfälischen Wilhelms-Universität in Münster begann. Mein Dank gilt auch den großartigen Mitarbeitern am Lehrstuhl und am Hasso-Plattner-Institut. Insbesondere möchte ich Katrin Heinrich, Arnd Schnieders, Frank Puhmann, Dominik Kuropka, Guido Laures, Harald Meyer, Harald Overdick für die vielen inspirierenden Diskussionen und konstruktive Kritik danken. Harald Schubert danke ich für seine Arbeiten im Rahmen des Praxisprojektes (siehe Kapitel 5.6). Bei der praktischen Umsetzung haben mir Diskussionen mit Michael Bräuer den richtigen Impuls gegeben. In diesem Rahmen möchte ich mich auch bei Ramzi Musa für die Unterstützung und die notwendigen Freiräume bedanken. Bei Norbert Gronau und Andreas Oberweis bedanke ich mich für die Begutachtung meiner Dissertationschrift.

Ein großes Dankeschön gilt meinem Freund und Kollegen Hilmar Schuschel, mit dem ich inzwischen viele Jahre gemeinsam gearbeitet und geforscht habe. Unsere wertvollen Diskussionen mir vielfach den richtigen Weg gewiesen. Darüber hinaus danke ich auch Christopher Robinson-Mallett sowohl für fachliche Diskussionen als auch private Ablenkung.

Ein kaum in Worte zu fassender Dank gilt auch meinen Eltern Gerda und Gerd, die mir alles überhaupt erst ermöglicht haben und mich zu dem gemacht haben, was ich bin. Ebenso möchte ich mich für den Rückhalt, die ständige Unterstützung und den notwendigen Ausgleich bei meinen Geschwistern Andree und Dörte, sowie allen Freunden aufrichtig bedanken.

Schließen möchte ich mit dem größten Dank: Ines, ohne dich wäre alles niemals möglich gewesen. Dein Anteil an dieser Arbeit ist unermesslich und die höchste Qualität finde ich täglich im Lächeln unserer Tochter Mina. Selbst das Qualitätsmodell meiner Arbeit könnte dies kaum erfassen ...

Jens Hündling

Berlin, 22. November 2008



# Zusammenfassung

Unternehmen wachsen heutzutage immer stärker zusammen, um gemeinsam Geschäftsziele zu verfolgen. Dabei ist ein Wandel von Produkten hin zu Services erkennbar. Dieser Wandel spiegelt sich in der Informationstechnologie im neuen Paradigma der Service-Orientierung wider. Ein Service stellt eine angebotene Leistung dar, die für einen Dritten erbracht werden kann. Service-Orientierung verspricht eine effiziente, eng verzahnte Integration der Unternehmen. Dahinter steht die Vision der weitgehend automatisierten Kooperation auf der Basis einer serviceorientierten Informationstechnologie. Gleichzeitig erhöht sich die Flexibilität, da weniger starre Abhängigkeiten zwischen den Geschäftspartnern entstehen und sich neue Geschäftspartner technisch relativ einfach integrieren lassen. Allerdings sind die Konzepte der Service-Orientierung noch nicht ausgereift und wesentliche Aspekte, insbesondere in Bezug auf Service-Auswahl und Geschäftsprozesse, sind kaum berücksichtigt.

Ein zentrales Problem ist bei der Suche nach einem Service unter verschiedenen, funktional äquivalenten Services denjenigen auszuwählen, dessen Eigenschaften (beispielsweise Kosten, Ausführungsdauer und Zuverlässigkeit) den eigenen Anforderungen am besten entsprechen. Vereinfacht ausgedrückt: Es ist das Ziel, einen vergleichsweise kostengünstigen, schnellen und zuverlässigen Service zu finden. Für derartige Service-Eigenschaften wird gezeigt, dass es bislang keine ausreichenden Beschreibungsmöglichkeiten gibt, so dass Suche und Auswahl nicht angemessen realisierbar sind. Es wird in dieser Arbeit ebenfalls gezeigt, dass durch eine fehlende konzeptionelle Grundlage auch die Möglichkeiten einen Service anzubieten eingeschränkt sind. Daher wird in dieser Arbeit ein Modell zur Beschreibung von derartigen Service-Eigenschaften konstruiert.

In einem ersten Schritt werden die Konzepte der Service-Orientierung untersucht, zentrale Entitäten identifiziert und in einem Service-Modell abgebildet. Darauf aufbauend werden Service-Eigenschaften als Qualitätsmerkmale modelliert. Hierzu wird ein formales Meta-Modell für Qualitätsmerkmale erarbeitet und untersucht. Präzision und Eindeutigkeit soll erreicht werden, indem unter anderem die Semantik der Werte und Maßeinheiten spezifiziert werden. Ein Anbieter kann dann eigene Services besser positionieren und von der Konkurrenz abgrenzen. Verfügbare Services können bezüglich ihrer Qualität untersucht werden und es kann für eine bestimmte Suche eine Rangfolge der passenden Services erstellt werden. Diese gibt neben dem besten Service auch alternative Services an, die genutzt wer-

---

den können, falls ein ausgewählter Service ausfallen sollte. Das in dieser Arbeit entwickelte Qualitätsmodell führt somit Angebot und Nachfrage in einer serviceorientierten Umgebung besser und effizienter zusammen. Dadurch kann der zeit- und kostenintensive Einsatz von menschlichen Experten reduziert und der Automatisierungsgrad erhöht werden. Das Qualitätsmodell ist als relationales Datenmodell realisiert worden und es ist eine prototypische, Web-basierte Anwendung zur Service-Auswahl implementiert worden, die zentrale Aspekte des Qualitätsmodells berücksichtigt.

Neben der Service-Auswahl besteht das Problem, dass aufgerufene Services die Ausführung eines Geschäftsprozesses beeinflussen, so dass beispielsweise Ausführungsdauer und Kosten des Geschäftsprozesses unklar sind. Derartige Angaben sind aber für die Entwicklung und das Management von Geschäftsprozessen und letztendlich für den unternehmerischen Erfolg von zentraler Bedeutung. Selbst wenn die Qualitätsmerkmale eines Services auf Basis des Qualitätsmodells dieser Arbeit spezifiziert sind, ist deren Einfluss auf den Geschäftsprozess nicht unmittelbar erkennbar. Als weiterer Beitrag dieser Arbeit werden daher Aggregationsmuster formal definiert, die auf dem Qualitätsmodell aufbauen. Die Aggregationsmuster beschreiben, wie die Ausprägungen der Qualitätsmerkmale der einzelnen Services in einem Geschäftsprozess aggregiert werden können. Es wird untersucht, wie unter Benutzung der Muster die Qualitätsmerkmale von Geschäftsprozessen ermittelt werden. Schließlich wird die Validierung der Arbeit im Rahmen eines Praxisprojektes durch die Anwendung auf Geschäftsprozesse beschrieben.

# Abstract

Nowadays, cooperations between companies are more and more expanding, up to achieving common business goals. Additionally, a shift from products to services can be observed. These changes are reflected in the Information Technology (IT) by the arising paradigm of service orientation. Basically, services are offered by their providers for a third party. Service Oriented Computing offers a promising approach for global businesses and integrated, virtual enterprises. A service oriented IT fosters the vision of a widely automated cooperation. Additionally, flexibility is enhanced, because dependencies between business partners are less static and technical integration of new business partners can relatively easy be achieved. Nevertheless, key aspects of service orientation are still immature, especially regarding service selection and business processes.

A major problem of service selection is selecting a specific service (among several functionally equivalent services) that offers the best non-functional properties (like cost, execution duration and reliability). In a nutshell: The aim is to select a comparatively cheap, fast and reliable service. In this thesis it is shown that sufficient models for characterizing such service properties are not existing. Thus, appropriate service discovery and selection is not feasible. Even more, it is shown that due to a missing conceptual underpinning, the possibilities to offer a service are also limited. Hence, a model for describing such service properties is developed in this thesis.

In a first step, the concepts of service orientation are examined, key entities are identified and a service model is build. Based on the service model, the considered service properties are modeled as quality attributes. For this purpose, a formal meta-model for quality attributes is developed and analyzed. Accuracy and unambiguosness is achieved, for instance, by specifying semantics of the values and measurement units. From the viewpoint of a service provider, this results in enhanced positioning options and competitive advantages. From the viewpoint of a service requestor, the quality of available services can be analyzed and a ranking order of suitable service can be established. The ranking order also holds alternative services that can be used, if the execution of a selected service fails. Thus, the quality model developed in this thesis consolidates demand and supply in a service oriented environment more efficiently. In the end, the time- and cost-intensive assignment of human experts can be reduced and the degree of automation can be enhanced. The quality model is realized as relational datamodel and a prototypical,



---

Web-based application for service selection has been implemented, which utilizes key concepts of the quality model.

Another problem arises, when business processes using services are considered: Invoked service are influencing the execution of business processes such that, for instance, the overall execution time and the overall cost of the business process can be ambiguous. However, these details are vital for development and management of business processes and eventually the business success. Even if the quality attributes are modeled according to the quality model of this thesis, their influence on the overall business process might not be directly identifiable. Thus, the definition of aggregation patterns for quality attributes forms another contribution of this thesis. The aggregation patterns are based on the quality model and describe, how the values of the quality attributes of each service can be aggregated in the context of a business process. Furthermore, it is examined how the patterns lead to quality attributes for the overall business process. Finally, the thesis is validated in a real-life project that applies the quality model and the aggregation patterns to business processes in practice.

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	Hintergrund und Motivation . . . . .	2
1.2	Problemstellung . . . . .	5
1.3	Lösungsansatz . . . . .	8
1.4	Wissenschaftlicher Beitrag und Aufbau der Arbeit . . . . .	11
<b>2</b>	<b>Grundlagen der Service-Orientierung</b>	<b>13</b>
2.1	Hintergrund . . . . .	13
2.1.1	Wirtschaftliche Entwicklung . . . . .	14
2.1.2	Entwicklung der Informationstechnologie . . . . .	15
2.1.3	Zusammenfassung . . . . .	18
2.2	Serviceorientierte Architektur . . . . .	18
2.2.1	Begriffsdefinitionen . . . . .	20
2.2.2	Service Provider . . . . .	23
2.2.3	Service Broker . . . . .	26
2.2.4	Service Requestor . . . . .	27
2.2.5	Weitere Rollen . . . . .	31
2.3	Web Services . . . . .	31
2.4	Zusammenfassung . . . . .	35
<b>3</b>	<b>Service-Modell</b>	<b>37</b>
3.1	Annahmen . . . . .	37
3.2	Formulierung des Service-Modells . . . . .	38
3.3	Abstraktionsebenen . . . . .	44
3.4	Prinzipien . . . . .	46
3.4.1	Allgemeine Prinzipien . . . . .	47
3.4.2	Kommunikationsprinzipien . . . . .	49
3.4.3	Zusammengesetzter Service . . . . .	53
3.4.4	Lose Kopplung . . . . .	56
3.5	Das Paradigma der Service-Orientierung . . . . .	58
3.6	Service-Modell und Web Services . . . . .	59
3.7	Zusammenfassung . . . . .	60

<b>4</b>	<b>Qualitätsmodell</b>	<b>63</b>
4.1	Beispiele . . . . .	64
4.1.1	Video-Service . . . . .	64
4.1.2	Transport-Service . . . . .	69
4.2	Annahmen und Anforderungen . . . . .	72
4.2.1	Allgemeine Anforderungen . . . . .	73
4.2.2	Anwendungsfälle . . . . .	75
4.3	Entwurf des Modells . . . . .	79
4.3.1	Begriffsdefinition und Abgrenzung . . . . .	79
4.3.2	Elemente des Qualitätsmodells . . . . .	81
4.3.3	Integration des Qualitätsmodells in das Service-Modell . . . . .	88
4.3.4	Metrik . . . . .	89
4.3.5	Skalenniveau . . . . .	90
4.3.6	Anfragen und Restriktionen . . . . .	94
4.3.7	Nutzenfunktion . . . . .	96
4.4	Realisierung des Modells . . . . .	99
4.4.1	Realisierung des Transport-Beispiels . . . . .	100
4.4.2	Realisierung als Relationales Datenmodell . . . . .	100
4.5	Diskussion möglicher Erweiterungen . . . . .	105
4.5.1	Maßeinheiten . . . . .	105
4.5.2	Richtung der Qualitätssteigerung . . . . .	109
4.5.3	Komplexe Qualitätsmerkmale . . . . .	110
4.5.4	Allgemeine und spezielle Qualitätsmerkmale . . . . .	111
4.5.5	Logische und physische Qualitätsmerkmale . . . . .	112
4.5.6	Statische und dynamische Qualitätsmerkmale . . . . .	113
4.6	Zusammenfassung . . . . .	114
<b>5</b>	<b>Geschäftsprozesse</b>	<b>117</b>
5.1	Grundlagen der Geschäftsprozesse . . . . .	118
5.1.1	Graphbasierte Prozessbeschreibungssprachen . . . . .	119
5.1.2	Blockstrukturierte Prozessbeschreibungssprachen . . . . .	120
5.1.3	Weitere Ansätze . . . . .	120
5.2	Service-Auswahl . . . . .	121
5.2.1	Prozessbeispiel . . . . .	123
5.2.2	Service-Auswahl zur Modellierungszeit . . . . .	124
5.2.3	Service-Auswahl während Prozessinstanziierung . . . . .	127
5.2.4	Service-Auswahl zur Laufzeit . . . . .	128
5.2.5	Zusammenfassung der Varianten . . . . .	130
5.3	Kontrollflussblöcke . . . . .	132
5.3.1	Workflow-Patterns . . . . .	132
5.3.2	Bildung von Kontrollflussblöcken . . . . .	135
5.4	Aggregationsmuster für Qualitätsmerkmale . . . . .	139
5.4.1	Vorbemerkungen und Annahmen . . . . .	139
5.4.2	Summe . . . . .	142

5.4.3	Produkt . . . . .	144
5.4.4	Extremwerte: Maximum und Minimum . . . . .	145
5.4.5	Aggregationen unter Wahrscheinlichkeiten . . . . .	146
5.4.6	Mittelwerte . . . . .	148
5.4.7	Aggregation der ersten N Prozessschritte . . . . .	150
5.4.8	Qualitätsmerkmal übernehmen . . . . .	152
5.4.9	Aggregation mehrerer Qualitätsmerkmale . . . . .	153
5.5	Metrik der aggregierten Werte . . . . .	154
5.6	Order-To-Cash-Prozess . . . . .	155
5.6.1	Qualitätsmerkmale . . . . .	156
5.6.2	Analyse des Geschäftsprozesses . . . . .	157
5.6.3	Modellierung der Aggregation . . . . .	159
5.6.4	Ermittlung der gültigen Service-Auswahl . . . . .	159
5.7	Zusammenfassung . . . . .	159
<b>6</b>	<b>Verwandte Arbeiten</b>	<b>161</b>
6.1	Einordnung des Qualitätsmodells . . . . .	161
6.1.1	Business-Oriented-Management . . . . .	162
6.1.2	Extended SOA (xSOA) . . . . .	162
6.1.3	Quality of Services als Web Service Schicht . . . . .	163
6.2	Rechnernetze und Software-Entwicklung . . . . .	164
6.2.1	QoS und SLA in Rechnernetzen . . . . .	164
6.2.2	Qualitätsmerkmale in der Software-Entwicklung . . . . .	164
6.3	Qualitätsmodelle für serviceorientierte Architekturen . . . . .	168
6.3.1	Qualitätsmodell im MAIS-Projekt . . . . .	168
6.3.2	WS-QoS Framework . . . . .	169
6.3.3	Web Services Selection Framework . . . . .	170
6.3.4	OASIS Web Services Quality Model (WSQM) . . . . .	170
6.3.5	Web Services Level Agreements (WSLA) . . . . .	172
6.3.6	Das WS-Policy Framework . . . . .	173
6.4	Qualitätsmodelle für zusammengesetzte Services . . . . .	174
6.4.1	Qualitätsmodelle für das METEOR-S-Projekt . . . . .	174
6.4.2	Blockstruktur-basierte Aggregation von Qualitätsmerkmalen	176
6.5	Automatisierte Geschäftsprozessoptimierung . . . . .	177
6.5.1	Überblick . . . . .	178
6.5.2	Ganzzahlige Lineare Optimierung . . . . .	179
6.5.3	Das QoS Capable Web Service Composition-Projekt . . . . .	180
<b>7</b>	<b>Diskussion</b>	<b>183</b>
7.1	Service-Modell . . . . .	183
7.2	Qualitätsmodell . . . . .	186
7.3	Geschäftsprozesse . . . . .	191

Inhaltsverzeichnis

---

<b>8 Zusammenfassung</b>	<b>195</b>
8.1 Rekapitulation . . . . .	195
8.2 Ausblick . . . . .	196
8.3 Fazit . . . . .	197
<b>Literaturverzeichnis</b>	<b>199</b>

# Abbildungsverzeichnis

1.1	Die Rollen und Beziehungen einer serviceorientierte Architektur in Anlehnung an [25] . . . . .	3
1.2	Einordnung des Modells . . . . .	10
2.1	Ablauf der Veröffentlichung eines Services durch einen Service Provider, dargestellt als UML Aktivitätsdiagramm . . . . .	24
2.2	Aktivitätsdiagramm eines Service Requestor zur Service-Auswahl	29
2.3	Wesentliche Elemente des WSDL-Modells . . . . .	33
3.1	Das Service-Modell . . . . .	43
3.2	Direkte Bindung an einen Service . . . . .	51
3.3	Dynamische Bindung an einen Service unter Benutzung eines Brokers . . . . .	52
3.4	Dynamische Bindung an einen Service und Verhandlung eines SLA's	53
3.5	Erweiterung des Service-Modells für zusammengesetzte Services .	54
4.1	Beispiel-Geschäftsprozess: Produktionprozess, dargestellt in BPMN	72
4.2	Integration des Qualitätsmodells und des Service-Modells . . . . .	87
4.3	Relationales Datenmodell (Auszug) . . . . .	102
4.4	Relationales Datenmodell der Anfragen und Nutzenfunktion (Auszug) . . . . .	103
4.5	Architektur-Skizze der Oracle APEX Anwendung [108] . . . . .	104
4.6	Eingabe der Nutzenwerte für eine Nutzenfunktion (Screenshot) . .	104
4.7	Ergebnis der Nutzenfunktions-Anfrage auf die Services des Logistikbeispiels (Screenshot) . . . . .	107
4.8	Zusammensetzung des Gesamtnutzen für den Service mit der ID 500 (Screenshot) . . . . .	108
5.1	Prozessdefinition des Produktionsprozess, erweitert um Qualitätsmerkmale Kosten und Zeit sowie Service-Typ $T_{shipping}$ , dargestellt in BPMN [107] . . . . .	123
5.2	Produktionsprozess mit ausgewähltem Service $S_4$ . . . . .	124
5.3	Produktionsprozess mit individuellem Service Level Agreement ( $S_4^{SLA}$ ) . . . . .	126

## Abbildungsverzeichnis

---

5.4	Prozessinstanz mit ausgewähltem Service $S_1$ . . . . .	127
5.5	Prozessinstanz mit verzögertem Prozessschritt Produktion und ausgewähltem Service $S_3$ . . . . .	129
5.6	Kontrollflussblöcke Sequenz . . . . .	136
5.7	Kontrollflussblöcke mit Parallel-Split (AND und AndNM) . . . . .	137
5.8	Kontrollflussblöcke mit Multiple und Exclusive Choice (OR und XOR) . . . . .	138
5.9	Kontrollflussblock zur Realisierung von Schleifen (Loop) . . . . .	139
5.10	Kontrollfluss-Blöcke mit Angabe der Wahrscheinlichkeiten . . . . .	148
5.11	Teilprozess Auftragsbearbeitung des Order-To-Cash-Prozesses, dargestellt in BPMN . . . . .	158
7.1	Explizite Service-Auswahl als Prozessschritt nach vorheriger Preisanfrage an Provider . . . . .	187

# Tabellenverzeichnis

3.1	Zusammenhang WSDL-Modell und Service-Modell . . . . .	60
4.1	Intuitive Darstellung der relevanten Qualitätsmerkmale für Video-Services . . . . .	65
4.2	Übersicht über die angebotenen Transport-Services und die Ausprägungen der Eigenschaften . . . . .	70
5.1	Die Qualitätsmerkmale für den Order-To-Cash-Prozess . . . . .	156
5.2	Aggregationsmuster für die Qualitätsmerkmale je Kontrollflussblock	158
5.3	Aggregierte Ausprägungen der Qualitätsmerkmale für den Order-To-Cash-Prozess für die Service-Kombinationen I-IV (mittlere Auslastung: 5 tps, Laufzeit: 1 Monat) [134] . . . . .	159
7.1	Nutzen für die Qualitätsmerkmale Sicherheit und Statusüberwachung. . . . .	190





# Kapitel 1

## Einleitung

*Things should be made as simple as possible, but no simpler.*  
*Albert Einstein*<sup>1</sup>

Mit seiner Aussage bezog sich Einstein auf erklärende Theorien in der Wissenschaft. Sein Hinweis lässt sich somit auf Modelle zur Erklärung und zum Verständnis von Theorien übertragen. Diese Argumentation soll einen Leitfaden für diese Arbeit darstellen, in der ein Modell für Qualitätsmerkmale von Services entwickelt wird.

In der Informatik sind Modelle ein probates Mittel, um komplexe Zusammenhänge abzubilden und zu erklären [123]. Dabei ist ein Modell eine Abbildung der realen Welt zum Zwecke eines Subjektes [11, 105, 133]. Der Zweck bestimmt dabei, von welchen Aspekten abstrahiert werden kann, um ein möglichst einfaches Modell zu erhalten, wie es von Einstein postuliert wurde. Dadurch wird das Modell verständlicher und einfacher zu handhaben. Darüber hinaus werden Modelle in der Informatik auch dazu benutzt, um reale Entitäten und komplexe Zusammenhänge so abzubilden, dass Lösungen in einer Informationstechnologie (IT) möglich werden. Somit wird auch hier von irrelevanten Aspekten abstrahiert und genau diejenigen werden abgebildet, die für die Aufgabe der IT benötigt werden. Der zweite Teil von Einsteins Aussage bedeutet, dass ein Modell nicht einfacher sein soll als möglich. Es sollte also nicht von relevanten Aspekten abstrahiert werden, um die Komplexität zu beherrschen.

Um zu erkennen, welche Aspekte für das zu entwickelnde Modell relevant sind, wird konsequent der Zweck des Modells beachtet. Vorrangiger Zweck dieses Modells ist es, bei der Suche nach einem Service, der eine bestimmte Leistung erbringen kann, unter mehreren adäquaten Services denjenigen zu erkennen, dessen Eigenschaften den eigenen Anforderungen am besten entsprechen.

---

<sup>1</sup>Dieses Zitat wird Albert Einstein zugeschrieben und ist eine Paraphrase des Ausspruchs „The supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience.“, Albert Einstein, 1933. Andere Quelle verwenden ähnliche Umformulierungen, wie z.B. „Everything should be made as simple as possible, but not one bit simpler.“[171].

## 1.1 Hintergrund und Motivation

Nachdem in der industriellen Revolution die Produktion von Massengütern begann und immer weiter optimiert wurde, stand im letzten Jahrhundert vor allem der Kunde im Mittelpunkt des Interesses [98]. Ein weiterer Entwicklungsschritt führt zur Service-Orientierung, die als zentrale Entität den Service fokussiert. Wesentlich sei hier zunächst, dass ein Service eine Funktionalität so zur Verfügung stellt, dass sie von Anderen genutzt werden kann. In der Service-Orientierung wird die Perspektive des Kunden ebenso betrachtet, wie die Perspektive des Anbieters mit seinem internen Service-Management [32, 118]. Das heißt, ein Service wird so entwickelt, verwaltet und zur Verfügung gestellt, dass er von Dritten benutzt und in dessen Geschäftsprozesse integriert werden kann. Es wird schon bei der Entwicklung der Services darauf geachtet, dass sie in unterschiedlichen Kontexten benutzbar sind. Dadurch ist es in einer serviceorientierten Umgebung möglich, von Anderen zur Verfügung gestellte Funktionalität effizient zu nutzen.

Nicht zuletzt wegen des Wandels zur Service-Orientierung, wachsen Unternehmen heutzutage immer stärker zusammen, bis hin zu so genannten virtuellen Unternehmungen [97], in denen mehrere Geschäftspartner gemeinsame Ziele verfolgen [27, 86]. Die Informationstechnologie ermöglicht eine effiziente, eng verzahnte Integration von Unternehmen [34]. In der Informationstechnologie ist die Service-Orientierung konsequenterweise ein aufstrebendes, neues Paradigma, das einen viel versprechenden Ansatz liefert, um die Komplexität der Integrationsaufgaben zu handhaben [4]. In der englischsprachigen Literatur wird dieses Paradigma auch als *Service Oriented Computing* bezeichnet [80, 139].

Durch die Anwendung des serviceorientierten Paradigmas in der Informationstechnologie wird die Integration der Unternehmen unterstützt und ausgebaut. Dahinter steht die Vision der weitgehend automatisierten Kooperation [96] auf der Basis einer serviceorientierten Informationstechnologie [117, 102]. Es soll also möglich sein, gemeinsame Geschäftsziele mit möglichst wenigen menschlichen Interaktionen zu erreichen. Somit wird der zeit- und kostenintensive Einsatz von menschlichen Experten auf ein Minimum reduziert. Gleichzeitig erhöht sich die Flexibilität, da weniger starre Abhängigkeiten zwischen den Geschäftspartnern entstehen. Neue Geschäftspartner können relativ einfach in einen Geschäftsprozess integriert werden, wenn sie beispielsweise einen günstigeren oder zuverlässigeren Service anbieten. An dieser Stelle soll die Idee der Service-Orientierung und ihre Motivation zunächst kurz skizziert werden; eine ausführliche Erläuterung der Service-Orientierung und eine Definition der zentralen Begriffe folgt in Kapitel 2 und eine formale Modellbildung in Kapitel 3.

**Service** Zentrales Element des serviceorientierten Paradigmas ist das Konzept eines Service. Dieser kapselt eine bestimmte Funktionalität und macht sie für Dritte verfügbar. Der Anbieter eines Services (in der englischen Literatur als *Provider* bezeichnet) erzeugt dazu eine Beschreibung des Services, die er bei einem Vermittler (englisch *Broker*) veröffentlicht (*publish*). Wird nun ein Service benötigt,

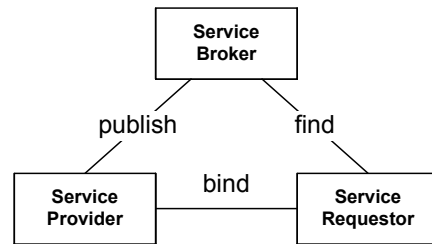


Abbildung 1.1: Die Rollen und Beziehungen einer serviceorientierten Architektur in Anlehnung an [25]

so wird eine Anfrage an einen Broker gestellt, der einen passenden Service vermittelt (*find*). Die Rolle eines derartigen Kunden wird in der englischen Literatur als *Requestor*, also Nachfrager, bezeichnet. Der Requestor kann dann die von einem Provider zur Verfügung gestellte Funktionalität nutzen (*bind*). Die Architektur einer auf diesen Prinzipien basierenden Umsetzung in der IT wird als *serviceorientierte Architektur* bezeichnet und ist in Abbildung 1.1 dargestellt.

Die Funktionalität wird auf der Seite des Providers realisiert; dadurch wird eine Leistung erbracht. Somit wird bei der Ausführung eines Services vom Provider für den Requestor eine Leistung erbracht. Ein Service ist also eine angebotene Leistung, die erbracht wird, wenn der Service aufgerufen wird. Die Leistung wird angeboten, indem eine Beschreibung bei einem Broker veröffentlicht wird. Um zu illustrieren, welche Arten von Leistungen möglich sind, sollen hier kurz einige Beispiel-Services skizziert werden:

- **Buchversand:** Ein Buchhändler bietet einen Service „Buchversand“ an, mit dem er Bestellungen für Bücher annimmt und die Bücher versendet. Der Buchhändler hat dazu eine Service-Beschreibung erstellt und beim Broker veröffentlicht, die beschreibt, wie ein Kunde ein Buch bestellen kann. Ein Kunde kann mit Hilfe dieser Beschreibung eine elektronische Bestellung erstellen, indem er beispielsweise die ISBN, die Lieferadresse und seine Bankverbindung für die Bezahlung als Service-Aufruf an den Buchhändler sendet. Erhält der Buchhändler nun eine Bestellung, so läuft bei ihm intern und für den Kunden transparent die Bestellungsbearbeitung ab. Das bestellte Buch wird dabei aus dem Lager geholt, verpackt, per Post versandt und der fällige Betrag wird beispielsweise per Lastschrift eingezogen. Die Leistungserbringung des Services besteht hier also aus verschiedenen Einzelschritten, die vom Provider gesteuert werden. Aus Sicht des Kunden wird allerdings eine kohärente Leistung angeboten und der interne Ablauf ist für ihn nicht interessant. Es sei hier darauf hingewiesen, dass ein Teil der Leistungserbringung über den Postversand des Buches an die angegebene Lieferadresse erfolgt. Dieser Anteil wird zwar nicht direkt vom Buchhändler erbracht, aber für den Kunden ist dies ebenfalls nicht relevant.
- **Transport:** Ein Transport-Unternehmen bietet einen Service „Transport“ an,

Güter von einem Ort abzuholen und sie an einen anderen Ort zu liefern. Hierfür ist eine Service-Beschreibung veröffentlicht worden. Die Leistung kann von einem Kunden auf elektronischem Wege bestellt werden, indem Abholadresse und -datum, sowie die Lieferadresse übermittelt werden. Das Unternehmen plant die Transporte intern, teilt dafür Fahrer und Fahrzeuge ein, und erstellt die Rechnungen, die den Kunden zugestellt werden. Der physikalische Transport wird dann von den Fahrern und Fahrzeugen des Transport-Unternehmens durchgeführt. Für den Kunden sind die internen Abläufe nicht relevant. Er lässt die Güter abholen und erhält eine Rechnung, die er dann bezahlt.

- **Video:** Ein Unternehmen bietet Spielfilme per Internet an. Die Auswahl und Bestellung des Films erfolgen über einen elektronischen Kommunikationskanal der serviceorientierten Umgebung. Die Leistungserbringung erfolgt dann über einen separaten Video-Stream, nicht über den Kommunikationskanal der serviceorientierten Umgebung.
- **Datenanalyse:** Eine statistische Datenanalyse wird als Service angeboten. Die Daten werden beim Service-Aufruf übermittelt, die Leistungserbringung erfolgt durch eine Software zur statistischen Datenanalyse auf einem Server des Providers. Die Ergebnisse werden als Antwort auf den Service-Aufruf über einen Kommunikationskanal der serviceorientierten Umgebung an den Requestor zurückgesandt.

Es zeigt sich, dass Services sowohl digitale Produkte [168] anbieten können (der Stream eines Films im Video-Service und die Ergebnisse der Datenanalyse) als auch nicht digitale Produkte (Buchversand) sowie Dienstleistungen (Transport). Auch bei den nicht digitalen Produkten und Dienstleistungen ist ein Teil der Leistungserbringung (beispielsweise Bestellung und Bezahlung) in elektronischer Form in die serviceorientierte Architektur eingebunden. Weiterhin fällt auf, dass beim Buchversand ein Teil der Leistung durch die Post, also einen weiteren Geschäftspartner, erbracht wird. Dennoch ist der Service die komplette angebotene Leistung, die bei einem Aufruf des Service erbracht wird.

Kernidee und Hauptanforderung an IT-Systeme zur Realisierung einer serviceorientierten Architektur ist der Einsatz standardisierter Spezifikationen, vor allem für die Service-Beschreibung, für Kommunikationsprotokolle und zur Verwaltung der Services beim Broker. Es existieren grundlegende Spezifikationen für eine informationstechnischen Umsetzung [178], die als De-facto-Standard gelten [4, 139].

Die Konzepte zur Umsetzung des serviceorientierten Paradigmas sind aber noch nicht ausgereift [96, 115, 151, 161]. Bislang ist kaum betrachtet, was neben der technischen Realisierung der Kommunikation noch notwendig ist, um dem serviceorientierten Paradigma zu genügen [32]. Dies zeigt sich vor allem daran, dass wesentliche Aspekte fehlen, die Voraussetzungen für einen Einsatz in der realen Geschäftswelt sind [91]. Im folgenden Abschnitt wird dies genauer betrachtet und die für diese Arbeit relevanten Probleme werden diskutiert.

## 1.2 Problemstellung

Ein zentrales Problem für einen Requestor ist, bei der Suche nach einem Service unter verschiedenen, funktional äquivalenten<sup>2</sup> Services denjenigen zu erkennen, dessen Eigenschaften den eigenen Anforderungen am besten entsprechen. Diese Anforderung eines Requestors resultiert aus der Übertragung der Prinzipien und Rollen eines Marktes [53] auf serviceorientierte Architekturen [42]. Konkreter ausgedrückt: Es ist das Ziel des Requestors, einen möglichst kostengünstigen, schnellen und zuverlässigen, sowie qualitativ hochwertigen Service zu finden.

Für derartige Service-Eigenschaften, die auch als *Qualitätsmerkmale* bezeichnet werden, existieren bislang keine ausreichenden Konzepte zur Beschreibung. Die Service-Beschreibung kann zwar um Zahlen oder Texte erweitert werden, beispielsweise kann der Provider des Transport-Services aus obigem Beispiel Angaben wie „Kosten: 100“ oder „Dauer: 24“ der Service-Beschreibung hinzufügen. Hierbei ist allerdings die Bedeutung der Werte unklar: In welcher Währung sind die Kosten angegeben? Ist die Mehrwertsteuer enthalten? In welcher Zeiteinheit ist die Dauer angegeben? Beschreibt die Dauer die Zeitdauer der Lieferung von Abholung bis Zustellung oder die Gesamtdauer ab Service-Aufruf? Die Bedeutung der Werte bleibt somit ebenso unklar wie die Maßeinheiten und das Messverfahren. Werden Services von unterschiedlichen Providern angeboten, so können die Werte beispielsweise bei unterschiedlicher Währung oder abweichender Bedeutung (inklusive und exklusive Steuern) nicht verglichen werden. Dadurch kann der Requestor nicht entscheiden, welches der beste Service ist<sup>3</sup>.

Die Suche und Auswahl eines geeigneten Services durch einen Requestor ist somit nicht angemessen realisierbar. Zusätzlich sind die Möglichkeiten eines Providers, seine Services angemessen zu beschreiben und anzubieten, eingeschränkt. Dass die Service-Eigenschaften nicht ausreichend betrachtet werden, resultiert aus dem bisherigen Fokus der Konzepte auf die rein technische Beschreibung eines Services beim Broker. Ziel der bisherigen Spezifikationen wie WSDL [181] zur Service-Beschreibung ist lediglich die Beschreibung der Kommunikationsschnittstellen [91]. Dadurch wird es einem Requestor ermöglicht, einen Service aufzurufen und somit die Leistungserbringung anzustoßen. Beim Service Broker wird hinterlegt, welche Funktionalität der Service anbietet, also *was* der Service macht. Dazu sind beispielsweise in der Service Broker-Spezifikation UDDI [35] textuelle Beschreibungen und Schlüsselwörter (so genannte tModel-Keys) vorgesehen. Deren Verwendung für Service-Eigenschaften ist aber weder standardisiert, noch sind ausreichende Konzepte dafür bislang vorgesehen.

Zur Abgrenzung sei an dieser Stelle anzumerken, dass die Eigenschaften sich nicht auf die Funktionalität eines Services beziehen. Es wird also nicht untersucht,

---

<sup>2</sup>Zur Erklärung der Funktionalität eines Services sei auf die Abschnitte 2.2.1 (Begriffdefinitionen) und 2.3 (Realisierung im Kontext von Web Services) verwiesen. Die Beschreibung der Funktionalität im Kontext des Service-Modells wird in Abschnitt 3.2 vorgenommen und der Stand der Technik wird in Abschnitt 7.1 diskutiert.

<sup>3</sup>Weitere Beispiele und Probleme werden in Abschnitt 4.1 geschildert.

wie Services einer bestimmten Funktionalität gesucht werden, sondern wie von funktional äquivalenten Services der mit den besten Qualitätsmerkmalen ausgewählt wird. Beispielsweise wird angenommen, dass ein Requestor einen Service, der eine Überweisung von einem Konto auf ein anderes Konto vornimmt, von einem Service, der eine Lieferung von einem Ort an einen anderen Ort transportiert, unterscheiden kann. In gewisser Hinsicht ist die Funktionalität beider Services ähnlich, denn beide bewegen etwas von einem Ort zu einem anderen Ort: Im ersten Fall handelt es sich um Geld von einem Konto zum anderen und im zweiten Fall um den physischen Transport einer Ware. Die Unterscheidung der Funktionalität wird nicht durch die Qualitätsmerkmale ermöglicht, für die das Modell dieser Arbeit entwickelt wird. Es wird davon ausgegangen, dass ein Requestor die Funktionalität eines Services (im Beispiel Geldüberweisung beziehungsweise Warentransport) erkennt. Im Folgenden werden die Probleme und Auswirkungen der fehlenden Beschreibung der Qualitätsmerkmale aus den Sichtweisen eines Requestors und eines Providers beschrieben:

**Requestor** Eine umfassende und informierte Auswahl eines besten Services kann nicht automatisiert, sondern nur durch einen menschlichen Experten getroffen werden, wenn keine ausreichenden und allgemein akzeptierten Konzepte für Qualitätsmerkmale existieren<sup>4</sup>. Zusätzlich sind die Ausprägungen der Qualitätsmerkmale angebotener Services nicht direkt vergleichbar. Sind die Informationen des Brokers nicht ausreichend, muss der Experte zusätzliche Recherchen anstellen: Zum Beispiel müssen die Kosten einer Service-Nutzung manuell aus einer an anderer Stelle veröffentlichten Preisliste ermittelt werden, die Ausführungsdauer durch eine telefonische Nachfrage des Experten beim Provider oder aus der Historie der bisherigen Benutzungen ermittelt werden.

Um die Anforderungen des Requestors zu bestimmen, ist ebenfalls ein Experte mit Fachwissen auf dem jeweiligen Gebiet notwendig, der den genauen Verwendungszweck des Services kennt. Wenn ein bisher genutzter Service nun ausfällt und nicht mehr verfügbar ist, so muss erneut recherchiert werden um den zweitbesten Service zu bestimmen. Ebenso können zwischenzeitlich neu angebotene Services eine bessere Alternative darstellen, die es zu berücksichtigen gilt. Bei der Ermittlung einer Rangfolge sind verschiedene Faktoren abzuwägen und zu gewichten, wozu menschliche Experten benötigt werden. Eine derartige Lösung ist offensichtlich zeit- und kostenintensiv und widerspricht dem Ziel eines hohen Automatisierungsgrad einer serviceorientierten Architektur.

Werden Services vom Requestor in komplexere Geschäftsprozesse eingebunden, so beeinflussen Qualitätsmerkmale wie Ausführungsdauer und Ausfallsicherheit das Verhalten dieser Geschäftsprozesse. Es kann beispielsweise nicht im Voraus ermittelt werden, wie lange die Ausführung des Geschäftsprozesses dauert, wenn die Ausführungsdauer des Services nicht feststeht. Derartige Aussagen sind aber für die Entwicklung und das Management von Geschäftsprozessen und letzt-

---

<sup>4</sup>Eine Analyse der verfügbaren Ansätze wird in Kapitel 6, insbesondere Abschnitt 6.3, vorgestellt.

endlich für den unternehmerischen Erfolg von zentraler Bedeutung.

**Provider** Ein zentrales Problem für den Provider ist, dass er die Qualitätsmerkmale seiner Services nicht ausreichend spezifizieren kann. Es existiert keine ausreichende und allgemein akzeptierte konzeptionelle Grundlage, um die Besonderheiten der von ihm angebotenen Leistung zu dokumentieren<sup>5</sup>. Dadurch ergibt sich das Problem, dass ein Provider sich kaum von der Konkurrenz abgrenzen kann, da die Ausprägungen der Qualitätsmerkmale für einen Requestor nicht vergleichbar sind. In einer Menge funktional passender Services sind alle Services somit für einen Requestor als gleichwertig bezüglich ihrer Qualitätsmerkmale anzusehen. Dies ist nicht im Interesse des Providers.

Weiterhin ergibt sich das Problem, dass ein Provider die Ausprägungen der Qualitätsmerkmale seines Services unter Umständen nicht kennt und somit nicht angeben kann. Dieses Problem tritt dann auf, wenn bei der Leistungserbringung ein Geschäftsprozess ausgeführt wird, der wiederum Services anderer Provider benutzt. Sind Qualitätsmerkmale wie Zeitdauer und Kosten der aufgerufenen Services nicht bekannt, hat der Provider das Problem, dass er die Qualitätsmerkmale seines Services nicht ermitteln kann. Anders formuliert: Werden einzelne Services zu einem neuen Service zusammengesetzt, so hängen die Qualitätsmerkmale des neuen Services offensichtlich von den Qualitätsmerkmalen der benutzten Services ab.

**Zusammenfassung** In dieser Arbeit werden die folgenden Probleme betrachtet:

1. Es existiert keine angemessene und allgemein akzeptierte konzeptionelle Grundlage für die Beschreibung der spezifischen Eigenschaften, insbesondere Qualitätsmerkmale, eines Services. Dies hat die folgenden Auswirkungen:
  - (a) Qualitätsmerkmale sind bislang nicht ausreichend spezifizierbar. Die Positionierung eines Services durch den Provider ist daher aufwendig und es kann zu Missverständnissen kommen, beispielsweise aufgrund unvollständiger Angaben.
  - (b) Die Auswahl des am besten geeigneten Services wird nur unzureichend unterstützt. Insbesondere ist das Verfahren und das Ergebnis des Vergleichs zweier Services bezüglich ihrer Qualitätsmerkmale unklar und es kann daher keine Rangfolge der Services ermittelt werden, die den Anforderungen des Requestors entspricht.
2. Die Bestimmung der Qualität eines Geschäftsprozesses, der aus Services zusammengesetzt ist, wird konzeptionell nicht unterstützt. Insbesondere ist zu berücksichtigen, in welcher Weise die Qualitätsmerkmale der Services die Qualität des Geschäftsprozesses beeinflussen.

---

<sup>5</sup>Siehe auch Kapitel 6.



## 1.3 Lösungsansatz

Um zu entscheiden, welcher der verfügbaren Services tatsächlich genutzt werden soll, sind zusätzliche Informationen über die Services selbst notwendig, die als *Qualitätsmerkmale* bezeichnet werden. Die Ausprägungen der Qualitätsmerkmale werden mit der Service-Beschreibung veröffentlicht und können dann in der Anfrage eines Requestors berücksichtigt werden. In dieser Arbeit wird ein Modell derartiger Service-Eigenschaften erarbeitet und im Folgenden als *Qualitätsmodell* bezeichnet. Es werden dadurch die oben geschilderten Probleme wie folgt gelöst:

**1. Modellierung von Qualitätsmerkmalen für Services** Damit eine angemessene Modellierung der Eigenschaften möglich ist, sind zunächst die Rollen und Interaktionen einer serviceorientierten Architektur genauer zu analysieren. Dazu wird betrachtet, welche Schritte aus Sicht eines Providers notwendig sind, um einen Service verfügbar zu machen. Weiterhin wird betrachtet, welche Aktionen ein Requestor unternimmt, um einen Service nutzen zu können. Darauf aufbauend wird ein *Service-Modell* entwickelt, das die wesentlichen Elemente der Service-Orientierung und ihren Zusammenhang abbildet. Dadurch wird die Grundlage geschaffen, das Qualitätsmodell zu entwickeln.

Der Ansatz für das Qualitätsmodell und somit auch den Ansatz zur Problemlösung basiert auf der Abbildung der Ausprägungen der Qualitätsmerkmale. Diese Ausprägungen stellen Werte spezifischer Service-Eigenschaften dar. Es ergeben sich dadurch die folgenden Auswirkungen auf die in Abbildung 1.1 dargestellten Interaktionen:

- (a) **Publish:** Die Qualitätsmerkmale können definiert werden und deren Ausprägungen sind dem Broker vom Provider für jeden Service mitzuteilen. Damit kann sie der Broker für die automatisierte Beantwortung der Anfrage eines Requestor benutzen. Durch eine eindeutige Definition werden Missverständnisse und unvollständige Angaben vermieden.
- (b) **Find:** Vom Requestor sind notwendige Anforderungen an die Ausprägungen der Qualitätsmerkmale anzugeben und Wünsche zu formulieren (beispielsweise „möglichst geringe Kosten“). Es sollen also beispielsweise Budget-Restriktionen oder zeitliche Fristen in der Anfrage angegeben werden können, so dass der Broker den besten Service ermitteln kann. Durch eine eindeutige und einheitlicher Definition der Qualitätsmerkmale ist die Vergleichbarkeit gewährleistet und eine Rangfolge kann einer Anfrage entsprechend erstellt werden.

**2. Qualitätsmerkmale von Geschäftsprozessen** Eindeutig definierte Qualitätsmerkmale alleine reichen allerdings nicht aus, um die Ausprägungen der Qualitätsmerkmale für Geschäftsprozesse zu ermitteln, die Services aufrufen. Daher wird eine Anwendung des Qualitätsmodells vorgestellt, die über die Service-Auswahl

hinaus geht. Es werden Aggregationsmuster für Qualitätsmerkmale definiert, die neben den Aspekten des Qualitätsmodells den Kontrollfluss des Geschäftsprozesses berücksichtigen. Die Muster beschreiben, wie die Ausprägungen der Qualitätsmerkmale der einzelnen Services in einem Geschäftsprozess aggregiert werden können. Mit den Mustern können dann die Werte der Qualitätsmerkmale von Geschäftsprozessen aus den Werten der benutzten Services ermittelt werden.

**Auswirkungen** Für einen Provider ergibt sich dadurch die Möglichkeit einer besseren Positionierung der eigenen Services. Er kann die Eigenschaften seiner Services präziser und umfangreicher darstellen und sich somit von der Konkurrenz abheben. Gleichzeitig werden die Services unterschiedlicher Provider vergleichbar, wenn dieselben Definitionen der Qualitätsmerkmale benutzt werden. Für den Requestor ist die Entscheidung, welchen Service er tatsächlich nutzen möchte, effizienter zu treffen, da der Automatisierungsgrad erhöht werden kann. Eine Entscheidung für einen bestimmten Service ist nachvollziehbar, denn die Rangfolge der Services ist ermittelbar. Durch die Berücksichtigung der Qualitätsmerkmale kann der Requestor eine präzisere und konsistentere Entscheidung treffen. Zusammengefasst lässt sich feststellen, dass Angebot und Nachfrage besser und effizienter zusammengeführt werden und somit die Markttransparenz unterstützt wird [21, 53, 167].

**Einordnung des Qualitätsmodells** Das in dieser Arbeit entwickelte Qualitätsmodell, ist ein Meta-Modell, also ein *M2-Modell* aus Sicht der von der OMG definierten *Meta Object Facility* [105]. Diese Einordnung ist in Abbildung 1.2 visualisiert und wird im Folgenden verdeutlicht: Auf der untersten Ebene (M0) befinden sich die konkreten Werte: die Ausprägungen der Eigenschaften eines Services, beispielsweise die Werte für Kosten (200 €) und Ausführungsdauer (30 Minuten). Auf der nächst höheren Abstraktionsebene (M1) liegt das Modell dieser Qualitätsmerkmale. Beispielsweise wird hier modelliert, dass die Kosten als Zahlen mit zwei Nachkommastellen und in der Währung Euro (€) angegeben werden. Hier werden also Beschreibungen der Eigenschaften des Services spezifiziert. Darüber liegt das Qualitätsmodell dieser Arbeit, das die Grundlagen für alle Eigenschaften definiert, also ein Modell der Eigenschaftsbeschreibungen von Services. Das Qualitätsmodell beschreibt unter anderem, dass die Eigenschaften durch einen Namen identifizierbar sind, sowie einen Wert und eine Maßeinheit haben. Das Qualitätsmodell ist somit ein Meta-Modell und daher auf der Abstraktionsebene M2 anzusiedeln. Da ein Meta-Modell ebenfalls ein Modell ist, wird in dieser Arbeit zur besseren Lesbarkeit der Begriff Modell verwendet, wenn der Bezug zur jeweiligen Ebene eindeutig ist.

**Zusammenfassung** Der Lösungsansatz schafft eine konzeptionelle Grundlage, um für eine Menge von Services die relevanten Qualitätsmerkmale zu definieren. Für ein konkretes Anwendungsgebiet oder eine konkrete Branche können die dort

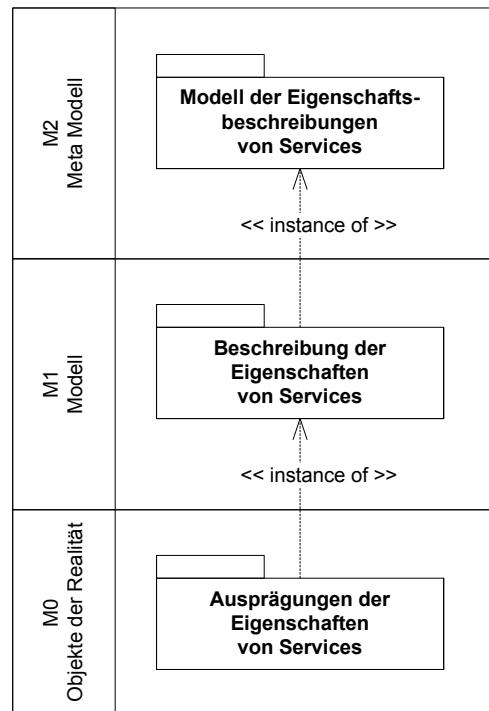


Abbildung 1.2: Einordnung des Modells

relevanten Qualitätsmerkmale von Branchenverbänden, Gremien oder anderen Interessengruppen als Standard spezifiziert werden. Je nach Anwendungsgebiet können die Qualitätsmerkmale sehr unterschiedlich sein und bei der Anwendung des Qualitätsmodells wird die Semantik der Qualitätsmerkmale und der Ausprägungen festgelegt.

Ein Provider kann somit für jeden anzubietenden Service und für jede der definierten Qualitätsmerkmale die konkreten Ausprägungen eindeutig und unmissverständlich angeben. Dadurch kann ein Requestor den für seine Anforderungen am besten geeigneten Service auswählen. Dazu ist eine Vergleichbarkeit der Services hinsichtlich der Qualitätsmerkmale notwendig. Die konkreten Werte bilden letztendlich die Grundlage für die Entscheidung, welcher der Services ausgewählt wird. Es muss zum Beispiel entscheidbar sein, welche Werte besser als andere oder äquivalent zu anderen sind. Der Vergleich der Services und die Auswahl des besten Service kann durch die konzeptionelle Grundlage automatisiert getroffen werden. Es werden nur die für den jeweiligen Zweck relevanten Aspekte betrachtet und somit verhält sich das Modell getreu dem Gedanken von Einsteins Aussage zu Beginn dieser Arbeit.

## 1.4 Wissenschaftlicher Beitrag und Aufbau der Arbeit

Generell ist bei der wissenschaftlichen Entwicklung von konzeptionellen Einheiten, wie zum Beispiel Modellen, Prozeduren oder Theorien, eine sorgfältige und vollständige Analyse und Erklärung der Annahmen bezüglich der Beobachtungen in der realen Welt notwendig. Hierbei ist insbesondere eine Abgrenzung dessen notwendig, was betrachtet werden soll; dies wird als Bildung der Diskurswelt bezeichnet. Darüber hinaus ist zu erklären, wie mit Annahmen umgegangen wird und welche Auswirkungen diese haben. Die zugrunde liegenden Annahmen und Auffassungen sind ebenso von zentraler Bedeutung für das Verständnis des Modells, wie eine kritische Stellungnahme zu den getroffenen Annahmen. [133]

**Wissenschaftlicher Beitrag** Der Beitrag dieser Arbeit ist die Formulierung, Untersuchung und Validierung eines Qualitätsmodells, das auf einem Service-Modell basiert. Es wird die Auswirkung eines Qualitätsmodells für eine serviceorientierte Architektur untersucht, wobei zum einen die Service-Auswahl und zum anderen die Modellierung und Optimierung von Geschäftsprozessen betrachtet werden. Dazu werden unter anderem Aggregationsmuster für Qualitätsmerkmale entwickelt. Die Validierung der Arbeit erfolgt durch zwei Maßnahmen: Zum einen wird die Benutzbarkeit durch eine Anwendung des Qualitätsmodells und der Aggregationsmuster in der Praxis aufgezeigt; zum anderen wird die Realisierung des Qualitätsmodells und der Service-Anfragen durch eine prototypische, Web-basierte Anwendung auf einem relationalen Datenmodell vorgestellt. Zur Eingrenzung der Diskurswelt sei festgehalten, dass die Modelle Konzepte eines IT-Systems beschreiben, das eine serviceorientierte Architektur realisiert. Zur weiteren Eingrenzung sei die Spezifikation der Funktionalität der Services aus der Betrachtung ausgeschlossen. Bei der Untersuchung der Modelle werden die Annahmen und Anforderungen diskutiert, sowie mögliche Anwendungen und Auswirkungen betrachtet.

Es wird in dieser Arbeit ein Ansatz des Konstruktivismus [49] verfolgt, wie es in der wissenschaftlichen Beschäftigung mit informationstechnischen Systemen häufig anzutreffen ist [188]. Beim Konstruktivismus basiert die Konstruktion der Modelle auf einer nachvollziehbaren Konstruktion von Begriffen und somit der Bildung einer gemeinsamen Terminologie [133]. Diese Vorgehensweise wird bei der Konstruktion der Modelle verfolgt.

**Aufbau der Arbeit** Zu Beginn der Arbeit werden die relevanten Entitäten einer serviceorientierten Architektur identifiziert, formalisiert und in Beziehung gesetzt. Ausgehend von der historischen Entwicklung und der Definition der zentralen Begriffe, werden die Rollen der Teilnehmer identifiziert. Deren Verhalten wird geschildert und mit einer aktuellen Realisierung durch Web Services-Technologie illustriert (Kapitel 2).

Nach den Grundlagen wird das Service-Modell formuliert, indem einige der Annahmen zusammengefasst, sowie die zentralen Elemente identifiziert und in

Beziehung gesetzt werden. Realisierbare Vorteile werden erarbeitet und die Prinzipien der Service-Orientierung formuliert. Anschließend kann das Paradigma der Service-Orientierung formuliert werden, das sowohl eine Umsetzung des Service-Modells als auch die Prinzipien der Service-Orientierung umfasst. Eine Betrachtung der mittels Web Services-Technologie umsetzbaren Aspekte schließt das Kapitel. An einzelnen Stellen des Kapitels wird bereits auf die Notwendigkeit und die realisierbaren Vorteile eines Qualitätsmodells hingewiesen (Kapitel 3).

Das Service-Modell dient als Basis für die Formulierung des Meta-Modells der Service-Eigenschaften, das als Qualitätsmodell bezeichnet wird. Es werden zwei Szenarien für Services und deren Qualitätsmerkmale geschildert, welche die Vielseitigkeit der Qualitätsmerkmale und die Anforderungen an das Meta-Modell illustrieren. Zusätzlich werden Anforderungen anhand von typischen Anwendungsfällen formuliert. Anschließend wird das Qualitätsmodell konstruiert. Zur Validierung wird eine Umsetzung anhand der Szenarien exemplarisch beschrieben und eine prototypische, Web-basierte Anwendung zur Service-Auswahl vorgestellt, die auf einer Umsetzung des Qualitätsmodells als relationales Datenmodell aufsetzt. Abschließend werden Erweiterungsmöglichkeiten diskutiert (Kapitel 4).

Die Bedeutung des Qualitätsmodells für Geschäftsprozesse und insbesondere deren Modellierung wird in Kapitel 5 aufgezeigt. Zunächst wird der Anwendungsfall der Service-Auswahl untersucht und die Auswirkungen eines Qualitätsmodells betrachtet. Darüber hinaus wird untersucht, wie Qualitätsmerkmale von Geschäftsprozessen ermittelt werden können, die Services aufrufen. Dazu werden Aggregationsmuster für Qualitätsmerkmale formuliert, die auf Geschäftsprozesse angewendet werden können. Hier werden insbesondere die Vorteile des Qualitätsmodells erarbeitet. Zusätzlich wird ein praktischer Einsatz des Qualitätsmodells in einem Projekt vorgestellt, in dem Qualitätsmerkmale für einen Order-To-Cash-Prozess modelliert und aggregiert worden sind.

Bei der Betrachtung der verwandten Arbeiten wird zunächst eine Einordnung des Qualitätsmodells vorgenommen. Hierzu werden ausgewählte konzeptionelle Arbeiten über serviceorientierte Architekturen, sowie weitere Literatur zu Qualitätsmodellen betrachtet. Es existieren verschiedene Ansätze Qualitätsmerkmale für bestimmte Zwecke abzubilden und zu definieren. Das Spektrum der Modelle reicht von betriebswirtschaftlichen Ansätzen, über Modellen für Rechnernetzen, bis hin zu Modellen für die Software-Entwicklung. Weiterhin werden vor allem Ansätze für Service-Eigenschaften betrachtet, die speziell für serviceorientierte Architekturen und im Kontext der Web Service-Technologie entwickelt wurden. Die verwandten Arbeiten werden kritisch betrachtet und eingeordnet. Hierbei werden Übereinstimmungen und Abweichungen untersucht, die sowohl das jeweilige Service-Modell als auch das Qualitätsmodell betreffen. Zum Abschluss des Kapitels wird auf Qualitätsmodelle für Geschäftsprozesse und die automatisierte Optimierung von Geschäftsprozessen eingegangen. (Kapitel 6).

In der Diskussion werden die Annahmen der Modelle betrachtet, sowie alternative Konzepte und mögliche Erweiterungen vorgestellt (Kapitel 7). Diese Arbeit schließt mit einer Zusammenfassung (Kapitel 8).

## Kapitel 2

# Grundlagen der Service-Orientierung

In diesem Kapitel wird ein erster Überblick über die Service-Orientierung gegeben. Zunächst werden sowohl die historische Entwicklung in Abschnitt 2.1 beschrieben, als auch die Interaktionen und Aufgaben der Teilnehmer in Abschnitt 2.2 illustriert. Die vorgestellten Konzepte entsprechen dem aktuellen Stand der Technik und der Sichtweise der Literatur. Abschnitt 2.3 illustriert eine bedeutende technische Realisierung zentraler Konzepte der Service-Orientierung durch die Web Service Technologie. Das Kapitel schließt mit einer Zusammenfassung in Abschnitt 2.4, welche die wesentlichen Punkte für das in Kapitel 3 zu erarbeitende Service-Modell herausstellt.

### 2.1 Hintergrund

Der Begriff Service-Orientierung ist zurzeit sehr populär und wird in unterschiedlichen Kontexten der Informatik und darüber hinaus verwendet. Die Semantik dieses Begriffes ist selten klar definiert. Zusätzlich werden die Begriffe Service und Service-Orientierung mit weiteren Begriffen aus unterschiedlichen Bereichen kombiniert, wie beispielsweise *Architektur*, *Paradigma*, *Umgebung* oder (im Englischen) *Engineering* und *Computing*. In diesen unterschiedlichen Kombinationen wird der Begriff Service-Orientierung zurzeit in vielen wissenschaftlichen Veröffentlichungen und populären Medien, sowie als Thema für Konferenzen und neue Technologien verwendet. Hierbei liegt oft ein unterschiedlicher Fokus der Service-Orientierung zugrunde.

Die inflationäre Verwendung des Begriffs Service-Orientierung kann zu Missverständnissen führen [129]. Es existieren allerdings verschiedene Organisationen, die Spezifikationen und Glossars veröffentlichen, wie beispielsweise das World Wide Web Consortium (W3C) [174], die Organization for the Advancement of Structured Information Standards (OASIS) [110] und die Web Services Interoperability Organization (WS-I) [162]. Dennoch werden vielfach abweichende Defi-

nitionen verwendet; in [139] wird ein Überblick gegeben. Daraus ergibt sich ein Potential für Missverständnisse und Frustrationen bei Anwendern und Forschern gleichermaßen [22, 58, 129, 155]. Dennoch kann die aktuelle Popularität der Service-Orientierung in unterschiedlichen Kontexten als ein Indiz für neue Inspiration und neue Anwendungsmöglichkeiten gewertet werden. Im Folgenden wird nun herausgestellt, dass hinter der Service-Orientierung eine viel versprechende Idee steht, die nicht nur wirtschaftliche Trends abbildet, sondern auch einen fundamentalen Wandel in der Informationstechnologie darstellt.

### 2.1.1 Wirtschaftliche Entwicklung

Die wirtschaftliche Realität wird von der Marktwirtschaft [53] beherrscht [21]. Dabei handelt es sich um die Organisationsform eines arbeitsteiligen Wirtschaftssystems. Eines ihrer grundlegenden Prinzipien ist das Zusammenspiel von Angebot und Nachfrage auf freien Märkten [41, 167]. Dieses Prinzip findet sich als Erklärungsmodell an vielen Stellen der Wirtschaftslehre wieder, so zum Beispiel bei der Ermittlung eines Preises, Optimierungen der Produktionsmengen und der Produktionsvielfalt [41, 167]. Auf Märkten werden Preise für Produkte und Dienstleistungen durch Angebot und Nachfrage bestimmt. Dies geschieht auf der Basis von Geld als allgemein akzeptiertem Tauschmittel, wodurch der indirekte Austausch von Produkten und Dienstleistungen erst möglich wird und somit letztendlich auch eine Service-Orientierung ermöglicht wird. Maßgeblich für die heutige Ausprägung der Marktwirtschaft war vor allem die industrielle Revolution, also die Massenproduktion von Handelswaren und der eng damit verknüpfte Taylorismus, der die Arbeitsteilung und Spezialisierung der Arbeiter hervorbrachte [1].

Ein innovatives Produkt wurde oftmals von Spezialisten für einen Kunden entwickelt und hergestellt. Wenn dieses neue Produkt erfolgreich war, wurde es auch anderen Kunden angeboten. Zur Einführung neuer Produkte sind Techniken entwickelt und eingeführt worden, um potentielle Kundengruppen, Zielgruppen des Marketings und ähnliche Gruppierungen zu erkennen [9]. Basierend auf den neu gewonnenen Erkenntnissen wird ein (neues) Produkt angeboten, um möglichst viele Kunden zu gewinnen. Dies wiederum bewirkt bei der Konkurrenz, ähnliche Produkte in lukrativen Marktsegmenten anzubieten. Für diese neuen Anbieter ist es dabei notwendig, sich geeignet von den Marktführern zu unterscheiden, beispielsweise durch niedrigere Preise oder bessere Qualität. Zusätzlich lassen sich leicht veränderte oder gar individuell angepasste Produkte anbieten. Neben der Abgrenzung von der Konkurrenz soll dadurch auch eine verstärkte Kundenbindung erreicht werden. Bei diesem Verhalten im Markt steht immer der Kunde beziehungsweise die Kundengruppen im Zentrum der Betrachtung und der Optimierungsmaßnahmen. [1, 9]

In der Service-Orientierung rückt der Service in den Mittelpunkt. Neben einer erweiterten Fokussierung auf potentielle Kunden, wird auch das interne Service-Management des Providers sowie die Konkurrenz beachtet [32, 36, 79]. Ein Service wird so entwickelt, verwaltet und zur Verfügung gestellt, dass er von Kunden und

Geschäftspartnern benutzt und in deren Geschäftsprozesse integriert werden kann. Gleichzeitig kann ein Anbieter auch Services von Dritten in seine Geschäftsprozesse integrieren. Die Auswirkungen dieser Entwicklung sind deutlich erkennbar an zunehmendem e-Business [149], an wachsendem Outsourcing und der Globalisierung. Ziel ist es, die Kernkompetenzen eines Unternehmens zu erkennen und sich darauf zu konzentrieren. Diese Kompetenzen können als Geschäftskomponenten definiert werden. Gleichzeitig wird ein Netzwerk kooperierender Partner aufgebaut, die ebenfalls ihre Kernkompetenzen als Geschäftskomponenten anbieten. Die Informationstechnologie, insbesondere aktuelle, effiziente Kommunikationstechniken, ermöglichen prinzipiell eine Verknüpfung derartiger Geschäftskomponenten. Allerdings ist die Kommunikation alleine nicht ausreichend. Ein grundsätzliches Verständnis der Komponenten untereinander muss erreicht werden. Somit kann zwar schnell und einfach kommuniziert werden, aber eine Integration der Geschäftskomponenten ist eine komplexere Aufgabe und deren Umsetzung ist bislang nur unzureichend gelöst. [34, 86, 149] Im Folgenden wird nun betrachtet, wie sich die Informationstechnologie und die angesprochene Kommunikation entwickelt hat.

### 2.1.2 Entwicklung der Informationstechnologie

In der Informationstechnologie und deren wissenschaftlichen Forschungsgebieten gewinnt die Service-Orientierung ebenfalls zunehmend an Bedeutung [32, 84, 115, 139, 161]. Es finden sich hier Unterschiede in den Auffassungen der Service-Orientierung. Die dortigen Unterschiede lassen sich durch die Forschungsbereiche und deren Hintergrund und Forschungsgegenstand begründen [139]. So werden beispielsweise im Forschungsgebiet der Netzwerktechnologie Services durch Bandbreite, Verfügbarkeit, Fehlerrate und ähnliches *definiert*<sup>1</sup>. Die Funktionalität des Services ist dabei immer dieselbe, nämlich eine Netzwerkverbindung zur Verfügung zu stellen. In der Telekommunikation versteht man unter einem Service eine Fähigkeit (Anrufweiterleitung, Anrufer-Identifikation etc.) oder einen Verbindungsdienst an sich. In der Systemtechnik und in der Anwendungsentwicklung stellt ein Service eine eigenständige Funktion dar (beispielsweise Abrechnung, Lagerverwaltung oder Personalverwaltung in einem ERP-System) und wird oft als das System unterstützend angesehen. Services werden hier als kombinierbare Module angesehen, die von vergleichsweise grober Granularität sind.

In der Informatik-Literatur wird die Service-Orientierung als konsequente Weiterentwicklung bereits erprobter und erfolgreich angewandter Techniken bezeichnet. Dazu zählen Prinzipien und Konzepte, die einzeln betrachtet nicht neu sind, wie beispielsweise Kapselung, Schnittstellen und nachrichtenbasierte Kommunikation und das Konzept eines Object Request Brokers. Es wird somit auf die logische Nähe zur Objektorientierung und zu dem Prinzip der Software-Komponenten verwiesen [139]. Im Folgenden wird daher auf die vorausgehenden Entwicklungen

---

<sup>1</sup>Aus diesem Bereich stammen auch die Begriffe *Quality of Service* (QoS) und *Service Level Agreement* (SLA), die in Abschnitt 4.3 genauer erläutert werden [146].



in der Informatik und insbesondere verwandter Programmierparadigmen eingegangen. [79]

**Entwicklung der Integrationstechniken** Im Laufe der letzten Jahrzehnte hat die Informatik in immer mehr Bereichen eines Unternehmens Einzug gehalten, wobei zunächst unterschiedliche Lösungen für spezielle Aufgaben entstanden, zum Beispiel für die Finanzbuchhaltung, Auftragsverwaltung oder Kundenbetreuung. Diese Insellösungen sind immer weiter integriert worden und es hat sich eine IT-Landschaft gebildet, die aus vielen heterogenen Systemen besteht. In dieser IT-Landschaft wurden die Geschäftsprozesse eines Unternehmens ausgeführt, die dabei auf immer mehr unterschiedliche Systeme zugriffen. Der erste Schritt, diese Insellösungen zur Performanzsteigerung der Geschäftsprozesse zu integrieren bestand darin, den Datenaustausch zwischen je zwei Teilsystemen (Komponenten) zu automatisieren. Dazu wurden spezielle Adapter für die jeweiligen Komponenten entwickelt. Dieses geschah teilweise in aufwendiger, von Spezialisten durchgeführter Einzelarbeit. Die entwickelte Lösung passte genau für die ursprüngliche Kommunikation zwischen den beiden Komponenten. In der Literatur wird eine solche Lösung als *Bridge* bezeichnet [54] und stellt einen intuitiven Schritt zur Integration dar.

Die eingesetzte Technologie zur Integration reifte im weiteren Verlauf: Ein einfaches Beispiel ist der Einsatz von Nachrichten-Zwischenspeichern für die Kommunikation, so genannten *Message Queues*. Es wurde damit eine bessere Qualität der Integration erreicht, in dem sichergestellt werden konnte, dass der Datenaustausch auch bei einseitigen Kommunikationsproblemen stattfand und die Reihenfolge der Daten korrekt blieb. In der folgenden Entwicklung wurden die Message Queues erweitert, so dass mit ihnen genauere Regeln für den Datenaustausch und die ansprechbaren Komponenten definierbar wurden. Eine Nachricht wird an einen *Message Broker* geschickt, der dafür sorgt, dass die relevanten Daten an Kommunikationspartner weitergeleitet werden. Diesen Broker kann man als Zwischenschritt zur nächsten Stufe der Integration ansehen, den *Application Server* darstellen. Mit ihnen ist ein erweitertes Management der Ressourcen möglich. Ebenfalls bieten Application Server oftmals die Möglichkeit, Teile der Geschäftsprozesse als Workflow abzubilden. Zusätzlich ist mit einem Application Server auch die Integration mit Datenbankmanagementsystemen möglich. [27, 86, 139]

Im Zuge dieser immer umfassenderen Integration sämtlicher Anwendungen, Datenbanksysteme und Workflow-Management-Systeme stellt die Service-Orientierung einen nahe liegenden Entwicklungsschritt dar. Mit der aktuellen Umsetzung, Web Services, ist eine einheitliche Kommunikationsmöglichkeit unter Benutzung der allgegenwärtigen Internet-Technologie verfügbar. Bislang wurde die Anbindung der einzelnen Komponenten an einen Application Server über die jeweils auf beiden Seiten verfügbaren Kommunikationsschnittstellen, beispielsweise eine Programmierschnittstelle, implementiert. Mit Web Services ist eine einheitliche und relativ einfache Lösung verfügbar, die auf den Austausch von Dokumenten in der

Extensible Markup Language (XML) [23] über Internet-Technologie basiert [32]. Dadurch wird eine serviceorientierte Architektur realisiert.

**Entwicklung kommerzieller Software-Systeme** Zur Entwicklung und zur Popularität der Service-Orientierung tragen vor allem auch Software-Hersteller aus verschiedenen Marktsegmenten bei. Sie preisen ihre Software-Systeme mit serviceorientierten Erweiterungen und Funktionen an, beispielsweise bei ERP-Systemen (englisch Enterprise Resource Planning), Datenbanksystemen und Middleware-Komponenten [58, 81]. Generell wird Service-Orientierung auch von den Herstellern kommerzieller Software als umfassendes Konzept verstanden, welches die bestehende Software-Landschaft erweitert, überarbeitet oder zum Austausch einzelner Komponenten führt [36]. Letztendlich stehen monetäre Interessen der Hersteller im Vordergrund, also neue Produkte oder Anpassungen und Erweiterungen bestehender Systeme zu verkaufen. So weist Jason Bloomberg in [20] darauf hin, dass viele Software-Hersteller ihre Produkte als Umsetzung einer serviceorientierten Architektur anpreisen. Dabei steht die Möglichkeit im Vordergrund, durch Erweiterungen bestehender Produkte neue Fähigkeiten und Leistungen zu verkaufen. Es muss allerdings unterschieden werden, ob die Produkte selbst serviceorientiert sind, also intern die Prinzipien der Service-Orientierung umsetzen, oder ob sie einer Organisation dabei helfen kann, die Software-Landschaft serviceorientiert zu gestalten [20]. Auch hier zeigt sich, dass die Begriffe nicht einheitlich verwendet werden und die Definitionen unscharf sind. Generell lässt sich erkennen, dass die Popularität der Service-Orientierung ausgenutzt und jeweils passend interpretiert wird.

Insbesondere von Software-Herstellern wird die Entwicklung von Standards als zentraler Faktor für den Erfolg der Service-Orientierung angesehen. Die Notwendigkeit lässt sich aus den grundlegenden Rollen und ihrer Interaktionen ableiten. Als wichtig werden daher vor allem die Verfügbarkeit standardisierter Service-Beschreibungssprachen und Kommunikationsprotokolle und ein gemeinsames und einheitliches Verständnis der Suche und der Verwaltung der Services (respektive deren Beschreibungen) durch einen Service Brokers. Diese Notwendigkeit wird inzwischen von den meisten Herstellern kommerzieller Software anerkannt, so dass sie sich in unterschiedlichen Gremien und Organisationen mit der gemeinsamen Spezifikation von Protokollen beschäftigen. Auch die wirtschaftliche Entwicklung und die Anforderungen der Kunden zu mehr Integrationsmöglichkeiten mit anderen Unternehmen fördern diese Entwicklung. Da das Prinzip der Service-Orientierung unternehmensübergreifend ausgerichtet ist, verfolgen einige Hersteller das Ziel, ihre Produkte mit denen anderer Hersteller kompatibel zu machen. Dieser Trend steht im diametralen Gegensatz zu vorherigen Versuchen, eigene, proprietäre Ansätze zu verbreiten. Dieser Sinneswandel resultiert sowohl aus wachsendem Kundendruck, als auch aus der Entwicklung der Integrationstechniken, die bereits erwähnt wurden. Hervorzuheben seien an dieser Stelle die aktuellen Spezifikationen des W3C und der OASIS, wie beispielsweise SOAP [175] zum

Austausch von XML-basierten Nachrichten, Web Services Description Language (WSDL, [180]) zur Service-Beschreibung und Universal Description Discovery & Integration (UDDI) Spezifikation für Service Broker [35]. Diese Spezifikationen stellen den De-facto-Standard für die zurzeit populärste Umsetzung der Service-Orientierung dar: Web Services [4, 61, 139]. Im Abschnitt 2.3 wird diese Umsetzung genauer betrachtet und ausführlich vorgestellt.

### 2.1.3 Zusammenfassung

Es lassen sich zwei wesentliche Trends in der Informationstechnologie erkennen:

1. Es wird eine immer mächtigere Technik zur Integration von Software-Komponenten entwickelt und eingesetzt.
2. Die großen IT-Konzerne und Software-Hersteller etablieren De-facto-Standards und forcieren Techniken für eine einfache und praktikable Integration.

Diese Trends der IT spiegeln die allgemeine wirtschaftliche Entwicklung des gesteigerten Integrationsbedarfs wider.

## 2.2 Serviceorientierte Architektur

In diesem Abschnitt werden nun die Aufgaben und Interaktionen der Rollen einer serviceorientierten Architektur ausführlich beschrieben und die zentralen Begriffe definiert. Die Begriffsdefinitionen und die verwendete Terminologie sind angelehnt an einige ausgewählte Literatur-Quellen, die hier einleitend kurz beschrieben werden sollen.

1. Das World Wide Web Consortium (W3C) [174] setzt insbesondere im Web Services Bereich viele Maßstäbe, spezifiziert De-facto-Standards und bildet zu aktuellen Fragestellungen kompetente technische Komitees [61, 139]. Aus den vielen Spezifikationen des W3C sind für das Service-Modell besonders das *Web Services Glossary* [179] und die *Web Service Architecture* [178] hervorzuheben.
2. Die Spezifikationen der Web Services Interoperability (WS-I) definieren, wie die Interoperabilität von Web Services sichergestellt werden soll. Die Terminologie ist eng an die W3C Terminologie angelehnt, aber durch Beschreibungen, wie Web Services zu benutzen sind, sind einige Konzepte detaillierter definiert; beispielsweise werden die erlaubten Inhalte eines WSDL-Dokumentes und der Ablauf der Kommunikation genauer beschrieben. [10]
3. Als drittes Standardisierungsgremium entwickelt die Organization for the Advancement of Structured Information Standards (OASIS [110]) zurzeit ein Referenzmodell für serviceorientierte Architekturen, in dem wesentliche

Begriffe definiert werden. Hier findet auch eine Abgrenzung statt, was nicht Bestandteil einer SOA ist [94]. Zusätzlich werden mit UDDI [35] und WS-BPEL [5] wesentliche Spezifikationen von der OASIS vorangetrieben.

4. Das Buch „Web Services - Concepts, Architectures and Applications“ von Gustavo Alonso et alii [4] konzentriert sich auf die technologische Umsetzung einer serviceorientierten Architektur und die existierenden Standards. Es motiviert aus der historischen Entwicklung der Software-Architekturen die Entstehung der Service-Orientierung. Die unterschiedlichen Aspekte einer serviceorientierten Architektur werden jeweils anhand existierender Spezifikationen beschrieben.
5. Munindar P. Singh und Michael N. Huhns' Buch „Service-oriented Computing - Semantics, Processes, Agents“ [139] geht über die technologische Realisierung hinaus und beschreibt wesentliche Grundlagen der Anwendung und die Auswirkungen auf eine Unternehmung. Es werden dabei die Konzepte der Service-Orientierung deutlich herausgearbeitet. Web Services werden dennoch auch hier als mögliche Technologie zur Realisierung beschrieben.

Die Quellen 1., 2. und 3. sind relevant, da sie De-facto-Standards spezifizieren und erläutern, mit der sich serviceorientierte Architekturen realisieren lassen. Dieses wird in 4. dann detailliert und zusammenhängend beschrieben. Die zentrale Quelle dieser Arbeit für die Konzepte und Prinzipien der Service-Orientierung ist 5. Dennoch gibt es nicht nur in diesen Quellen, sondern allgemein in der Literatur unterschiedliche Auffassungen darüber, was ein Service sein kann. Es wird teilweise nicht klar definiert, was genau ein Service ist, so dass es zu Missverständnissen kommt. In diesem Abschnitt sollen daher die zentralen Begriffe definiert werden.

Wie bereits erwähnt wurde, ist der Gedanke der Service-Orientierung nicht auf IT-Systeme beschränkt. Dennoch bilden IT-Systeme die Basis für eine effiziente und effektive Realisierung der grundlegenden Konzepte. Mit Hilfe einer effizienten Kommunikationstechnik entsteht eine komplexe, verteilte Umgebung, in der sich die essentiellen Verhaltensweisen und Eigenschaften realisieren lassen, die bereits in Abbildung 1.1 dargestellt wurden [116]. Dann sind verschiedene Services verfügbar und können von allen Teilnehmern genutzt werden. Es wird somit eine serviceorientierte Architektur realisiert, die als geeignet angesehen wird, die Anforderungen moderner, flexibler Unternehmen in einem hochdynamischen Umfeld zu erfüllen [74, 96, 139]. Der englische Begriff *Service Oriented Architecture* (SOA) wurde von Steve Burbeck als einem der Ersten in [25] umfassend beschrieben. Im Folgenden wird herausgearbeitet, dass eine serviceorientierte Architektur eine Software-Architektur ist, die durch ein entsprechendes IT-System beziehungsweise Software-System umgesetzt werden kann. Dazu sind zunächst einige Begriffsdefinitionen notwendig.

### 2.2.1 Begriffsdefinitionen

Der Begriff System stammt etymologisch vom griechischen *systema*, welches unter anderem „das Gebilde“ und „das Zusammengestellte“ bedeutet. Somit besteht ein System aus mehreren Einzelteilen. Wesentlich ist dabei, dass diese Teile in Beziehung stehen und eine Einheit darstellen. Ein System ist durch zweckmäßig definierte Systemgrenzen von anderen Systemen, seiner Umwelt, abgegrenzt. Bezogen auf die Kommunikation mit dieser Umwelt gibt es geschlossene und offene Systeme. Daher sei der Begriff System hier wie folgt definiert. [67]

**Definition 1 System:** Ein System besteht aus mehreren interagierenden Einzelteilen und hat wohldefinierte Grenzen.

Die Informatik beschäftigt sich mit informationstechnischen Systemen (IT-System). Mit einem IT-System ist in der Regel ein Software-System gemeint, das auf einer Hardware läuft. Im Folgenden wird eine existierende Hardware vorausgesetzt und insofern nicht weiter betrachtet.

**Definition 2 Software-System:** Ein Software-System ist ein System, dessen Einzelteile Software sind.

Die Einzelteile eines Systems werden, insbesondere bei Software-Systemen, üblicherweise als Komponente bezeichnet. Eine Komponente grenzt sich durch bestimmte Merkmale von den anderen Teilen des Systems ab. Die einzelnen Komponenten eines (Software-) Systems sind somit bis zu einem gewissen Grad unabhängig voneinander. Das Wort Komponente ist etymologisch abgeleitet vom lateinischen *componere*, das „zusammensetzen“ bedeutet, bzw. *componendum* („das Zusammensetzende“). In der Software-Entwicklung impliziert eine Komponente die Möglichkeit, mehrere Komponenten zusammensetzen beziehungsweise eine Komponente in anderen Software-Systemen wiederverwenden zu können.

**Definition 3 Komponente:** Eine Komponente ist ein Teil eines Software-Systems und hat wohldefinierte Grenzen, die es von anderen Komponenten abgrenzt.

Eine Komponente stellt ein unabhängiges Softwaremodul dar. Die Grenzen einer Komponente werden durch eine Schnittstelle definiert. Die Schnittstelle einer Software-Komponente stellt eine Beschreibung der Kommunikationsverbindung nach außen dar. Nun kann der Begriff Software-Architektur definiert werden.

**Definition 4 Software-Architektur:** Eine Software-Architektur ist eine Beschreibung der strukturierten Anordnung von Komponenten eines Software-Systems sowie ihrer Beziehungen [11]. Zu einer Architektur gehören auch Prinzipien, die den Entwurf und die Entwicklung des Systems bestimmen [67].

Als nächster Schritt ist der Begriff Service zu definieren. Ausgangspunkt ist hier der bereits informal verwendete Begriff *Leistung*: In der ursprünglichen Wortbedeutung ist eine Leistung eine gezielte Handlung, die zu einem bestimmten Ergebnis beziehungsweise der Lösung einer Aufgabe führt. Hier soll eine Leistung angesehen werden als ein Verhalten. Ein Verhalten kann dabei eine aktive Handlung, eine Duldung oder das Unterlassen einer Handlung sein. Somit kann eine Leistung zum Beispiel eine Produktion, eine Warenlieferung, eine Dienstleistung, ein Darlehen, Werbung, Vermittlung oder Vermietung sein. Mit einer Leistung wird in der Regel ein wirtschaftlicher Erfolg [167] angestrebt, also kann die Leistung als Gegenstand des Wirtschaftsverkehrs angesehen werden.

**Definition 5 Leistung:** Eine Leistung ist ein Verhalten, das Gegenstand des Wirtschaftsverkehrs ist.

Der Provider bietet an, eine Leistung für jemanden Anderen zu erbringen. Das Ziel eines Requestors ist also, dass ein Provider eine Leistung für ihn erbringt. Die Leistung ist konkret: Auf das Transport-Beispiel bezogen wird das Transportgut  $g_1$  vom Ausgangsort  $ort_1$  abgeholt und zum Zielort  $ort_2$  transportiert.

Die Leistung eines Services wird erbracht, wenn er ausgeführt wird, indem *Ressourcen* eingesetzt werden. Allgemein bezeichnet eine Ressource ein zur Lösung einer Aufgabe benötigtes Mittel. In der Wirtschaftslehre wird eine Ressource als ein sich verbrauchendes, begrenzt zur Verfügung stehendes Gut, oder eine Fähigkeit definiert [53]. Ebenfalls wird die menschliche Arbeitskraft als Ressource angesehen [21, 167]. Eine ähnliche Auffassung wird auch im Kontext von Workflow-Management vertreten [159]. In anderen Disziplinen der Informatik wird als Ressource oftmals ein Hardware-System beziehungsweise die auf einer Hardware laufende Software angesehen. Im Kontext von Web Service wird eine Ressource durch einen URI [15] eindeutig identifiziert. In dieser Arbeit stellen Ressourcen eine Grenze der Diskurswelt dar. Sie werden vom Provider identifiziert (siehe Abschnitt 2.2.2) und können zur Leistungserbringung eingesetzt werden. Somit wird eine Ressource wie folgt definiert:

**Definition 6 Ressource:** Eine Ressource ist ein verfügbares Gut oder eine Fähigkeit, die eingesetzt werden kann, um eine Leistung zu erbringen.

Der Provider ist also dafür verantwortlich, dass Ressourcen eingesetzt werden, wenn ein Service aufgerufen wird. Beispielsweise können im Transport-Beispiel der LKW  $L_1$  und der Fahrer  $F_1$  eingesetzt werden, um eine konkrete Transportleistung zu erbringen, das Transportgut  $g_1$  vom  $ort_1$  abzuholen und zum  $ort_2$  zu transportieren (siehe oben). Die Auswahl konkreter Ressource ist nicht durch einen Requestor durchzuführen, sondern wird vom Provider vorgenommen (siehe Abschnitt 2.2.2). Diese Überlegungen führen schließlich zu der folgenden Definition eines Services:

**Definition 7 Service:** Ein Service ist eine angebotene Leistung, für die folgende Bedingungen gelten:

1. Das Angebot erfolgt in einem Software-System.
2. Das Angebot beinhaltet die Veröffentlichung einer Beschreibung der Leistung.

Die Beschreibung der Leistung wird als *Service-Beschreibung* bezeichnet. ◇

Aufbauend auf dieser Definition werden die folgenden Begriffe definiert: Der Anbieter eines Services wird *Service Provider* genannt. Ein *Service Broker* bezeichnet die Rolle desjenigen, der die Beschreibungen mehrerer Services verwaltet. Der *Service Requestor* ist derjenige, der einen Service aufruft.

Folgende Verben, die Aktionen repräsentieren, sind im Zusammenhang mit dem Begriff Service zu betrachten: Eine Service-Beschreibung wird *veröffentlicht*, anschließend kann die Beschreibung *gefunden* werden und man kann einen Service *nutzen*. In der Literatur wird teilweise der Begriff „Beschreibung“ zur Vereinfachung weggelassen. So wird zum Beispiel die Formulierung „einen Service veröffentlichen“ verwendet. Einen Service *aufrufen* bedeutet, dass dem Service eine Nachricht gesendet wird und erwartet wird, dass der Service korrekt ausgeführt wird. Gleichbedeutend wird die Formulierung „einen Service *ausführen*“ verwendet. Die erste Variante betont, dass eine Nachricht versendet wird und nimmt die Requestor-Sichtweise der Kommunikation ein, während „ausführen“ die Leistungserbringung des Services fokussiert, die *beim* Provider und *für den* Requestor stattfindet. Es ist weiterhin zu berücksichtigen, dass die Nachricht nicht direkt an einen Service (im Sinne einer angebotenen Leistung, siehe Definition 7) gesendet wird. Vielmehr wird die Nachricht an eine Komponente gesendet, die den Provider dazu veranlasst, die Leistung zu erbringen. An dieser Stelle sei vom genauen Ablauf der Kommunikation abstrahiert<sup>2</sup>, so dass die oben beschriebenen Verwendung der Verben ausreichend ist.

**Serviceorientierte Architektur** Eine serviceorientierte Architektur (SOA) ist eine spezielle Software-Architektur, die sich auf Services bezieht. Dies bedeutet, dass eine serviceorientierte Architektur die strukturierte Anordnung und die Beziehungen von Services beschreibt. Es lassen sich dann die Rollen Service Provider, Service Requestor und Service Broker identifizieren. Anders formuliert ist eine serviceorientierte Architektur ein Modell eines offenen, verteilten Software-Systems, in dem Services von ihren Providern für einen Requestor dadurch verfügbar gemacht werden, dass Service-Beschreibungen bei einem Service Broker veröffentlicht werden. Von einem *offenen* System wird hier ausgegangen, da jederzeit Services sowohl hinzukommen können als auch entfernt werden können. Gleiches gilt offensichtlich auch für die Teilnehmer, also Service Provider und Requestor. So gesehen interagiert das System auch mit seiner Umwelt und ist daher offen. Die Umsetzung einer serviceorientierten Architektur muss Techniken und Mechanismen zur Verfügung stellen, um die Dynamik zu handhaben. Ein IT-System, dass

---

<sup>2</sup>Die Kommunikation mit einem Service wird bei der Formulierung des Service-Modells in Abschnitt 3.2 detaillierter beschrieben.

eine serviceorientierte Architektur realisiert, wird im Folgenden als *serviceorientierte Umgebung* bezeichnet. Die Systemkomponenten in einer serviceorientierten Umgebung sind die Services selbst, die Infrastruktur zur Kommunikation und der Service Broker. Durch die Interaktionen der Teilnehmer beziehungsweise der Services untereinander wird ein System gebildet. Im Folgenden werden nun die Rollen der Teilnehmer einer serviceorientierten Architektur betrachtet und ihre Interaktion geschildert.

Zum Abschluss der Begriffsdefinition sei darauf hingewiesen, dass die erbrachte Leistung als Produkt im Sinne des Electronic Commerce anzusehen ist, wie in [168] definiert. Dies bedeutet, dass in einer Umsetzung einer serviceorientierten Architektur nicht nur digitale Produkte, sondern auch physikalische Produkte angeboten werden können. In dieser Arbeit wird die serviceorientierte Architektur als Software-Architektur für *e-Business* betrachtet, im Sinne der integrierten Ausführung der digitalisierten Bestandteile ökonomischer Prozesse [149]. An dieser Stelle sei darauf hingewiesen, dass bei der späteren Betrachtung der Qualität eines Services auch die nicht-digitalisierbaren Bestandteile relevant sind, da sie aus ökonomischer Sicht zum Service gehören und somit beispielsweise für die Auswahl eines Services relevant sein können.

### 2.2.2 Service Provider

Ein Provider bietet Services an, also Leistungen, die in einem Software-System angeboten und für andere verfügbar gemacht werden. Wird ein Service aufgerufen, so wird die angebotene Leistung erbracht. Zur Leistungserbringung werden Ressourcen eingesetzt, das heißt, beim Aufruf eines Services hat der Provider intern den Einsatz der Ressourcen zu planen und durchzuführen, so dass die Leistung erbracht wird.

Die Ressourcen bilden eine Grenze der Diskurswelt. Sie werden im Service-Modell als Entitäten identifiziert, da eine wesentliche Aufgabe für den Provider das interne Service-Management ist, bei dem der Einsatz der Ressourcen zu planen und durchzuführen ist. Die Ressourcen stellen eine konzeptionelle Unterscheidung zwischen dem Service und der zur Leistungserbringung einzusetzenden Entitäten dar. Zusätzlich muss ein Markt existieren, damit es sich wirtschaftlich lohnt, die Leistung als Service anzubieten. Dabei muss der Provider auch den zusätzlichen Aufwand berücksichtigen, der durch eine Integration in die serviceorientierte Umgebung entsteht. Ziel ist es nun nicht, einzelne Ressource anzubieten, sondern Services, bei deren Aufruf die Ressourcen zur Leistungserbringung eingesetzt werden können. Damit ein Service aufgerufen werden kann, wird eine Komponente zur Kommunikation benötigt, so dass der Service für Andere erreichbar ist. Aus der Sicht eines Service Providers lässt sich somit folgende grobe Vorgehensweise zur Bereitstellung eines Services definieren, die auch in Abbildung 2.1 dargestellt ist. [7]

1. Zunächst gilt es die potenziellen verfügbaren Ressourcen zu identifizieren



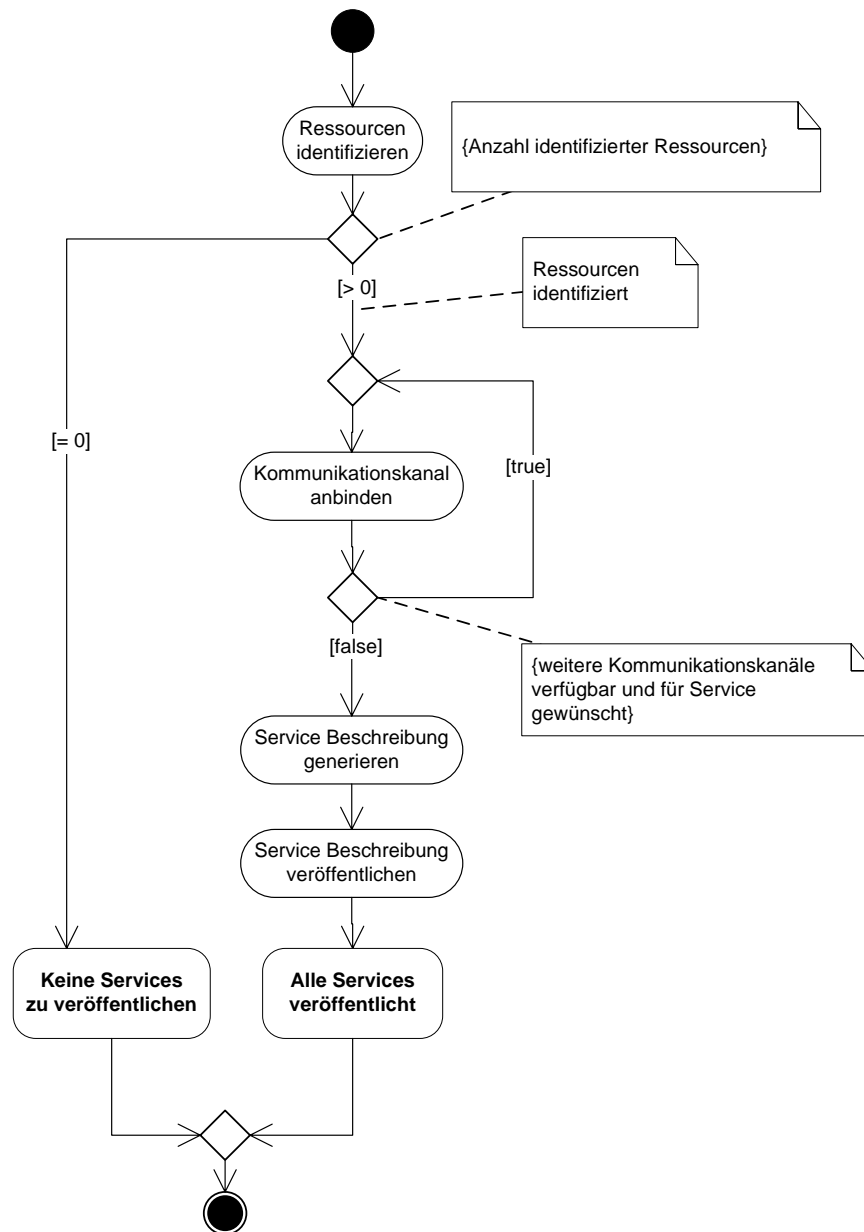


Abbildung 2.1: Ablauf der Veröffentlichung eines Services durch einen Service Provider, dargestellt als UML Aktivitätsdiagramm

und den potentiellen Markt dahingehend zu analysieren, ob ein Bedarf für die anzubietenden Leistungen existiert.

2. Die Services sind in eine serviceorientierte Umgebung zu integrieren. Dazu muss der Service von Anderen aufgerufen werden können, das heißt, eine Verbindung mit einem Kommunikationskanal ist herzustellen.
3. Wenn in der serviceorientierten Umgebung mehrere Kommunikationskanäle existieren und der Service an diese angeschlossen werden soll, so ist die Anbindung für alle gewünschten Kanäle zu iterieren.
4. Die Beschreibung des Services ist zu erzeugen. Diese beschreibt auch die Kommunikationsschnittstellen, also die Anschlüsse an die Kommunikationskanäle.
5. Diese Beschreibung ist bei einem oder mehreren Service Brokern zu veröffentlichen, so dass die potentiellen Kunden die Beschreibung des Services finden können.

Bei welchen Brokern die Service-Beschreibung veröffentlicht wird, sollte aus den Ergebnissen der in Schritt 1 erzeugten Marktanalyse hervorgehen. Gegebenenfalls sind nach der Veröffentlichung zusätzliche Marketing-Maßnahmen zu starten. Wenn die Services eines Providers nun bereitgestellt sind, gilt es im laufenden Betrieb die Services beziehungsweise die Ressourcen zu verwalten. Dieses wird in der Literatur als Service-Management bezeichnet [118]. Zu den Aufgaben während des Betriebs gehören unter anderem:

- Die Service-Aufrufe sind zu empfangen und die Eingabedaten weiterzuleiten. Hierbei sind möglichst geeignete Ressourcen auszuwählen beziehungsweise die Last auf die verfügbaren Ressourcen geeignet zu verteilen.
- Die Leistung ist zu erbringen, indem Ressourcen eingesetzt werden. Die Ausführung ist dabei zu beobachten (in der englischsprachigen Literatur als *Monitoring* bezeichnet) und auf Fehlerfälle ist geeignet zu reagieren.
- Findet die Leistungserbringung außerhalb der serviceorientierten Umgebung statt, so können Fehler dazu führen, dass weitere Services aufgerufen werden. Kann beispielsweise das verschickte Buch nicht zugestellt werden oder wird es zurückgesandt, so ist die Adresse zu überprüfen beziehungsweise die Rechnung zu stornieren. Scheitert die Datenanalyse aufgrund fehlerhafter Daten, so ist zu ermitteln, ob diese Daten ignoriert werden können oder ob bereinigte Daten zugesandt werden können.
- Sofern zutreffend, sind die Ergebnisse der Service-Ausführung an den Requestor zurückzusenden. Hierbei kann es sich um die eigentliche Leistung handeln, wie beispielsweise die Ergebnisse der Datenanalyse, oder um eine einfache Benachrichtigung als Bestätigung, beispielsweise durch den Buchhändler, dass das Buch verschickt worden ist.

- Die Bezahlung der Leistungserbringung ist zu regeln. Dabei sind klassische betriebswirtschaftliche Wege vorstellbar, wie Rechnungsdruck, -versand und anschließende Bezahlung. Außerdem gibt es bereits einige Anbieter von Internet-Bezahldiensten, die eingebunden werden können. In der Literatur wird dieser Themenkomplex als *Service Payment* bezeichnet und die Rolle eines *Billers* für die monetären Angelegenheiten vorgeschlagen [34].
- Aus den Erkenntnissen während der Ausführung kann das Angebot optimiert werden. Außerdem sind gegebenenfalls neue Ressourcen hinzuzufügen oder der Provider muss auf konkurrierende Services geeignet reagieren. Änderungen im Umfeld sind zu berücksichtigen, wie beispielsweise gesetzliche Anforderungen, neue Service Broker und neue Kommunikationskanäle. Derartige Änderungen führen zu Änderungen der Service-Beschreibung oder der Art und Weise, wie und wo die Beschreibung veröffentlicht wird.

Bislang wurde ein erster, zentraler Abstraktionsschritt der Service-Orientierung betrachtet: Die konkreten Leistungen, die durch den Einsatz von Ressourcen erbracht werden können, werden als abstrakter Service angeboten. Im nächsten Abstraktionsschritt werden Services zusammengefasst. Dies wird erreicht, indem funktional äquivalente Services dieselbe standardisierte Kommunikationsschnittstelle (englisch *Interface*) implementieren. Dadurch können zusätzliche Kunden gewonnen werden, für die eine Kommunikation einfacher ist, wenn eine standardisierte Schnittstelle benutzt wird. Für Kunden ist somit der Umstieg auf einen alternativen Provider technisch relativ einfach realisierbar.

### 2.2.3 Service Broker

Eine zentrale Rolle einer serviceorientierten Architektur übernimmt der Service Broker. Der Service Broker speichert und verwaltet die Beschreibungen der Services. Daher ist prinzipiell vor jeder Benutzung eines Services eine Anfrage an den Broker zu stellen. Ein Service Broker agiert dabei als Händler, der zwischen Provider und Requestor vermittelt und dadurch die direkte Kommunikation ermöglicht. Seine primäre Aufgabe ist die Weitergabe der Informationen, die er von den Providern erhält.

Neben dieser zentralen Aufgabe eines Brokers gibt es weitere Funktionalität, die hier beschrieben werden soll: Services können bestimmten Kategorien zugeordnet und klassifiziert werden, um das Auffinden eines Services zu erleichtern. Hierbei wird beispielsweise die angebotene Leistung so abgebildet, dass bei einer entsprechenden Anfrage eines Requestors alle funktional adäquaten Services gefunden werden können. Weiterhin wird eine mächtige Anfragesprache im Allgemeinen als notwendig angesehen, um Services zu finden. Ein Broker muss diese zur Verfügung stellen und effizient auswerten können. Bezieht man diese Aufgaben mit ein, so erscheint der Begriff Broker nicht unbedingt angemessen. In der Literatur finden sich daher einige alternative Bezeichnungen für diese Rolle:

- **Service Registry** [2, 3, 4, 35, 76]: Dieser Begriff fokussiert vor allem auf das Registrieren von Services. Ein Provider registriert (und löscht) verfügbare Services. In einem Register gibt es üblicherweise eine Ordnung, beispielsweise eine Kategorisierung. Dadurch können Services mit einer bestimmten Art der Leistung besser gefunden werden.
- **Service Repository** [161]: Ein Repository kann als Datenbanksystem verstanden werden, also ein integrierter Datenbestand eines Anwendungsgebietes. Durch die Verwendung dieses Begriffes werden die Datenverwaltung und die Eigenschaften eines Datenbanksystems betont. Dazu gehören unter anderem mächtige Anfragesprachen, Mehrbenutzerbetrieb, Autorisierung, Fehlertoleranz und Transaktionalität.
- **Service Directory** [84]: Ein Verzeichnis beschreibt lediglich eine einfache Möglichkeit, Einträge geordnet zu verwalten. Die Sortierung ist hier aber weniger mächtig als bei einem Repository. Wird dieser Begriff verwendet, so wird die Rolle eher als passiver, zentraler und standardisierter Dienst verstanden.
- **Service Bus** [116, 132]: Von einigen Software-Herstellern wird der Begriff Service Bus geprägt. Dabei wird die Kommunikation und die Funktionalität des Brokers beziehungsweise des Repositories als untrennbar zusammengefügt.

In dieser Arbeit soll dennoch die Bezeichnung Service Broker [25, 139, 62] verwendet werden. Hier wird der Aspekt in den Vordergrund gestellt, dass diese Rolle ein *Intermediär* ist. Ein Intermediär in der Betriebswirtschaftslehre bezeichnet im Allgemeinen einen Zwischenhändler am Markt [167]. In der Wertschöpfungskette bringt ein Intermediär somit einen Zwischenschritt ein und gleichen Marktunvollkommenheiten aus [41]. In einer serviceorientierten Umgebung ist die Marktunvollkommenheit dadurch gegeben, dass nicht jeder Requestor alle verfügbaren, für ihn potentiell relevanten Services und deren Provider kennt.

Erhält ein Service Broker eine Anfrage für einen Service, so ist es seine Aufgabe adäquate Services zu suchen und die Liste der verfügbaren, passenden Services zurückzugeben. Eine Sortierung nach geeigneten Kriterien, so dass eine Rangfolge entsteht, ist offensichtlich sinnvoll.

### 2.2.4 Service Requestor

An dieser Stelle sei vorausgesetzt, dass der Service Requestor eine Leistung benötigt, die ein Service Provider für ihn erbringen soll. Er formuliert dazu eine Anfrage an einen Service Broker um verfügbare Services zu ermitteln, die eine von ihm benötigte Leistung anbieten. Somit sei auch vorausgesetzt, dass dem Service Requestor mindestens ein relevanter Broker bekannt ist und er über die Technologie und Legitimation verfügt, Anfragen an diesen Broker zu stellen. In der Literatur finden

sich für die Rolle eines Teilnehmers, der einen Service nutzen möchte folgende alternative Bezeichnungen:

- **Service Consumer** [62, 78]: Der Teilnehmer, der den Service wirklich benutzt, also konsumiert. Hier steht die Leistungserbringung im Vordergrund, übertragen auf eine eingesetzte Ressource ergibt sich der Konsum dieser Ressource. In einer serviceorientierten Umgebung steht vor dem Konsum üblicherweise die Suche nach einem entsprechenden Service.
- **Service Customer**: Die allgemeine Bezeichnung eines Kunden wird oft verwendet als Gegenstück zum Anbieter (englisch Supplier). Aus der Sicht des Providers ist der Kunde derjenige, der die Ausführung des Services beantragt und das Ergebnis erhält. Hier wird die wirtschaftliche Geschäftsbeziehung zwischen den beiden Rollen betont.
- **Service Requestor** [4, 84] : Aus der Sicht eines Service bezeichnet dieses die Rolle des Teilnehmers, der diesen Service anfragt. In der englischen Literatur findet sich sowohl die Schreibweise Requester [179] als auch die (veraltete) Schreibweise Requestor. In der überwiegenden Anzahl wird in der Literatur aber explizit die Schreibweise Requestor verwendet. Daher wird diese Schreibweise auch in dieser Arbeit verwendet.

Damit ein Requestor eine Anfrage an den Broker korrekt stellen kann, muss neben der Technologie und Anfragesprache auch bekannt sein, wie die Services im Broker organisiert sind. Dazu sind beispielsweise Kategorien oder Suchschlüssel zu bestimmen, welche Art der gewünschten Leistung repräsentieren. Hier soll zunächst davon ausgegangen werden, dass die Beschreibung der Art der angebotenen Leistung aus spezifizierten Terminologien stammen können sowie aus De-facto-Industriestandards, wie bereits in Abschnitt 2.2.2 erwähnt. Aus Sicht eines Service Requestors lässt sich grob der folgende Ablauf bis zur Auswahl und Benutzung eines Services definieren, der auch in Abbildung 2.2 dargestellt ist:

1. Die Anforderungen an den benötigten Service sind zusammenzustellen. Zunächst ist hierbei die notwendige Funktionalität zu spezifizieren<sup>3</sup>.
2. Wenn es mehrere Broker gibt: Ermittlung der relevanten Broker, die potentielle Services vermitteln können und bei denen der Requestor die Zugangsvoraussetzungen erfüllt.
3. (Je Broker:) Um die Anfrage zu formulieren sind die für diesen Broker notwendigen Informationen und Anforderungen zusammenzutragen und in die Anfragesprache des Brokers zu übersetzen.

---

<sup>3</sup>Im Laufe der Arbeit wird die Notwendigkeit gezeigt, ebenfalls Anforderungen an die Qualität zu spezifizieren.

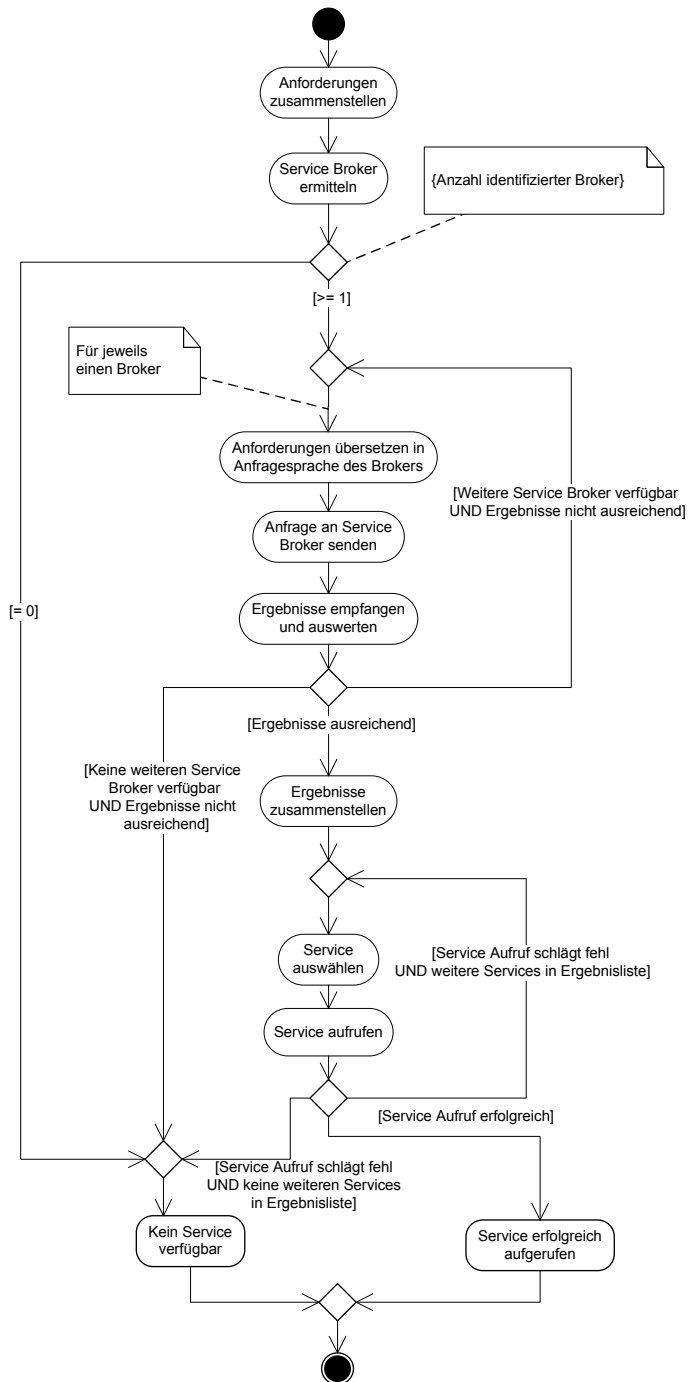


Abbildung 2.2: Aktivitätsdiagramm eines Service Requestor zur Service-Auswahl

4. (Je Broker:) Senden der Anfrage über den Kommunikationskanal an den Broker. Im Allgemeinen wird dabei die in 3. zusammengestellte Anfrage als Nachricht verpackt und an den Broker gesendet.
5. (Je Broker:) Empfangen und auswerten der Ergebnisse. Dazu sind die Ergebnisse des Brokers aus der Antwort zu extrahieren.
6. Sind ausreichend Ergebnisse eingetroffen, so sind diese zusammenzuführen und zu sortieren. Dabei sind Duplikate zu eliminieren, wenn mehr als ein Broker angefragt wurde und einzelne Services von mehreren Brokern zurückgegeben wurden.
7. Wurden auf die Anfrage passende Services erkannt, so kann der am besten geeignete Service zum Aufruf ausgewählt wird.
8. Service aufrufen durch senden einer Nachricht.
9. Wenn es Probleme beim Aufrufen des Services gibt, kann auf die anderen Ergebnisse zurückgegriffen werden. Lässt sich kein geeigneter Service finden beziehungsweise erfolgreich aufrufen, so ist kein Service verfügbar. Gegebenenfalls ist die ursprüngliche Anfrage anzupassen und ein manueller Eingriff eines Benutzers erforderlich.

Zum besseren Verständnis wird im Folgenden der einfache Fall betrachtet, dass es genau einen relevanten Service Broker gibt. Die komplette Anfrage wird an den Broker gesendet und wie in Abschnitt 2.2.3 beschrieben, wird eine Liste adäquater Services zurückgeliefert, welche die benötigte Leistung anbieten. Im einfachsten Fall passt die Anfrage eines Requestors exakt zu einem angebotenen Service. In einer serviceorientierten Umgebung ist es wahrscheinlicher, dass ein möglichst geeigneter Service aus den passenden ausgewählt werden muss, weil die Anfrage kein eindeutiges Resultat liefert [132].

Aus der Rückgabeliste muss der Service Requestor einen Service auswählen, den er aufrufen will. Offensichtlich ist diese Auswahl nicht trivial und wird mit steigender Zahl funktional äquivalenter Services schwerer zu treffen. Die in 6. und 7. angesprochenen Aktionen sind somit komplexe Aufgaben. Diese können nur dann gelöst werden, wenn weitere Informationen über die Services, beispielsweise über die Kosten oder die Ausführungsdauer, vorhanden sind. Es ist die dann Aufgabe des Requestors, aus den unterschiedlichen, funktional äquivalenten Services den am Besten geeigneten Service auszuwählen. Ist ein Service ausgewählt, kann der Requestor den ausgewählten Service aufrufen (Schritt 8.). Dazu wird mit Hilfe der Service-Beschreibung eine Nachricht erstellt, die dann zum Service gesendet wird.

Nachdem die Nachricht an den Service gesendet wurde, wartet der Requestor in Schritt 9. gegebenenfalls auf die Antwort des Services. Dabei sind zwischenzeitlich eventuell Nachrichten oder Bestätigungen zu empfangen und auszuwerten. Zusätzlich überwacht der Requestor die Fristen, in denen der Service antworten

muss. Tritt eine Fehlermeldung auf, weil der Service nicht korrekt ausgeführt werden kann oder wird eine Frist verletzt, so muss der Requestor aus der Liste der passenden Services einen nächsten Service auswählen und diesen aufrufen.

### 2.2.5 Weitere Rollen

Neben den drei zentralen Rollen finden sich in der Literatur weitere Rollen, die hier zur Vollständigkeit erwähnt werden sollen und weil sie weiteres Potenzial der Service-Orientierung aufzeigen. In [34] wird ein *Broker* so definiert, dass er den besten registrierten Service aufgrund von Kriterien, die der Requestor angibt, ermittelt. Hier werden offensichtlich weitere Informationen wie beispielsweise Kosten oder Ausführungsdauer benötigt. Demgegenüber ist ein *Registrar* definiert, der Services registriert und veröffentlicht, Anfragen entgegen nimmt und verfügbare Services ausgibt, dass heißt, es wird also keine Vorauswahl getroffen.

Ein *Reseller* stellt lediglich einen Proxy für einen Service Provider dar und ein *Host* bietet physikalisches Hosting für Service Provider an. Es fließen auch Lösungsvorschläge für aktuelle Probleme ein, beispielsweise indem eine Rolle vorgeschlagen wird, die für die Authentifizierung von Teilnehmern zuständig ist [126]. Ein solcher *Authenticator* erlaubt die Identifikation von Service Requestoren. Diese Authentifizierung kann aber auch für Provider angewendet werden und von Brokern genutzt werden.

## 2.3 Web Services

Im Folgenden wird eine aktuelle Technologie geschildert, die eine Umsetzung einer serviceorientierten Architektur ermöglicht. Üblicherweise wird diese Technologie als *Web Services* bezeichnet [4]. Web Services basieren auf dem Kerngedanken der Nutzung vorhandener Kommunikations-Infrastruktur, indem sie auf der allgegenwärtigen Internet-Technologie aufsetzen. Bisher steht bei der Web Services Technologie eine möglichst einfache und pragmatische Realisierung der wesentlichen Rollen und Interaktionen im Vordergrund. Web Services basieren im Wesentlichen auf den Spezifikationen *SOAP* zum Nachrichtenaustausch, *WSDL* zur Service-Beschreibung und *UDDI* als Service Broker. Wesentliche Gemeinsamkeit der Spezifikationen ist die Benutzung der Extensible Markup Language (XML) [23] als Basissprache. Im Folgenden werden diese Spezifikationen kurz vorgestellt und herausgestellt, welche Konzepte einer serviceorientierten Architektur sich mit ihnen realisieren lassen.

**SOAP** *SOAP* [175, 177] definiert das Aussehen von Nachrichten die in einer serviceorientierten Architektur verschickt werden. Dies betrifft sowohl Service-Aufrufe und die entsprechenden Antworten, als auch die *publish*- und *find*-Kommunikation mit dem Broker. Innerhalb eines Umschlags (*Envelope*) gibt es einen



Kopfteil (*Header*) der die Adressaten und Absender der Nachricht definiert, sowie den eigentlichen Nachrichtenteil (*Body*). Die SOAP-Spezifikation definiert ein XML-Schema und gibt mögliche Anbindungen an Kommunikationsprotokolle an; üblicherweise wird SOAP über das Internet-Protokoll HTTP verschickt. Hieraus resultiert auch die Bezeichnung *Web Services*, da dasselbe Transport-Protokoll verwendet wird wie für die Seiten des World Wide Web.

**WSDL** Die *Web Services Description Language* (WSDL) [180] wird von den wichtigsten Software-Anbietern unterstützt und ist der De-facto-Standard für die Beschreibung von Web Services. WSDL definiert ein XML-Schema für die konkreten Details, wie und wo der Service angeboten wird. Die wesentlichen Elemente einer Web Service Beschreibung in WSDL sind in Abbildung 2.3 dargestellt. Dabei wird ein Web Service auf zwei Abstraktionsebenen beschrieben:

1. Zunächst werden die Nachrichten, die ein Web Service sendet und empfängt, durch XML-Schema-Typen beschrieben. Eingabe- und Ausgabe-Nachrichten, eine Definition der Nachrichtenreihenfolge und die logischen Rollen werden in *Operationen* zusammengefasst. Ein *Interface* (Schnittstelle) gruppiert logisch zusammengehörige Operationen. Da mehrere Web Services ein Interface implementieren können, wird die Wiederverwendung vereinfacht. Das Interface schließt somit die erste Abstraktionsebene ab. Diese Beschreibungen sind unabhängig von einem konkreten Kommunikationsprotokoll.
2. Auf der zweiten Abstraktionsebene wird dann das Kommunikationsprotokoll (in der Regel SOAP über HTTP, siehe oben) als *Binding* spezifiziert (ausgedrückt durch die *spezifiziert Transport-Beziehung*), sowie die Netzwerkadresse (entsprechend eine URI [15]) in einem *Endpoint* festgelegt. Mehrere dieser so definierten Endpunkte werden zu einem *Web Service* gruppiert. Dadurch ermöglicht es WSDL einem Requestor, eine korrekte Nachricht für einen Aufruf eines Web Services zusammenzustellen und diese an einen entsprechenden Endpunkt zu schicken. An diesem Endpunkt nimmt der Provider die Nachricht entgegen, führt den Web Service aus und verschickt gegebenenfalls eine Antwortnachricht.

Diese beiden Abstraktionsebenen spiegeln dabei die Ebenen wider, die bereits in Abschnitt 2.2.2 beschrieben wurden. Im ersten Schritt wird von der konkreten Leistungserbringung abstrahiert, indem ein Service angeboten wird. Im zweiten Schritt werden dann mehrere Services durch eine gleiche Kommunikationsschnittstelle zusammengefasst. Die Interfaces, die ein Web Service dabei implementiert, sind dieselben, die den gruppierten Endpoints beziehungsweise deren Bindings entsprechen; dies wird durch die Anmerkung „assoziert dasselbe Objekt“ in Abbildung 2.3 abgebildet.

**UDDI** Die Rolle des Service Brokers nimmt eine Instanz eines Universal Description Discovery & Integration (UDDI) ein [35]. UDDI ist eine Spezifikation, die

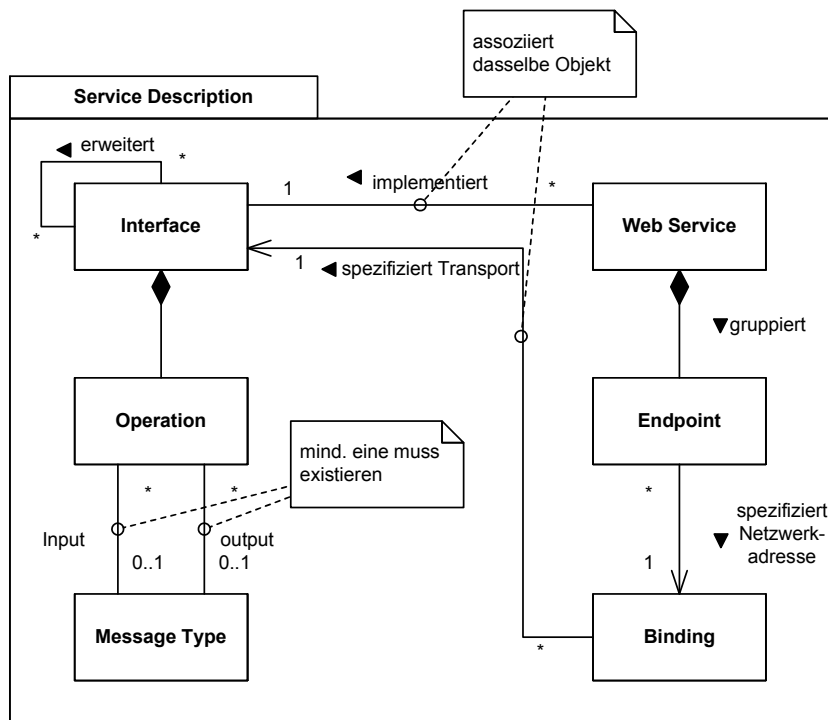


Abbildung 2.3: Wesentliche Elemente des WSDL-Modells

eine als Registry bezeichneten Broker und dessen Funktionalität beschreibt (zum Begriff Registry siehe auch Abschnitt 2.2.3). Dazu sind Datenmodelle und Funktionen für die Beschreibung und die Suche nach Service Providern, Web Services und technischen Aspekten der Kommunikationsschnittstellen für die Web Services definiert. Zusätzlich ist es möglich, den Web Service und den Provider textuell zu beschreiben.

**Diskussion der Web Services-Technologie** Mit SOAP wird eine einheitliche Grundlage für die Kommunikation unter Benutzung allgemein verfügbarer Technologie geschaffen. Web Services können diese Kommunikationstechnik benutzen und ihre Schnittstellen in WSDL-konformen Dokumenten beschreiben, die dann bei einem UDDI-konformen Broker veröffentlicht werden. Diese technologische Grundlage ist für viele Anwendungsgebiete benutzbar. Der Begriff *Web Services* bestimmt lediglich die Kommunikationstechnologie. Dennoch zeigt sich die Vielfalt der unterschiedlichen Begriffsdefinitionen und damit verbundener Sichtweisen besonders im Zusammenhang mit dem Begriff Web Services. Auch wenn über die oben geschilderte technologische Realisierung weitestgehend Einigkeit herrscht, so bleibt viel Spielraum über die Interpretation des Begriffs Web Service; in [139] werden einige Interpretationen aufgezählt.

Das W3C definiert einen Service als eine abstrakte Ressource [178] (im Sinne

einer Abstraktion von konkreten Ressourcen), die eine Fähigkeit repräsentiert, bestimmte Funktionen zu verrichten. Diese Funktionen bilden laut W3C sowohl aus Provider- als auch aus Requestor-Sicht eine kohärente Gesamtfunktionalität. Ein Service ist aus Sicht des W3C zunächst unabhängig von einer konkreten Ressource und lediglich durch eine Service-Beschreibung identifiziert. Damit ein Service jedoch benutzbar ist, muss er durch einen konkreten Provider (beziehungsweise dessen Ressource) realisiert werden. Um einen Web Service zu finden, der die gewünschte Leistung erbringen kann, stellt der Requestor eine Anfrage an ein UDDI Registry. Problematisch ist hierbei, dass UDDI lediglich so genannte *tModel-Keys* (technical model) anbietet, mit denen eindeutige Schlüsselwerte für technische Konzepte vergeben werden können. Die Schlüsselwerte müssen daher die Art der Leistung beziehungsweise die Funktionalität repräsentieren und dem Web Service zugeordnet sein. Sind dem Requestor die Schlüsselwerte bekannt, können passende Web Services automatisch identifiziert werden. Da WSDL sich ebenfalls auf die technischen Aspekte der Kommunikation konzentriert, passen UDDI und WSDL zwar gut zueinander, denn den Elementen eines WSDL-Dokumentes lassen sich tModel-Keys zuweisen. UDDI und WSDL lösen aber nur die technischen Aspekte des Kommunikationsaufbaus. Die Anforderungen eines wirtschaftlichen Umfeldes müssen ebenso berücksichtigt werden, wie die konkreten Bedürfnisse eines Requestors. Um eine informierte und anforderungsgerechte Auswahl unter passenden Web Services zu treffen, sind also zusätzliche Informationen notwendig, beispielsweise über die Kosten oder Ausführungsdauer der Web Services.

Es besteht zwar die Möglichkeit alle verfügbaren Web Services zu einem Interface<sup>4</sup> oder von bestimmten Service Providern zu finden, letztendlich muss aus den passenden Web Services einer manuell durch einen menschlichen Experten ausgewählt werden. Die Abbildung der Funktionalität über tModel-Keys ist nicht ausreichend. Zusätzlich fehlen für nicht-funktionale Eigenschaften (z. B. Kosten oder Ausführungsdauer) ein angemessenes und einheitliches Konzept [37, 100]. Der Einsatz eines UDDI-konformen Service Brokers wird in der Praxis kritisch gesehen [62] und in der Forschung werden verschiedene Verbesserungsvorschläge diskutiert [119, 138, 187]. Dennoch wird zumindest der unternehmensinterne Einsatz von Web Services als viel versprechend angesehen. Wie beispielsweise die Computer-Zeitung bereits in ihrer Ausgabe 34-35 vom 28. August 2006 berichtete, sieht das IT-Beratungsunternehmen Gartner die Web Services-Technologie zumindest für unternehmensinterne Anwendungen als für produktive Umgebungen realisierbare Lösung. Gartner ordnet dementsprechend in ihrem *Hypecycle für Emerging Technologies* interne Web Services der Phase *Plateau der Produktivität* zu<sup>5</sup>.

---

<sup>4</sup>*Interface* bezeichnet hier das XML-Element in einem WSDL-Dokument. Dieses XML-Element kann mehrfach verwendet werden, indem der qualifizierende Name des Elementes verwendet wird.

<sup>5</sup>Gartners Hypecycle bezeichnet fünf Phasen in der Technikentwicklung. Zu Beginn gibt es einen Zündpunkt der Technikidee, der sich bis zum Höhepunkt der Euphorie steigert. Anschließend folgt das Tal der Desillusionierung, wo die Technik erst wieder mit steigender Erkenntnis erholt. Ziel ist am Ende das Plateau der Produktivität.

## 2.4 Zusammenfassung

Der Kerngedanke der Service-Orientierung ist zunächst relativ einfach: Ein Service wird von einem Provider zur Verfügung gestellt, indem er eine Beschreibung des Services veröffentlicht gibt. Ein Requestor hat dann die Möglichkeit diese Beschreibung zu finden, wenn er einen entsprechenden Service nutzen möchte. Die Beschreibung enthält alle notwendigen technischen Informationen, um den Service zu nutzen. In den vorherigen Abschnitten ist für die Rollen einer serviceorientierten Architektur geschildert worden, dass die Eigenschaften der Services, die zur Auswahl eines Services relevant sind wie beispielsweise Kosten und Ausführungsdauer, eine zentrale Rolle einnehmen. In dieser Arbeit wird soll daher ein Meta-Modell für derartige Service-Eigenschaften erstellt werden. Wie bereits geschildert, lassen sich die Eigenschaften der Services intuitiv in zwei Arten unterteilen:

1. **Funktionale Eigenschaften** beschreiben, *was* der Service macht. Es wird die vom Provider für Andere angebotene Art der Leistung beschrieben.
2. **Qualitätsmerkmale** beschreiben, *wie* ein Service seine Leistung erbringt, wenn er ausgeführt wird, beziehungsweise wie die Ergebnisse der Service-Ausführung beschaffen sind.

Die Notwendigkeit und die möglichen Potentiale einer Einbeziehung der 2. Art der Eigenschaften wurden in den Abschnitten 2.2.2, 2.2.3 und 2.2.4 jeweils aus der Perspektive dieser Rolle geschildert. In Abschnitt 2.3 wurde diskutiert, dass die bisherige Realisierung mittels Web Services als Kommunikationstechnologie diesbezüglich nicht ausreichend ist.

Im folgenden Kapitel wird ein Service-Modell erarbeitet. Hierbei werden die in diesem Kapitel formulierten Konzepte der Service-Orientierung abgebildet, so dass ein konzeptioneller Rahmen für die Elemente der Service-Beschreibung gebildet wird. Das Service-Modell bildet dann die Grundlage für das Qualitätsmodell.



## Kapitel 3

# Service-Modell

In diesem Kapitel wird, basierend auf den grundlegenden Konzepten und Begriffsdefinitionen des vorherigen Kapitels, ein Service-Modell formuliert. Dazu werden in einem ersten Schritt die Annahmen kurz zusammengefasst (Abschnitt 3.1). Zur Modellbildung werden zunächst zentrale Konzepte wie Funktionalität, die Kommunikation und die konzeptionelle Unterscheidung zwischen Service und Service-Typ erarbeitet. Dies wird dann in dem Service-Modell abgebildet (Abschnitt 3.2).

Das Modell wird anschließend untersucht, indem die Abstraktionsebenen erläutert werden. Es wird untersucht, welche Vorteile bei einer Umsetzung des Service-Modells, insbesondere durch die Konzepte Service und Service-Typ, realisierbar werden (Abschnitt 3.3). Anschließend werden zentrale Prinzipien der Service-Orientierung präzisiert, die für eine effektive und effiziente Umsetzung einer serviceorientierten Architektur zu beachten sind (Abschnitt 3.5). Anschließend wird der Zusammenhang zwischen dem Service-Modell und der Web Services-Technologie untersucht. Hier werden insbesondere die Elemente des WSDL-Modells betrachtet (Abschnitt 3.6). Das Kapitel schließt mit einer Zusammenfassung in Abschnitt 3.7.

### 3.1 Annahmen

Die folgenden Annahmen gelten für das Service-Modell. Teilweise sind diese bereits im Zuge der Grundlagen der Service-Orientierung in Kapitel 2 erklärt worden. Daher wird hier nur eine kurze Zusammenfassung gegeben. Eine weiterführende Erklärung, welche Auswirkungen die Annahmen auf die Elemente des Service-Modells haben, folgt in den anschließenden Abschnitten.

1. Der Provider will eine Leistung, die er unter Einsatz seiner Ressourcen erbringen kann, als Service anbieten. Er veröffentlicht dazu eine Beschreibung bei einem Service Broker.
2. Der Requestor will eine Leistung durch einen Dritten erbracht haben. Er

kann dazu passende Services, die bei einem Broker veröffentlicht sind, finden und somit die Beschreibung der Services erhalten.

3. Es existiert mindestens ein Service Broker, der sowohl Provider als auch Requestor bekannt ist und mit dem beide kommunizieren können.
4. Für die funktionalen Eigenschaften des Services existieren standardisierte Beschreibungen, auf die sich die Teilnehmer einer serviceorientierten Umgebung geeinigt haben.
5. Es existieren allgemein akzeptierte Standards für die Kommunikation, insbesondere für den Service-Aufruf und die entsprechende Antwort (*bind*) sowie für die Kommunikation mit dem Broker (*publish* und *find*).
6. Es existiert eine Kommunikationsmöglichkeit, auf die sowohl Provider, als auch Requestor zugreifen können und somit miteinander kommunizieren können.
7. Es existieren allgemein akzeptierte Standards für Schnittstellen, welche die Kommunikation mit Services definieren.

## 3.2 Formulierung des Service-Modells

Die Beschreibung der Services ist von zentraler Bedeutung: Sie wird vom Provider erstellt und veröffentlicht, vom Broker verwaltet und an einen Requestor weitergegeben. Dieser wiederum benutzt die Beschreibung, um einen Service aufzurufen. Zusätzlich ist bereits angesprochen worden, dass es durchaus mehrere Services geben kann, die der Anfrage des Requestors genügen, so dass eine Auswahl durch den Requestor erfolgen muss. Da der Requestor zunächst nur auf die Informationen des Brokers zugreifen kann, ist die Service-Beschreibung selbst von zentraler Bedeutung [34]. Als Annahme wurde herausgearbeitet, dass eine gemeinsame Auffassung aller Beteiligten über die Konzepte einer serviceorientierten Architektur existieren muss und somit auch über die Elemente einer Service-Beschreibung. Es existiert im Kontext von Web Services mit WSDL bereits eine Spezifikation zu diesem Thema. Wie in Abschnitt 2.3 diskutiert wurde, bezieht sich WSDL vor allem auf die technische Realisierung eines Service-Aufrufes, also auf die zur Kommunikation erforderlichen Aspekte. Die Elemente einer Service-Beschreibung in WSDL wurden in Abbildung 2.3 aufbereitet. Allerdings sind einige Elemente des WSDL-Modells stark auf den Kontext der Web Services zugeschnitten (beispielsweise die Elemente *Binding* und *Endpoint*) und andere Aspekte sind nicht ausreichend dargestellt. Dies betrifft neben den bereits erwähnten Qualitätsmerkmalen eines Services auch die Spezifikation der Funktionalität, wie im Folgenden erläutert wird.

**Funktionalität** In der 2. Annahme (Abschnitt 3.1) ist festgelegt, dass ein Requestor einen Service mit passenden funktionalen Eigenschaften finden kann. Er er-

kennt also, dass ein bestimmter Service die von ihm benötigte Leistung erbringen kann. Bei der Begriffsdefinition in Abschnitt 2.2 wurde in Definition 5 die Leistung definiert. Es ist dabei darauf hingewiesen worden, dass eine Leistung *konkret* ist:

„Transportgut  $g_1$  wird vom Ausgangsort  $ort_1$  abgeholt und zum Zielort  $ort_2$  transportiert“

Demgegenüber sucht ein Requestor, wie in Abschnitt 2.2.4 beschrieben, in einer serviceorientierten Architektur nach einer bestimmten *Funktionalität*, damit die von ihm benötigte Leistung erbracht werden kann. Dazu spezifiziert er die benötigte Funktionalität und sucht (über einen Broker) nach einer (formalen) Beschreibung einer Funktionalität. Die Beschreibung einer Funktionalität ist, im Gegensatz zur Leistung, *abstrakt*; im Transport-Beispiel kann eine Beschreibung wie folgt aussehen:

„Transport eines Gutes  $G$  von einem Ort  $A$  zu einem anderen Ort  $B$ “.

Eine Funktionalität spezifiziert also die (abstrakte) Art der Leistung, die bei der Ausführung eines Services (konkret) erbracht wird. In der Service-Orientierung ist die Funktionalität eine standardisierte Spezifikation der Art einer Leistung, die erbracht werden kann. Eine Funktionalität kann auf unterschiedliche Art und Weise und mit verschiedenen Techniken spezifiziert werden. Beispielsweise kann die Beschreibung der Funktionalität in einer informalen, textuellen Beschreibung erfolgen, in Form von Vorbedingungen und Effekten [136], in einer formalen Sprache [64] oder unter Benutzung einer gemeinsamen Ontologie [47, 143] erfolgen.

Im Kontext von Web Services können beispielsweise Schlüsselwörter als tModel-Keys in einem UDDI als Beschreibung der Funktionalität angesehen werden. Hinter diesem Schlüsselwort verbirgt sich eine Beschreibung, wie eine Transport-Dienstleistung funktioniert. Darauf haben sich die Beteiligten eines Gremiums geeinigt und ein tModel-Key wurde dafür vereinbart. Zum Beispiel stehe der Schlüssel *shipping* für den Transport eines Gutes von einem Ort zu einem anderen Ort.

**Schnittstelle** Die Annahmen 5. und 7. beziehen sich auf eine standardisierte Kommunikationstechnik beziehungsweise standardisierte Kommunikationsschnittstellen. In der Informatik definiert eine Schnittstelle im Allgemeinen die Kommunikation einer Software-Komponente. Eine *Schnittstelle* fasst mehrere *Operationen* zusammen, die aufgerufen werden können. Neben einem eindeutigen Namen hat eine Operation Parameter, wobei mindestens einer als *Eingabe* oder einer als *Ausgabe* existieren muss. Zusätzlich lassen sich Fehlermeldungen als (Ausgabe-) Parameter angeben. In der Service-Orientierung werden Parameter durch *Nachrichten* verschickt, die durch einen Typ (*Nachrichtentyp*) definiert sind. Dies wird durch die folgenden Begriffsdefinitionen festgehalten. Ausgangspunkt sei die in Abschnitt 2.2 in der Definition 3 als Teil eines Software-Systems definierte Komponente (Seite 20). Eine Komponente hat wohldefinierte Grenzen, die durch eine Schnittstelle definiert sind:



**Definition 8 Schnittstelle:** Eine Schnittstelle ist die Spezifikation der Kommunikation mit einer Komponente.

Eine Schnittstelle definiert also die Art und Weise, wie mit dieser Komponente kommuniziert werden kann. Dazu werden einzelne Interaktionen als Operationen spezifiziert.

**Definition 9 Operation:** Eine Operation definiert eine Interaktion mit einer Komponente und spezifiziert Nachrichtentypen.

Eine Operation definiert somit einen Teil der Kommunikation mit einer Komponente und stellt somit einen Teil der Schnittstelle einer Komponente dar. Eine Operation hat üblicherweise einen Namen und kann von außen aufgerufen werden; ein Synonym ist der Begriff *Methode*. Wenn eine Komponente von außen aufgerufen wird, so wird eine Nachricht an die Komponente geschickt. Üblicherweise wird dabei eine durch eine Schnittstelle nach außen bekannt gemachte Operation aufgerufen.

**Definition 10 Nachrichtentyp:** Ein Nachrichtentyp spezifiziert den Aufbau und die Datentypen einer Nachricht.

Eine Nachricht enthält verpackte Informationen für eine Komponente und wird zwischen Komponenten ausgetauscht. Für jede Nachricht gibt es einen Sender und mindestens einen Empfänger. Die Antwort einer Komponente auf den Erhalt einer Nachricht ist in der Regel ebenfalls eine Nachricht, die gegebenenfalls die Antwort enthält. Der Erhalt einer Nachricht kann dem Absender durch eine Bestätigungsnachricht mitgeteilt werden. Eine Fehlernachricht ist ebenfalls als Nachrichtentyp zu definieren.

**Endpunkt** Die Interaktion zwischen Komponenten eines Systems wird als Kommunikation bezeichnet und findet in einer serviceorientierten Umgebung per Nachrichtenaustausch über einen Kommunikationskanal statt. Die Art des Kommunikationskanals bestimmt unter anderem, wie Nachrichten verpackt und verschlüsselt werden. Es kann in einer serviceorientierten Umgebung mehrere Kommunikationskanäle geben und somit mehrere Implementierungen einer Schnittstelle für einen Service. Dadurch ergibt sich das Problem, dass jede Implementierung der Schnittstelle eine eigene Beschreibung benötigt. Der Aufwand für den Provider würde erhöht werden, wenn er jede Beschreibung einzeln veröffentlichen muss. Auch das Ergebnis einer Anfrage eines Requestors wird dann unnötig vergrößert.

Daher wird das Konzept des *Endpunktes* eingeführt. Ein Endpunkt (der Kommunikation) implementiert eine Schnittstelle, indem die Anbindung an einen Kommunikationskanal realisiert wird. Ein Endpunkt wird konzeptionell unterschieden vom Service, für den diese Anbindung realisiert wird. Ein Service gruppiert alle Endpunkte, die genau *die* Schnittstelle implementieren, die zu seinem Service-Typ

gehört. Für das Beispiel des Buchversands seien jeweils ein Endpunkt als Web Service per SOAP über HTTP (also Web-Technologie) und SOAP über SMTP (E-Mail) implementiert. Beide Endpunkte implementieren die standardisierte Schnittstelle des Buchversandhandels. Veröffentlicht wird dann eine Beschreibung des Services, die beide Endpunkte beschreibt. Ein Requestor kann dann entscheiden, über welchen Endpunkt er kommunizieren möchte. Ohne die konzeptionelle Unterscheidung von Services und Endpunkten müsste für jeden Kommunikationskanal ein eigener Service angeboten werden. Dieses wird in den folgenden Definitionen festgehalten.

**Definition 11 Kommunikationskanal:** Ein Kommunikationskanal ist das Medium über das die Komponenten in Software-Systemen interagieren.

Jede Komponente im Software-System, die mit anderen Komponenten interagiert, verfügt über einen Anschluss an einen Kommunikationskanal. Dieser Anschluss wird als *Endpunkt* bezeichnet.

**Definition 12 Endpunkt:** Ein Endpunkt realisiert die Anbindung einer Komponente an einen Kommunikationskanal in einem Software-System.

Für einen Service kann es mehrere Endpunkte geben. Aus Sicht eines Providers ist es daher sinnvoll, die Kommunikatoren einem Service zuzuordnen. Veröffentlicht wird in der *Service-Beschreibung* der Service und seine Endpunkte, sowie der Bezug zu den realisierten Schnittstellen und der spezifizierten Funktionalität.

**Service und Service-Typ** Nachdem nun die Funktionalität und die Kommunikation definiert ist, können nun die zentralen Begriffe des Service-Modells, insbesondere Service und Service-Typ, abschließend definiert werden. Die Definition 7 (Seite 21) wird hierbei lediglich in der zweiten Bedingung erweitert.

**Definition 13 Service** Ein Service ist eine angebotene Leistung, für die folgende Bedingungen gelten:

1. Das Angebot erfolgt in einem Software-System.
2. Das Angebot beinhaltet die Veröffentlichung einer Beschreibung der Leistung. Diese spezifiziert die Funktionalität und die Kommunikation mit dem Anbieter zur Erbringung der Leistung.

Die Beschreibung der Leistung wird als *Service-Beschreibung* bezeichnet. ◇

Einem Service ist mindestens ein Endpunkt (siehe Definition 12) zugeordnet, der eine Kommunikationsschnittstelle implementiert. Durch einen Endpunkt kann die Ausführung eines Services angestoßen werden, so dass die Leistung erbracht wird.

In dem Service-Modell ist das eingangs des Abschnitts beschriebene Problem zu berücksichtigen, dass vom Provider zwar ein Service angeboten wird, aber der Requestor eine Funktionalität benötigt, die durch einen Service realisiert wird. Die Lösung des Problems ist die konzeptionelle Unterscheidung zwischen einem konkreten *Service* und der *Funktionalität*, die dieser Service realisiert. Zusätzlich ist die standardisierte Schnittstelle zu berücksichtigen, damit eine einfache und effiziente Kommunikation zum Service-Aufruf möglich wird. Diese konzeptionelle Unterscheidung wird manifestiert durch einen *Service-Typ*:

**Definition 14 Service-Typ:** Ein Service-Typ fasst Services mit gleicher Funktionalität und gleicher Schnittstelle zusammen.

Anders ausgedrückt spezifiziert ein Service-Typ Funktionalität und Schnittstelle. Jeder Service implementiert genau einen Service-Typ. Zum Beispiel sei  $T_{shipping}$  der Service-Typ mit der Beschreibung der Funktionalität *shipping* (siehe oben) und einer (hier nicht näher betrachteten) Schnittstelle. Weiterhin sei  $WS_1$  ein Web Service des Typs  $T_{shipping}$ . Die Beschreibung von  $WS_1$  sei in Form eines WSDL-Dokuments bei einem Service Broker veröffentlicht und ein Endpunkt bei einem Applikations-Server aktiviert. Der Endpunkt ist eine Software-Komponente, die entsprechende SOAP-Nachrichten entgegen nimmt und verarbeitet, indem die Ausführung des Services angestoßen wird.

**Zusammenfassung des Service-Modells** Die vorangegangenen Beschreibungen der zentralen Elemente und Konzepte einer serviceorientierten Architektur, sowie die Begriffsdefinitionen werden im Service-Modell zusammengefasst, das in Abbildung 3.1 als UML Klassendiagramm dargestellt ist. Dieses Modell bildet die beschriebenen Elemente und ihre Zusammenhänge ab. Die Kardinalitäten ergeben sich aus den Definitionen beziehungsweise aus den Erklärungen zu den Definitionen.

Ein Service-Typ hat also genau eine spezifizierte Schnittstelle und es können Funktionalitäten für einen Service-Typ definiert sein. Eine ähnliche Sichtweise auf das Konzept eines Service-Typen findet sich auch in [92]. Ein *Service* implementiert genau einen Service-Typ und ein Service-Typ kann durch mehrere Services implementiert werden. Dadurch ist es möglich, dass ein Requestor bei der Suche nach einer Funktionalität über den Service-Typ alle Services erhält, die entsprechende Service-Typen implementieren. Andererseits kann ein Provider einen standardisierten Service-Typ implementieren und dadurch von potentiellen Kunden gefunden werden.

Das Service Modell in Abbildung 3.1 enthält auch die Entität *Ressourcen*. Ressourcen werden zur Leistungserbringung eingesetzt, wenn ein Service aufgerufen wird. Aus Sicht des Services werden Ressourcen also *benutzt*. Im übertragenen Sinne werden die Ressource als Endpunkt an einen Kommunikationskanal angeschlossen. Es kann aus Sicht des Requestors auch mehrere Endpunkte geben, die

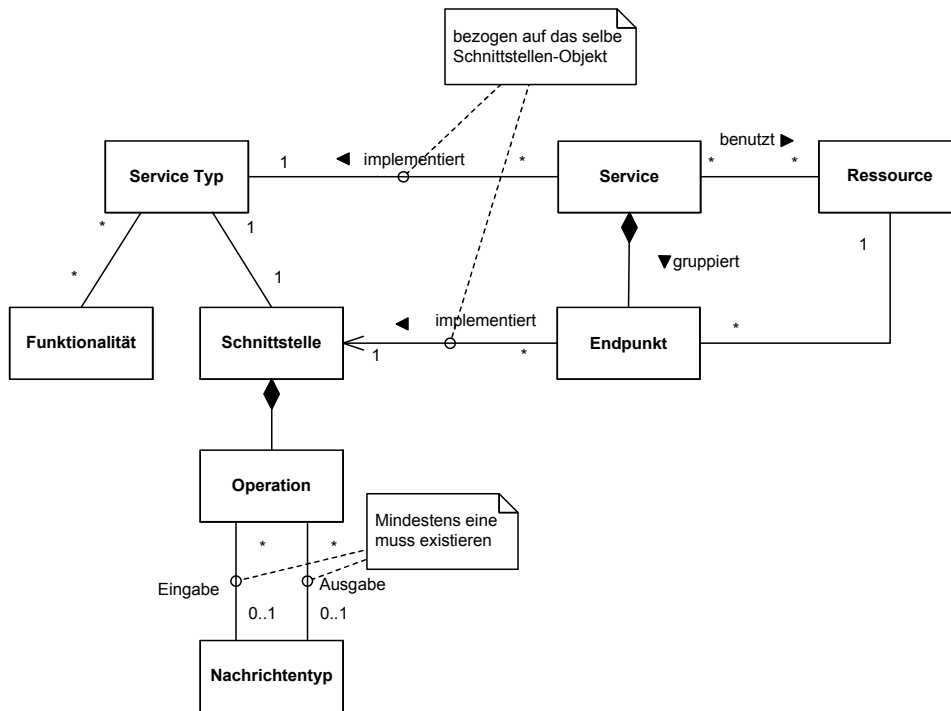


Abbildung 3.1: Das Service-Modell

zur selben Ressource auf der Provider-Seite geleitet werden. Weiterhin ist eine Ressource von mehreren Services benutzbar. An dieser Stelle sei auf einen Unterschied zu den Definitionen des W3C eingegangen werden. Im Kontext von Web Services wird eine Ressource dort mit einem Uniform Resource Identifier (URI) [15] gleichgesetzt beziehungsweise identifiziert [178]. Somit entspricht dort eine Ressource genau einem Kommunikationsendpunkt (Endpoint). Im Gegensatz dazu wird hier der wirtschaftliche Hintergrund berücksichtigt. Es wird dadurch eine deutlichere Abstraktion ermöglicht, die in Abschnitt 3.3 betont wird. Die Ressource dient der konzeptionellen Unterscheidung zwischen der als Service angebotenen Leistung und der tatsächlichen Leistungserbringung durch eine Ressource. Die Ressource ist allerdings kein Bestandteil der Service-Beschreibung.

Ziel des Service-Modells ist die Klärung der Begrifflichkeiten und Abgrenzung der Diskurswelt. Sowohl Ressourcen als auch Endpunkte stellen den Rand der betrachteten Diskurswelt dar. Es wird eine konzeptionelle Trennung von Service, Ressource und Endpunkt formuliert. Insbesondere vor dem Hintergrund der verbreiteten Web Services Technologie (Endpunkt-Begriff) und der Workflow-Management Konzepte (Ressourcen-Begriff) dient diese Abgrenzung unter anderem auch dazu, ein einheitliches Verständnis zu schaffen.

### 3.3 Abstraktionsebenen

Bereits in Abschnitt 2.2 wurde eine zweistufige Abstraktion als zentrales Konzept der Service-Orientierung grundlegend erläutert. In diesem Abschnitt wird untersucht, wie diese Abstraktionsebenen im Service-Modell abgebildet werden. Es sei an dieser Stelle bereits darauf hingewiesen, dass diese Abstraktionsebenen ebenfalls einen Ansatzpunkt für das Qualitätsmodell in Kapitel 4 darstellen.

**Service** Ein Service ist eine Abstraktion von der konkreten Leistung beziehungsweise von den Ressourcen, die zur Leistungserbringung eingesetzt werden, wenn der Service aufgerufen wird. Ein Service fasst auch mehrere Endpunkte zusammen (siehe Definition 12). Für die Rollen in einer serviceorientierten Umgebung lassen sich die folgenden Vorteile erkennen:

- **Provider:** Der Hauptvorteil liegt für den Provider darin, dass er seine Ressourcen selbst verwalten kann. Er nimmt die über verschiedene Endpunkte eintreffenden Service-Aufrufe entgegen und setzt geeignete Ressourcen ein. Daher kann der Provider beispielsweise je nach Auslastung, in Abhängigkeit vom jeweiligen Kunden oder von den Parametern des Aufrufs entsprechende Ressourcen einsetzen.

Zusätzlich kann der Provider neue Ressourcen hinzufügen beziehungsweise Ressourcen austauschen, ohne die veröffentlichte Service-Beschreibung ändern zu müssen. Dies ist möglich, da die Ressourcen nicht Bestandteil der Service-Beschreibung sind. Weiterhin kann eine Ressource bei der Leistungserbringung von mehreren Services eingesetzt werden.

Neben zusätzlichen Ressourcen können zusätzliche Endpunkte implementiert werden, beispielsweise für neue Kommunikationskanäle. Dabei muss allerdings die veröffentlichte Service-Beschreibung angepasst werden.

- **Requestor:** Vorteilhaft für den Requestor ist, dass er für einen ausgewählten Service einen der angebotenen Endpunkte auswählen kann. Günstig ist hierbei beispielsweise einen Endpunkt für einen Kommunikationskanal zu wählen, für den bereits eine Anbindung des Requestors existiert, die dann genutzt werden kann.

Die Auswahl der Ressource für die Leistungserbringung muss nicht durch den Requestor vorgenommen werden, sondern durch den Provider. Dennoch garantiert der Provider, dass die Leistungserbringung in einer angegebenen Qualität erfolgt, beispielsweise bezüglich Antwortzeit und Preis.

- **Broker:** Für den Broker wird die Verwaltung der veröffentlichten Service-Beschreibungen einfacher, weil nicht die konkreten Leistungen oder die einzelnen Ressourcen verwaltet werden müssen, sondern die Services. Dadurch wird eine gröbere Granularität erreicht, der Suchraum für den Broker verkleinert und somit die Bearbeitung von Anfragen erleichtert.

**Service-Typ** Als zweiter Abstraktionsschritt werden die Gemeinsamkeiten mehrere Services in einem gemeinsamen Service-Typ zusammengefasst. Konkret bedeutet dies, dass ein Service-Typ die Funktionalität und die Schnittstelle spezifiziert. Der Service-Typ abstrahiert von der Realisierung der Funktionalität (also der Leistung, die bei Aufruf des Service erbracht wird) und der Schnittstelle (also dem Endpunkt). Für die Rollen in einer serviceorientierten Umgebung lassen sich die folgenden Vorteile erkennen:

- **Requestor:** Durch die standardisierte Schnittstelle wird der Aufruf aller Services eines Service-Typs immer nach dem gleichen Schema und mit denselben Parametern ablaufen. Genauer gesagt bleiben Operationen und Nachrichtentypen identisch (siehe Definitionen 9 und 10). Wird vom Requestor ein bewährter und oft benutzter Service gegen einen anderen Service ausgetauscht, so wird der Aufwand für die Benutzung eines anderen Services geringer, wenn derselbe Service-Typ implementiert wird. Generell sind Services desselben Service-Typs daher relativ einfach austauschbar. Dieses Prinzip wird im Allgemeinen als lose Kopplung bezeichnet (siehe auch Abschnitt 3.4.4). Letztendlich wächst dadurch auch der Konkurrenzdruck, was zwar vorteilhaft für den Requestor ist, sich aber als Nachteil für den Provider herausstellen kann. Aus Sicht eines Requestor ergibt sich somit der Vorteil, dass es für einen Service-Typ mehrere Services und somit auch mehrere Provider geben kann, die er relativ einfach austauschen kann.

Zusätzlich ist die Suche nach einem Service, der für eine bestimmte Aufgabe eingesetzt werden soll, einfacher, wenn der Service-Typ bereits feststeht. Die benötigte Funktionalität braucht dann nicht mehr in der Anfrage an den Broker beschrieben werden. Gleichzeitig ergibt sich der oben erwähnte Vorteil, dass die Anfragebearbeitung durch den Broker erleichtert wird.

- **Provider:** Für einen Provider ist es möglich, mehrere Services anzubieten, die alle den gleichen Service-Typ implementieren und somit die gleiche Funktionalität und Schnittstelle besitzen, sich aber in anderen Eigenschaften voneinander unterscheiden. Dies können beispielsweise Qualitätsmerkmale wie Antwortzeit oder Kosten sein. Dadurch besteht für den Provider die Möglichkeit der Diversifikation [1, 9]. Es werden somit mehrere Services angeboten, zwischen denen der Requestor auswählen kann. Für alle Services eines Service-Typs ist aber sichergestellt, dass Funktionalität und Schnittstelle seinen Anforderungen entsprechen. Ohne das Konzept der Service-Typen muss für jeden einzelnen Service zusätzlich zu den Qualitätsmerkmalen überprüft werden, ob sowohl Funktionalität als auch Schnittstelle passen. Weiterhin besteht ohne standardisierte Service-Typen die Gefahr, dass die Services eines Providers nicht vom Requestor gefunden werden, beispielsweise weil unterschiedliche Auffassungen über die Beschreibung einer Funktionalität vorliegen.

Es ist also vorteilhaft, wenn ein Provider seinen Service als Implementierung

von standardisierten Service-Typen anbieten kann, weil er so mehr potentielle Kunden erreichen kann. Ebenso können De-facto-Standards von Marktführern etabliert werden und auch von Konkurrenten genutzt werden. Letzteres kann für den Marktführer selbst einen Nachteil darstellen, für alle anderen Marktteilnehmer aber einen Vorteil bedeuten.

Ein Provider kann mit denselben Ressourcen mehrere Services anbieten, die unterschiedliche Service-Typen implementieren. Zum Beispiel kann es in verschiedenen Branchen unterschiedliche Standards für eine Transport-Dienstleistung geben, die sich in unterschiedlichen Service-Typen für derartige Services widerspiegeln. Der Provider aus dem Transport-Beispiel aus Abschnitt 1.1 kann nun mehrere Endpunkte für die jeweiligen Schnittstellen realisieren. Somit kann er für jeden Service-Typ einen Service anbieten. Dadurch kann ein Provider seine Ressourcen nicht nur als unterschiedliche Services anbieten, sondern auch mehrere Service-Typen mit denselben Ressourcen realisieren.

- **Broker:** Der Broker muss die Service-Typen verwalten. Er ist somit als zentraler Teilnehmer für die Etablierung von Standard-Typen anzusehen und kann sich beispielsweise auf bestimmte Branchen spezialisieren. Dadurch hat ein Service Broker eine wichtige Stellung im Markt. Auf eine Anfrage eines Requestors kann er eine Liste der passenden Service-Typen zurückgeben oder direkt passende Services. Durch die Einführung von Service-Typen ist eine direkte Kategorisierung vorgegeben und eine vereinfachte Suche für den Broker realisierbar. Eine Kategorie kann dann einem Service-Typ entsprechen.

### 3.4 Prinzipien

In diesem Abschnitt werden Prinzipien beschrieben, die in einer serviceorientierten Umgebung beziehungsweise für Services gelten sollten, damit sich die Realisierung als serviceorientierte Architektur lohnt. Bei der Realisierung einer serviceorientierten Architektur als Software-Systems als ist zusätzlicher Aufwand nötig, beispielsweise für die Anbindung an die Kommunikations-Infrastruktur und die Einhaltung der Standards und Spezifikationen. Daher ist eine serviceorientierte Architektur nicht immer sinnvoll und nicht immer die effizienteste Lösung. Neben einer Beschreibung der Prinzipien werden sie in der Literatur verankert und kritisch betrachtet.

Der Zweck dieser Prinzipien ist nicht, zu entscheiden, ob ein konkretes Software-System eine serviceorientierte Architektur realisiert oder nicht. Es handelt sich vielmehr um charakteristische Eigenschaften, die den Entwurf und die Entwicklung der Software-Architektur beeinflussen [67]. In Definition 4 (Seite 20) werden diese Prinzipien als Bestandteil der Software-Architektur definiert. Diese Prinzipien sind also zu beachten, um die Vorteile einer serviceorientierten Archi-

tektur ausschöpfen zu können. Außerdem ist die allgemeine Hoffnung, dass durch die Einhaltung der Prinzipien die Software-Entwicklung produktiver und in besserer Qualität erfolgt [139].

Die hier vorgestellten Prinzipien finden sich meistens in textuellen Beschreibungen und eher unpräzisen Aussagen über Service-Orientierung in verschiedenen Quellen [4, 7, 25, 32, 46, 94, 111, 116, 129, 139, 178, 179]. Allerdings werden nicht immer alle der hier vorgestellten Prinzipien genannt beziehungsweise beschrieben. Es lassen sich je nach Sichtweise und Fokussierung einige der hier definierten Prinzipien auch unter anderen Bezeichnungen in der Literatur finden. In [139] werden die Prinzipien als *Anforderungen an Services* und Teilnehmer einer serviceorientierten Architektur definiert. Auch hier ist dazu angegeben, dass diese Prinzipien eingehalten werden *sollten*, um die Vorteile einer SOA auszuschöpfen. Dies impliziert allerdings, dass die Anforderungen nicht unbedingt eingehalten werden müssen. Insofern ist der Begriff Anforderungen nicht passend, so dass hier der Begriff Prinzipien verwendet wird.

Die Prinzipien bilden zusätzlich die Grundlage für eine Software-Entwicklung nach dem serviceorientierten Paradigma, welches in Abschnitt 3.5 erörtert wird. Weiterhin sind die Prinzipien zu beachten, wenn das Qualitätsmodell in Kapitel 4 erarbeitet wird. Zunächst werden im Folgenden einige allgemeine Prinzipien erläutert, anschließend wird die Kommunikation, serviceorientierte Prozesse und das Prinzip der losen Kopplung genauer betrachtet.

#### 3.4.1 Allgemeine Prinzipien

Im Folgenden werden einige Prinzipien der Service-Orientierung kurz aufgezählt. Die Prinzipien sind teilweise direkt aus dem Modell beziehungsweise den Definitionen der Elemente des Modells ableitbar. Ein anderer Teil der Prinzipien ist bereits in den Annahmen zum Service-Modell in Abschnitt 3.1 beschrieben worden. Ansonsten stellen sie Verhaltensmuster oder eher anschauliche Beschreibungen dar. Falls zutreffend, ist jeweils ein Verweis auf die Definition angegeben.

- **Verteilte Umgebung:** Eine serviceorientierte Architektur ist eine verteilte Umgebung, in der verschiedene Software-Komponenten an physikalisch getrennten Orten miteinander durch Nachrichtenaustausch kommunizieren.
- **Offenes System:** Eine serviceorientierte Architektur ist ein offenes System, so dass ein Requestor von außerhalb mit einem Service kommuniziert. Ebenso kommen neue Services und neue Provider hinzu und bestehende Services können ausfallen oder entfernt werden (Definitionen 7 und 1).
- **Standardisierung:** Damit eine serviceorientierte Architektur effizient funktionieren kann und Märkte aufgebaut werden können, an denen verschiedene Teilnehmer zusammenkommen, ist eine standardisierte Grundlage notwendig. Dies betrifft die Kommunikation, die Benutzung des Service Brokers und die Einhaltung der Prinzipien. [74] (siehe auch 5. und 7. Annahme)



- **Implementierungsneutralität:** Die Beschreibung eines Services ist unabhängig von seiner Implementierung. Von zentraler Bedeutung ist dabei die Schnittstelle, die nicht auf die Details der Implementierung eingeht. Eine Schnittstelle ist unabhängig von Programmiersprachen (Definition 8). Zu beachten ist, dass beispielsweise Datentypen im Sinne von Byte-Anzahl und Vorzeichenkodierung eindeutig zu definieren sind. Im Kontext von Web Services wird durch den Einsatz von Unicode und XML die Lösung des Problems erheblich vereinfacht.
- **Automatisierung:** Ziel einer serviceorientierten Architektur ist ein hoher Automatisierungsgrad bei der Integration der Geschäftspartner. Der Kommunikationsaufbau, die Umwandlung von Parametern in syntaktisch korrekte Nachrichten und die Überwachung sollten weitgehend automatisiert ablaufen. Ebenso ist es das Ziel, das Verhalten im Fehlerfall zu automatisieren oder Reaktionen vorher zu definieren. Fällt beispielsweise im Ablauf eines Geschäftsprozesses ein aufgerufener Service aus, so kann ein vorher bestimmter, alternativer Service aufgerufen werden.<sup>1</sup>
- **Zustand von Services:** Prinzipiell ist ein Service zustandslos. Das bedeutet, dass der erste Aufruf eines Services unabhängig von einem zweiten Aufruf ist. Somit kann sich die Nachricht, die an einen Service geschickt wird, nicht direkt auf vorhergehende Nachrichten beziehen. Die Wiederverwendung eines Services wird durch die Zustandslosigkeit vereinfacht, da ein Service unabhängig von seinem Kontext ausgeführt werden kann [116]. Dies stellt einen Vorteil für den Requestor dar, da er einen Service einfacher benutzen kann. Andererseits muss jede Nachricht so zusammengebaut werden, dass alle notwendigen Informationen mitgeschickt werden. Dadurch wird zwar der Aufwand erhöht, aber gleichzeitig sinkt die Anfälligkeit für Fehler durch Seiteneffekte oder fehlende Informationen. Weiterhin kann als Vorteil angesehen werden, dass weniger vertraulichen Daten beim Provider vorgehalten werden. Der Provider kann so jede einzelne Anfrage individuell bearbeiten und braucht potentiell weniger lokale Informationen zu speichern.
- **Grobe Granularität:** Services sind eher von grober Granularität. Ein Service ist auf hohem Abstraktionsgrad angelegt und repräsentiert eher eine komplexe Fähigkeit als eine einzelne Aktivität. Die Interaktion mit einem Service ist ebenfalls eher von grober Granularität, da nicht einzelne Details kommuniziert werden, sondern alle notwendigen Daten auf einmal; siehe auch Abschnitt 3.4.2. Dies ist im Service-Modell auch dadurch abgebildet, dass ein Service mehrere Ressourcen zur Leistungserbringung benutzen kann. [172]

Wie die Granularität hier zu verstehen ist, sei zusätzlich durch ein Beispiel

---

<sup>1</sup>Alternative Services in einem Geschäftsprozess werden im Zusammenhang mit der Anwendung des Qualitätsmodells in Kapitel 5 genauer betrachtet.

illustriert: Der Schnittstelle des Service-Typs *Buchversand* sei so definiert, dass es verschiedene Operationen gibt, die aufgerufen werden können, um ein Buch zu bestellen. Eine Operation erlaubt eine Buchbestellung per ISBN, eine andere durch Angabe von Buchtitel und Autor. Ebenso sind Operationen spezifiziert, die eine einfache Versandadresse benötigen und andere, die separate Liefer- und Rechnungsadresse erfordern. Dafür sind jeweils einzelne Operationen spezifiziert. Andere Aufgaben eines Buchhandels sind aber nicht Bestandteil des Services. So ist eine Anfrage nach allen Büchern eines Autors oder die Anfrage nach dem Inhaltsverzeichnis eines Buches als andere Services bzw. Service-Typen zu spezifizieren. Die Granularität ist hier aber jeweils der komplette Buchversand und nicht aufgeteilt in einen Service, der den Provider veranlasst, das Buch vom Lager zu holen und zu verpacken, einen weiteren Service, der die Rechnung druckt oder den Bank-einzug durchführt und ein dritter Service, der den Postversand initiiert.

- **Lange Lebenszeit:** Als weiteres Prinzip wird in [139] eine relativ lange Lebenszeit der Services herausgestellt. Damit ist gemeint, dass ein Service zumindest so lange existieren muss, bis seine Beschreibung veröffentlicht und von einem Requestor gefunden werden kann. Ein Service wird in seiner Lebenszeit in der Regel mehrfach aufgerufen.
- **Flexible Konfigurierbarkeit:** Anwendungen in einer serviceorientierten Architektur werden spät und flexibel konfiguriert. Das bedeutet, dass die interagierenden Komponenten erst spät miteinander verbunden werden und die Konfiguration sowie die Auswahl der Services dynamisch geändert werden kann. Eine Anwendung kann also konfiguriert werden, indem sie verschiedene Services aufruft. Dies wird in der englischsprachigen Literatur als *late binding* bezeichnet. Auch dieses wird vereinfacht durch das Konzept des Service-Typs.

Diese grundlegenden Prinzipien werden in der Literatur durchaus kontrovers diskutiert. So wird beispielsweise in [87] argumentiert, dass der Kommunikationsaufwand und der XML-bezogene, zusätzliche Aufwand möglichst gering sein soll, indem Services mit wenigen, einfachen Parameter definiert werden sollten. Diese Sichtweise findet sich auch in anderen Quellen wieder und führt sogar zu Überlegungen, Services möglichst feingranular zu gestalten [142] und dem Ziel möglichst einfach zu integrierende Schnittstellen zu generieren [79]. Letztendlich können allerdings je nach Anwendungsbereich unterschiedliche Granularitätsgrade sinnvoll sein [74].

#### 3.4.2 Kommunikationsprinzipien

Ein Service ist eine Komponente eines Softwaresystems (siehe Definition 3) und besitzt eine spezifizierte Schnittstelle (siehe Definition 8). Ein Service kann durch

Nachrichten (siehe Definition 10), die an den Endpunkt (siehe Definition 12) geschickt werden, benutzt werden. Somit besagt das Prinzip der **Nachrichten-Kommunikation**, dass die Komponenten in einer serviceorientierten Umgebung durch den Austausch von Nachrichten kommunizieren.

Betrachtet man die Kommunikation einer serviceorientierten Architektur aus informationstechnischer Sicht, so stellt eine Weiterentwicklung der Client-/Server-Kommunikation dar [139] (siehe hierzu auch die Ausführungen in Abschnitt 2.1.2). Die Benutzung eines Services in einer SOA entspricht prinzipiell einem entfernten, asynchronen Methodenaufruf (Remote Procedure Call, RPC) [17], der über einen verweisenden Service Broker ausgeführt wird. Generell kann in einer serviceorientierten Umgebung wie bei einem RPC die Bindung zwischen Provider und Requestor statisch (zur Übersetzungszeit ist der Service bzw. sein Endpunkt bekannt), halb-statisch (beim Start der Anwendung auf Requestor-Seite wird der Service festgelegt) oder dynamisch (direkt vor dem Aufruf wird der Service ermittelt) hergestellt werden. Offensichtlich ist eine dynamische Bindung die flexibelste Variante. Neben dieser Sichtweise sind noch weitere Aspekte der Kommunikation zu betrachten, die im Folgenden beschrieben und untersucht werden. Es zeigt sich dabei, dass die ursprüngliche publish-find-bind Interaktion nicht vollständig ist [92].

**Dokument- und RPC-Stil** In [139] werden zwei grundsätzliche Sichtweisen auf das Verständnis einer serviceorientierten Architektur genannt. Diese Sichtweisen werden als *RPC-Stil* und *Dokument-Stil* (oder *dokumentenzentriert*) bezeichnet [10]. Der RPC-Stil steht für einen entfernten Prozeduraufruf und bedeutet, dass ein Provider seine Services als Sammlung von Operationen versteht, die als Parameter serialisierte Geschäftsobjekte (zum Beispiel in Form von XML-Dokumenten) erhalten. Auf diesen Geschäftsobjekten werden diese Operationen dann ausgeführt. Hierbei wird also für eventuell bereits bestehende Anwendungen das Entwurfsmuster einer Fassade [54] realisiert, die sie als Service in einem offenen, verteilten Software-System verfügbar zu machen. Somit bestimmt die (bestehende) Anwendung die Funktionalität des Services. Diese Sichtweise ist intuitiv und relativ einfach realisierbar, wenn man existierende, unabhängig voneinander entwickelte Komponenten interoperabel machen möchte.

In der dokumentenzentrierten Sichtweise stellen die Dokumente selbst den Hauptgrund und Zweck der verteilten Bearbeitung dar. Jede Komponente liest, bearbeitet, erzeugt und verschickt Dokumente, wobei die Dokumente zur Bearbeitung temporäre Geschäftsobjekte erzeugen. Diese lokalen Geschäftsobjekte werden aber nicht weitergeleitet oder nach außen sichtbar. Bei dieser Sichtweise werden Services als Möglichkeit angesehen, Geschäftsbeziehungen zu realisieren. Die Abgrenzung zwischen diesen Ansätzen ist fließend. Es lässt sich nur schwer sagen, ob ein Service einzeln betrachtet dokumentenzentriert entwickelt wurde oder er eine RPC-Fassade darstellt. Dennoch soll hier darauf hingewiesen werden, dass in einer serviceorientierten Architektur als offenem und verteiltem System, in dem

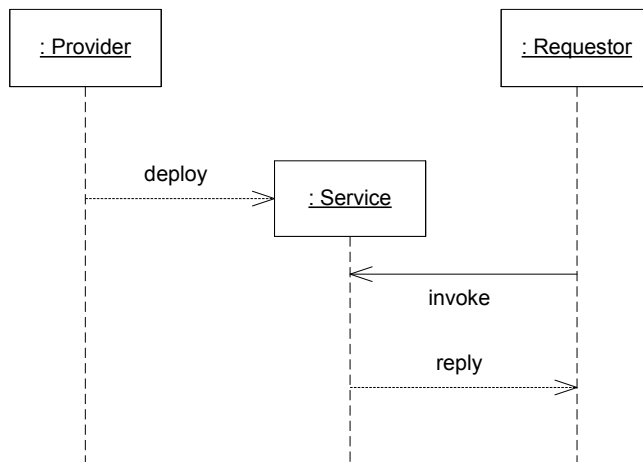


Abbildung 3.2: Direkte Bindung an einen Service

die Services weitgehend unabhängig voneinander existieren, ein dokumentenzentrierter Ansatz vorzuziehen ist [139].

**Direkte und asynchrone Kommunikation** Ein weiteres Kommunikationsprinzip sagt aus, dass die Kommunikation mit einem Service *direkt* und *asynchron* erfolgt. Direkt bedeutet, dass kein dritter Partner benutzt wird, wenn der Service aufgerufen wird. Die Kommunikation mit einem Service ist dabei prinzipiell asynchron. Das bedeutet, man schickt einem Service eine Nachricht mit den notwendigen Informationen zur Ausführung. Anschließend wird der Service ausgeführt und gegebenenfalls werden die Ergebnisse der Ausführung zurückgesendet. In der Literatur wird diese Asynchronität betont, um darauf hinzuweisen, dass die Ausführung des Requestors nicht blockiert ist. Dies soll einen Vorteil einer serviceorientierten Architektur betonen. Dennoch gibt es die Möglichkeit, den Kommunikationskanal offen zu halten, so dass direkt die Antwort zurückgesendet werden kann, so dass technisch gesehen eine synchrone Kommunikation erfolgt (siehe dazu im Kontext von Web Services [10, 180]).

**Varianten zum Aufruf eines Services** Es gibt unterschiedliche Varianten zum Aufruf eines Services. Diese sind in den Abbildungen 3.2, 3.3 und 3.4 dargestellt. Im einfachsten Fall in Abbildung 3.2 wird ein Service aktiviert (in der englischsprachigen Literatur wird dieser Vorgang als *deploy* bezeichnet) und vom Requestor direkt aufgerufen (*invoke*). Es wird eine direkte Bindung des Requestors an genau diesen Service realisiert. Besteht beispielsweise eine feste vertragliche Vereinbarung zwischen Provider und Requestor, so braucht nicht vor jedem Service-Aufruf ein Broker angefragt zu werden. Im Vorfeld kann eine Suche nach einem geeigneten Service beziehungsweise Service-Typ bei einem Broker stattgefunden haben. Diese Variante ist besonders im Kontext von Geschäftsprozessen relevant, die in

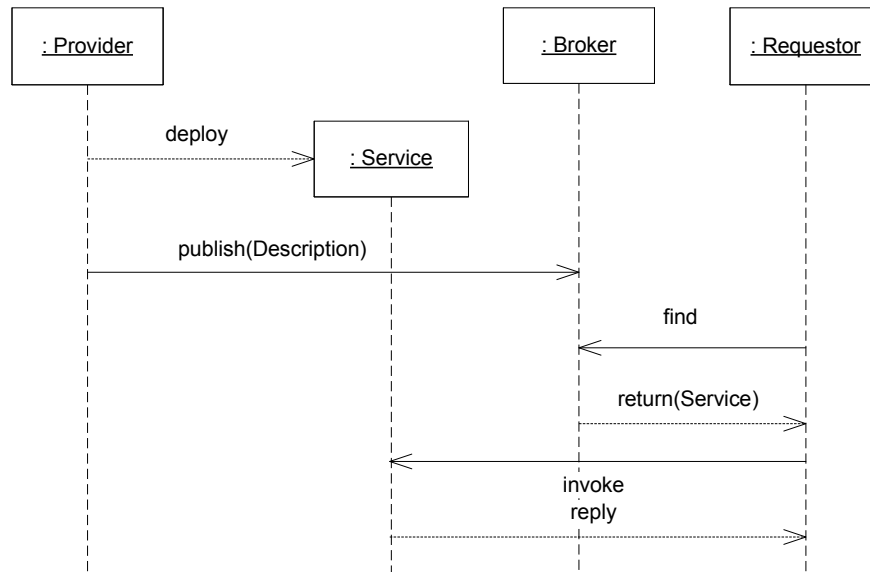


Abbildung 3.3: Dynamische Bindung an einen Service unter Benutzung eines Brokers

Abschnitt 3.4.3 vorgestellt werden und bei Überlegungen zur unternehmensinternen Integration von existierenden Altsystemen [142].

Abbildung 3.3 zeigt die Kommunikationsvariante, die einen Broker mit einbezieht. Beim Broker hat der Service Provider die Beschreibung des Services veröffentlicht, wo der Requestor sie finden kann. Vor jedem Service-Aufruf wird der Broker angefragt. Erst durch die Einbeziehung eines Brokers wird eine vollständige Interaktion im Sinne der serviceorientierten Architektur umgesetzt. Diese Variante entspricht dem in Abschnitt 2.2 vorgestellten Verfahren und ist vor allem für einzelne Service-Aufrufe angemessen.

Ein weiterer Zwischenschritt ist in Abbildung 3.4 eingefügt. Hier wird vor dem eigentlichen Aufruf des Services mit dem Provider verhandelt (*negotiate*) [42], wie der Service auszuführen ist [34]. Üblicherweise werden hier die Ausprägungen der Qualitätsmerkmale wie Preis oder Antwortzeit ausgehandelt. Darüber kann ein Vertrag zwischen den Beteiligten geschlossen werden, der in der Literatur als *Service Level Agreement* (SLA) bezeichnet wird [40, 73]<sup>2</sup>. Ein SLA dokumentiert vertraglich relevante Vereinbarungen und gibt den Beteiligten rechtliche Sicherheit. In der Wirtschaft ist es durchaus üblich, vor einem Auftrag beispielsweise einen Preisrabatt, Qualitätsaspekte und Lieferfristen auszuhandeln. In einer serviceorientierten Umgebung entsteht dadurch ein zusätzlicher Aufwand, wobei hier ebenfalls ein möglichst hoher Automatisierungsgrad angestrebt wird. Soll ein Ser-

<sup>2</sup>In Kapitel 4 werden die Verhandlungen über die Eigenschaften eines Services genauer betrachtet (Abschnitt 4.2.2) und Service Level Agreements in das Qualitätsmodell integriert (Abschnitt 4.3.2, Seite 86)

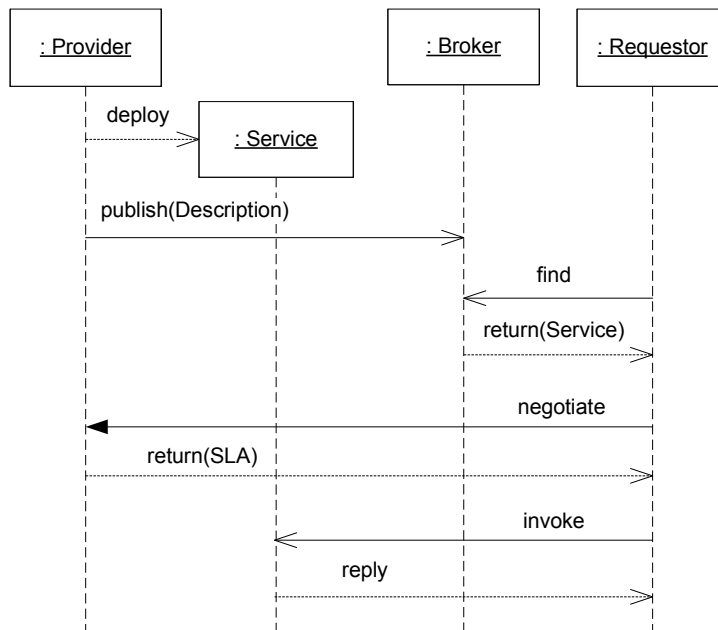


Abbildung 3.4: Dynamische Bindung an einen Service und Verhandlung eines SLA's

vice mehrfach genutzt werden, so ist es vorstellbar, dass diese Variante einmal vor der ersten Benutzung des Services ausgeführt wird. Für alle weiteren Aufrufe kann dann wie in Abbildung 3.2 dargestellt verfahren werden.

### 3.4.3 Zusammengesetzter Service

Ein weiteres, zentrales Prinzip der Service-Orientierung ist die **inhärente Prozessorientierung** [84, 116, 139]. Dieses Prinzip drückt aus, dass einzelne Services zu einem neuen Service zusammengesetzt werden können; dieser wird dann als zusammengesetzter Service bezeichnet. In der Literatur wird dieser Vorgang als *Service Composition* [139] bezeichnet, bei dem dann ein *Composite Service* entsteht<sup>3</sup>.

Ein zusammengesetzter Service verfügt über eine interne Struktur, die einer Prozessdefinition entspricht. Die Prozessdefinition spezifiziert vor allem die Reihenfolge der Prozessschritte; jeder Prozessschritt kann die Interaktion mit einem Service darstellen. Die Prozessdefinition spezifiziert dazu auch die Daten (bzw. Nachrichten), die an diese Services gesendet werden und die als Antworten empfangen werden. Die Ergebnisse beziehungsweise die Antworten der aufgerufenen Services können dann in nachfolgenden Prozessschritten verwendet werden. Darüber hinaus können die Ergebnisse sich auch auf die weitere Ausführung einer Prozessinstanz auswirken. Die Prozessdefinition spezifiziert also Kontrollfluss und

<sup>3</sup>In Kapitel 5 werden die Grundlagen der Geschäftsprozesse ausführlicher beschrieben. An dieser Stelle erfolgt eine kurze Beschreibung des Prinzips der serviceorientierten Prozesse.

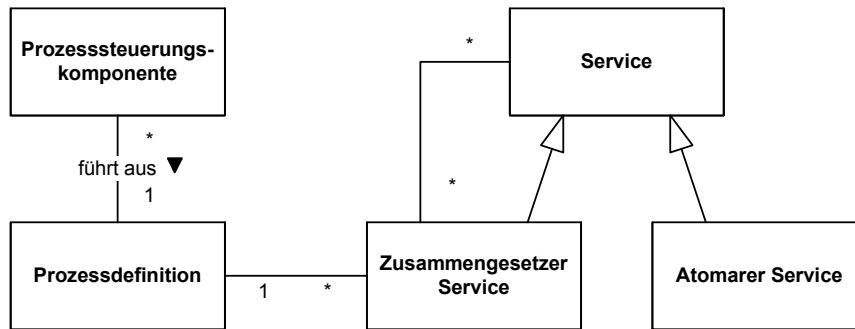


Abbildung 3.5: Erweiterung des Service-Modells für zusammengesetzte Services

Datenfluss [83, 159] und bildet so einen *serviceorientierten Prozess*<sup>4</sup>. Zusätzlich wird eine Prozesssteuerungskomponente benötigt, die Prozessinstanzen entsprechend ihrer Definition ausführt.

Der Prozess kann wiederum als neuer (zusammengesetzter) Service angeboten werden. Wird der zusammengesetzte Service aufgerufen, so wird eine Prozessinstanz erzeugt, die von der Prozesssteuerungskomponente ausgeführt wird. Dies ist prinzipiell einem Workflow [55, 69, 83, 164] sehr ähnlich und führt zu der folgenden Definition:

**Definition 15 Zusammengesetzter Service:** Ein zusammengesetzter Service ist ein Service mit einer Prozessdefinition, anhand derer eine Prozesssteuerungskomponente Prozessinstanzen erzeugen und ausführen kann.

Der Begriff serviceorientierter Prozess soll als Synonym angesehen werden. Wird ein zusammengesetzter Service als neuer Service angeboten, so ergibt sich eine Verschachtelung oder Hierarchie, die in der Literatur als rekursive Benutzung bezeichnet wird [34]. Der Begriff Rekursion ist hier missverständlich, da nicht derselbe Service verschachtelt aufgerufen wird, sondern lediglich Services. Als eine alternative Bezeichnung für die Rolle eines Teilnehmers der einen serviceorientierter Prozess anbietet, wird in [34] *Service Aggregator* vorgeschlagen. In dieser Arbeit wird diese Rolle weiterhin als Provider bezeichnet.

**Erweiterung des Service-Modells** Abbildung 3.5 illustriert die Entitäten eines zusammengesetzten Services in einer serviceorientierten Umgebung. Die Verbindung zum Service-Modell in Abbildung 3.1 ist über das Element Service gegeben; somit stellt die Abbildung eine Erweiterung des Service-Modells dar. Für den Service sind zwei Spezialisierungen dargestellt: ein atomarer Service und ein zusammengesetzter Service. Der atomare Service sei hier ein Service, der keine weiteren

<sup>4</sup>Die Begriffe serviceorientierter Prozess und zusammengesetzter Service stellen Synonyme dar. Während der Begriff serviceorientierter Prozess den Prozesscharakter und die interne Prozessdefinition betont, wird der Begriff zusammengesetzter Service eher verwendet, wenn der Prozess als Gesamteinheit betont und von der interne Prozessdefinition abstrahiert werden soll.

Services aufruft und somit unabhängig von anderen Services ausgeführt werden kann. Demgegenüber hat ein zusammengesetzter Service genaue eine Prozessdefinition und ruft andere Services auf. Die Prozessdefinition spezifiziert einen serviceorientierten Prozess, insbesondere den Kontroll- und Datenfluss. Eine Prozesssteuerungskomponente führt die Prozessdefinition aus. Wird also ein zusammengesetzter Service aufgerufen, so erzeugt die Prozesssteuerungskomponente unter Benutzung einer Prozessdefinition eine Prozessinstanz und führt diese aus [55, 166]. Dabei werden dann die definierten Services aufgerufen.

**Vorteile** Die Prinzipien der Service-Orientierung erlauben eine vergleichsweise einfache Integration verschiedener Services zu einem Prozess. In diesem Sinne argumentiert auch Preston Gralla in [58], der betont, dass die heute bereits verfügbaren Web Services eine gute Möglichkeit darstellen, unternehmensübergreifende Geschäftsprozesse zu integrieren. Es werden verschiedene Studien zitiert, die ähnliche Aussagen treffen. Im Grunde wird aber weniger auf spezielle Eigenschaften einer serviceorientierten Architektur eingegangen, sondern die Notwendigkeit eines standardisierten und damit relativ unkomplizierten Datenaustausches hervorgehoben und motiviert. Dennoch kommt Gralla zu dem Schluss, dass sich Service-Orientierung eignet, um Geschäftsprozessen leichter zu integrieren.

Zweifelsohne ermöglicht eine serviceorientierte Architektur die Flexibilisierung der Geschäftsprozesse durch die Kombination bestehender Services mit geringem technischen Aufwand [129]. Aktuell gibt es vor allem im Kontext von Web Services Ansätze, Sprachen für die Prozessbeschreibung zu spezifizieren, wie zum Beispiel die Business Process Execution Language for Web Services (WS-BPEL) [5] und Web Services Choreography Description Language (WS-CDL) [77]. Es existieren bereits verschiedene Implementierungen einer Prozesssteuerungskomponente, die zusammengesetzte Web Services ausführen. Eine detaillierte Betrachtung der Vorteile serviceorientierter Prozesse, insbesondere unter Berücksichtigung der Qualitätsmerkmale, findet in Kapitel 5 statt.

**Nachteile** Zusätzlich ergeben sich neue Herausforderungen: So müssen Sicherheitsaspekte beachtet werden, beispielsweise wenn vertraulichen Daten zu Services von Dritten geschickt werden. Wie bei allen unternehmensübergreifenden Geschäftsprozessen kann ein Prozessschritt, der von Dritten ausgeführt wird (also ein aufgerufener Service) bei Problemen den gesamten Prozess beeinträchtigen. Ebenso ist die für Geschäftsprozesse wichtige Transaktionalität schwierig zu erreichen wenn Teile eines Geschäftsprozesses von Dritten ausgeführt werden. [89, 115, 145, 172].

Ein weiterer relevanter Aspekt betrifft die Qualitätsmerkmale eines serviceorientierten Prozesses. Eigenschaften wie Ausführungsdauer und Kosten hängen direkt von den aufgerufenen Services beziehungsweise deren Eigenschaften ab [14, 31, 140, 183, 186]. Auch zu diesem Zweck wird das Qualitätsmodell in Kapitel 4 entwickelt und in Kapitel 5 wird dann untersucht und aufgezeigt, wie die Informa-



tionen aus einem Qualitätsmodell genutzt werden können, um die Eigenschaften eines serviceorientierten Prozesses zu ermitteln. Wie bereits in der Problemstellung in Abschnitt 1.2 geschildert, ist ein weiteres Ziel dieser Arbeit, aus den Eigenschaften der einzelnen Services die Eigenschaften des zusammengesetzten Services zu ermitteln.

### 3.4.4 Lose Kopplung

Eines der wichtigsten Prinzipien der Service-Orientierung ist die lose Kopplung der Services [172]. In einer serviceorientierten Umgebung besagt das Prinzip der losen Kopplung, dass ein Service relativ einfach gegen einen anderen ausgetauscht werden kann. Dies soll im Folgenden genauer untersucht werden und es wird aufgezeigt, wie das Service-Modell dazu beiträgt, eine lose Kopplung zu realisieren. Eine lose Kopplung ist besonders dann vorteilhaft, wenn Qualitätsmerkmale berücksichtigt werden, so dass beispielsweise ein Service gegen einen schnelleren und günstigeren Service ausgetauscht werden kann. Zunächst soll zum besseren Verständnis der Begriff Kopplung definiert werden:

**Definition 16 Kopplung:** Kopplung bezeichnet in der Informatik den Grad der Abhängigkeit zwischen (Software-)Komponenten.

Kopplung wird in der Regel in Zusammenhang mit einer Wertung verwendet und gibt also den Grad der gegenseitigen Abhängigkeit von zwei Komponenten an. Allgemein betrachtet sind Komponenten *eng gekoppelt*, wenn sie nicht mit Nachrichten über eine Schnittstelle kommunizieren, sondern beispielsweise über direkten Methodenaufruf oder gemeinsamen Speicherbereich. In der Objekt-Orientierung beschreibt das Prinzip der Kopplung den Grad der Abhängigkeiten zwischen Klassen [11].

In verteilten Umgebungen ist eine enge Kopplung nicht über direkten Methodenaufruf oder gemeinsamen Speicher möglich. Von einer engen Kopplung spricht man hier, wenn der Anwendungsentwickler der einen Komponente genaue und detaillierte Informationen über das Verhalten der anderen Komponente benötigt, um die erforderliche Interaktion der beiden Komponenten zu realisieren. Die Informationen betreffen zum Beispiel die Funktionsweise der entfernten Methodenaufrufe, das Kommunikationsprotokoll und die Semantik der ausgetauschten Nachrichten. Wie in Abschnitt 2.2 erläutert, ist eine serviceorientierte Architektur eine verteilte Umgebung.

Das Gegenteil einer engen Kopplung stellen *vollständig entkoppelten* Anwendungen dar. Hier benötigt man keinerlei Informationen über die Funktionsweise der anderen Seite der Kommunikation. Von einer *losen Kopplung* spricht man, wenn zwar gewisse Informationen notwendig sind, aber keine vollständige Kenntnis über die gegenseitige Funktionsweise erforderlich ist [19, 79].

**Lose gekoppelte Services** Somit gibt die Kopplung die Intensität und die Abhängigkeit der verbundenen Komponenten an, wobei die extremen Grade der Kopplung eng gekoppelt und vollständig entkoppelt sind. Services einer serviceorientierten Architektur werden als lose gekoppelt bezeichnet, welches eine eher geringe Abhängigkeit beschreibt. Dennoch sind Services in einer serviceorientierten Umgebung nicht völlig unabhängig voneinander, da sie durch Nachrichtenaustausch kommunizieren und sich gegenseitig benutzen können. Dies gilt insbesondere im Kontext der im vorigen Abschnitt definierten serviceorientierter Prozesse (siehe Definition 15). Um einen Service zu nutzen, benötigt man zunächst Informationen über den Service selbst, seine Semantik und wie mit ihm kommuniziert werden kann. Neben technischen Aspekten der Kommunikation sind dazu auch Informationen über die auszutauschenden Daten notwendig. Daher werden Services nicht als vollständig entkoppelt bezeichnet.

Die lose Kopplung wird über die standardisierte Beschreibung der Schnittstellen erreicht, sowie über standardisierte Kommunikationstechniken. Desweiteren trägt insbesondere das Konzept der Service-Typen zur Umsetzung einer lose Kopplung bei, da Implementierungen desselben Typs relativ einfach auszutauschen sind, denn die Schnittstelle und die Funktionalität ist bei allen Services eines Typs identisch (siehe Definition 14). Somit sind die Operationen und die Nachrichtentypen bekannt, wenn ein Service gegen einen anderen des gleichen Typs ausgetauscht werden soll. Insbesondere bei serviceorientierten Prozessen können somit einzelne Prozessschritte durch verschiedene Services des gleichen Typs ausgeführt werden.

**Vorteile** Generell lässt sich festhalten, dass es durch die Realisierung einer losen Kopplung für den Requestor relativ einfach ist, einen Service gegen einen anderen Service des gleichen Service-Typs auszutauschen. Daher können diese funktional äquivalenten Services bezüglich ihrer nicht-funktionalen Eigenschaften (Qualitätsmerkmale) untersucht werden. Dieses ist dann sinnvoll möglich, wenn die Ausprägungen der betrachteten Qualitätsmerkmale von Services des gleichen Typs verglichen werden können. Dann kann beispielsweise ein besserer Service gefunden werden, also einer mit besseren nicht-funktionalen Eigenschaften. Zukünftig kann dieser Service anstelle des ursprünglichen aufgerufen werden. Diese Thematik wird in Kapitel 4 erneut aufgegriffen und stellt eine zentrale Motivation für das Qualitätsmodell dar. Ebenso sei hier bereits die Notwendigkeit einer Vergleichbarkeit der Services als Anforderung an das Qualitätsmodell festgehalten.

Auch aus Sicht des Providers ergibt sich ein weiterer Vorteil: Ein Requestor ruft einen Service über den Endpunkt auf und nicht etwas eine konkrete Ressource. Somit hat ein Provider den Vorteil, dass er seine Ressourcen selbst verwalten kann. Der Provider kann zur Laufzeit selbst entscheiden, welche Ressourcen er einsetzt, um die Leistung zu erbringen. Darüber hinaus können die Ressourcen vom Provider verändert oder ausgetauscht werden. Der Requestor kann weiterhin den gleichen Service aufrufen, solange der Endpunkt verfügbar bleibt.

**Nachteile** Ein Nachteil der losen Kopplung ist, dass Transaktionalität schwieriger zu realisieren ist, da die Kommunikation über den Endpunkt ohne weiteres keinen direkten Zugriff auf die Leistungserbringung erlaubt. Es existieren daher einige Bemühungen, Transaktionskonzepte in eine serviceorientierte Architektur beispielsweise durch standardisierte Protokolle zu integrieren [38, 39, 89, 145]. Wird die serviceorientierte Architektur um Transaktionskonzepte erweitert, so wird die Kopplung zwischen den Services verstärkt.

Einen Überblick über die Bedeutung der lose Kopplung im Kontext von Web Services beschreibt Jason Bloomberg in [19]. Es existiert bereits umfassende Werkzeugunterstützung in existierenden Web Service Entwicklungsumgebungen um lose Kopplung durch Web Services zu realisieren.

### 3.5 Das Paradigma der Service-Orientierung

In diesem Abschnitt wird das Paradigma der Service-Orientierung formuliert. Hierbei werden das Service-Modell und die Konzepte und Prinzipien der Service-Orientierung zusammengefasst. Dies untermauert die Notwendigkeit für die Abbildung von Qualitätsmerkmalen, um die Vorteile der Service-Orientierung vollständig ausnutzen zu können.

Etymologisch betrachtet entstammt das Wort Paradigma dem Griechischen (*parádeigma*; *para* = neben und *deiknynai* = zeigen, begreiflich machen) und bedeutet einerseits Beispiel, Muster und Vorbild, andererseits auch Abgrenzung. Ursprünglich wird Paradigma als epistemologischer Ausdruck benutzt, um wissenschaftliche Denkweisen zu beschreiben. Dabei dienen Definitionen von Paradigmen auch dazu, grundsätzliche Ansichten voneinander abzugrenzen. Im aktuellen Sprachgebrauch wird das Wort Paradigma mit unterschiedlichen Bedeutungen verwendet, wobei es weniger für eine umfassende Weltanschauung oder Ansicht steht, als für eine fokussierte Sichtweise auf einen (grundlegenden) Aspekt. In der Informatik gibt es viele Teilgebiete, in denen sich Paradigmen aus unterschiedlichen Gesichtspunkten entwickelt haben. So wird die Objekt-Orientierung als Paradigma bezeichnet. Objekt-Orientierung ist zunächst eine Programmiertechnik mit dazugehörigen Programmiersprachen, allerdings gibt es unter anderem Konzepte zur objektorientierten Analyse und zum objektorientierten Design [123], so dass sich durchaus ein Paradigma der Objekt-Orientierung erkennen lässt.

In der Literatur wird Service-Orientierung oft ebenfalls als neues Paradigma bezeichnet [102, 139, 161], wobei ähnlich wie in der Objekt-Orientierung argumentiert wird, indem Zusammenhänge und Auswirkungen auf die Software-Entwicklung und Software-Landschaft betrachtet werden. Allerdings lässt sich erkennen, dass Service-Orientierung als Paradigma über die eigentliche Software-Entwicklung hinaus geht [96]. Beispielsweise ist die Verwaltung der Ressourcen durch den Provider so durchzuführen, dass die angebotene Leistung jederzeit angemessen erbracht werden kann und dass die oben beschriebenen Prinzipien der Service-Orientierung eingehalten werden [128]. Zusätzlich spiegelt eine service-

orientierte Architektur auch eine wirtschaftliche Entwicklung wieder, wie bereits in Abschnitt 2.1.1 beschrieben. Diese Überlegungen werden in der folgenden Definition manifestiert.

**Definition 17 Paradigma der Service-Orientierung:** Das Paradigma der Service-Orientierung beschreibt die grundlegende Ausrichtung der Ressourcenverwaltung und der Software-Entwicklung hin zu Services. Das Paradigma umfasst dabei alle Techniken, Methoden und Komponenten, die zur Umsetzung der serviceorientierten Prinzipien und zur Realisierung einer serviceorientierten Architektur benötigt werden.

In [94] wird als Paradigma angesehen, dass eine serviceorientiert Architektur die verteilten Fähigkeiten von verschiedenen Providern organisiert und benutzbar macht. Eine derartige Auffassung des Paradigmas ist allerdings nicht so umfassend und berücksichtigt nicht alle Aspekte.

## 3.6 Service-Modell und Web Services

In diesem Abschnitt wird das Service-Modell im Zusammenhang mit der Web Services-Technologie untersucht. In Abschnitt 2.3 ist die Web Services-Technologie vorgestellt worden. Es wurde bereits darauf hingewiesen, dass Web Services Standards spezifizieren, mit denen sich zentrale Aspekte einer serviceorientierten Architektur realisieren lassen. Zentraler Bestandteil ist dabei das in Abbildung 2.3 (Seite 33) entwickelte Modell der Elemente der WSDL-Spezifikation. Das WSDL-Modell stellt vor allem die Elemente einer Web Service-Beschreibung dar, die benötigt werden, um mit einem Web Service zu kommunizieren. Weiterhin kann ein WSDL-Dokument bei einem UDDI-Broker veröffentlicht werden, wo weitere Informationen verwaltet werden können.

Tabelle 3.1 zeigt eine Übersicht über die Elemente des Service-Modells und die entsprechenden Elemente im WSDL-Modell. Einige Elemente des Service-Modells lassen sich direkt mit Elementen des WSDL-Modells vergleichen (Schnittstelle, Operation, Nachrichtentyp und Service). Die Funktionalität eines Web Services wird nicht mittels WSDL beschrieben, sondern im UDDI-konformen Broker abgelegt. Dort können standardisierte Spezifikationen von externen Gremien genutzt werden. Somit lässt sich auch hier für Web Services ein ähnliches Konzept erkennen.

Es gibt für den Service-Typ im Service-Modell keine direkte Entsprechung in Web Services. Allerdings kann ein Interface mehrfach verwendet werden. Dadurch lassen sich auf der Ebene der Kommunikation ähnliche Vorteile erzielen, wie bei der Untersuchung des Service-Modells geschildert (zum Beispiel lose Kopplung, standardisierte Schnittstellen).

Der Endpunkt des Service-Modells entspricht dem Web Services Endpoint, der ein bestimmtes Binding realisiert. Weiterhin ist ein Web Service, der mehrere Endpoints gruppiert und ein Interface implementiert, mit einem Service ver-

Service-Modell	WSDL-Modell	Bemerkung
Schnittstelle	Interface	—
Operation	Operation	—
Nachrichtentyp	Message Type	—
Funktionalität	—	In WSDL nicht vorgesehen, im UDDI-Broker textuell oder per <i>tModel-Keys</i> beschreibbar
Service-Typ	—	Interface kann von mehreren Web Services benutzt werden, daher ähnlich einem Service-Typ
Endpunkt	Binding und Endpoint	Endpoint vergleichbar mit Adresse des Endpunkts, Binding spezifiziert Protokoll
Ressource	evtl. Endpoint	Betrachtung von WSDL endet am Endpunkt der Kommunikation
Service	Web Service	—

Tabelle 3.1: Zusammenhang WSDL-Modell und Service-Modell

gleichbar, der mehrere Endpunkte gruppiert und einen Service-Typ implementiert. Für die Ressource im Service-Modell gibt es keine direkte Entsprechung im Web Services-Modell, da Web Services lediglich die Kommunikation betrachten. Lediglich die Tatsache, dass ein spezifiziertes Interface implementiert wird, ist aus Web Services-Sicht relevant. Daher kann der Endpoint auch als eine Ressource angesehen werden. Somit lässt sich eine Abbildung der Elemente des WSDL-Modells auf das Service-Modells vornehmen, wobei allerdings die beiden Abstraktionsebenen nicht explizit erkennbar sind.

### 3.7 Zusammenfassung

In diesem Kapitel ist ein Service-Modell formuliert und das Paradigma der Service-Orientierung erarbeitet worden. Das Service-Modell ist untersucht worden und die Vorteile des Paradigmas, sowie die Abstraktionsebenen sind aus den Perspektiven der Rollen untersucht worden. Neue Arten flexibler und hoch dynamischer Geschäftsanwendungen und Geschäftsprozesse werden in einer serviceorientierten Umgebung ermöglicht. Durch Technologien, die das serviceorientierte Paradigma umsetzen, können Unternehmen auch bei komplexen geschäftlichen Abläufen extrem schnell reagieren. Damit ändern sich die Grundstruktur der Geschäfte und ermöglichen neue Quantitäten und Qualitäten [139].

Das Service-Modell definiert die zu veröffentlichen Informationen der Service-Beschreibung und weitere Aspekte der Perspektiven des Requestors, Providers und Brokers. Es ist auf offene Fragestellungen hingewiesen worden und es wurde insbesondere gezeigt, wo und warum die Berücksichtigung von Qualitäts-

merkmalen nötig ist, um die Vorteile, die sich aus einer Umsetzung des Paradigmas der Service-Orientierung ergeben, vollständig ausnutzen zu können. Das Modell untermauert damit die Notwendigkeit für die Abbildung von Qualitätsmerkmalen. In dieser Arbeit soll im weiteren Verlauf dieser zentrale, noch fehlende Aspekt betrachtet werden. Im nächsten Kapitel wird ein Qualitätsmodell konstruiert, untersucht und mit dem Service-Modell in Verbindung gesetzt.



## Kapitel 4

# Qualitätsmodell

In diesem Kapitel wird ein Qualitätsmodell für Services formuliert und untersucht. Das Qualitätsmodell ist ein Modell der Eigenschaftsbeschreibungen von Services. Daher ist es wie in der Einordnung in Abschnitt 1.3, Abbildung 1.2 beschrieben, ein Meta-Modell. Als Meta-Modell liefert es einen einheitlichen konzeptionellen Rahmen für die Beschreibung von Qualitätsmerkmalen für Services. Für eine bestimmte Menge von Services wird ein Modell der Qualitätsmerkmale erstellt, so dass die Ausprägungen der Qualitätsmerkmale eines Services präzise und eindeutig spezifiziert werden können. Services sollen bezüglich der Ausprägungen der Qualitätsmerkmale vergleichbar sein. Ohne die konzeptionelle Unterstützung eines Qualitätsmodells ist für einen Requestor nur mit relativ großem manuellen Aufwand entscheidbar, welchen der Services er tatsächlich aufrufen soll. Ziel des Requestors ist es, den Service auszuwählen, der aus seiner Sicht die beste Qualität hat; daher wird das Modell der Eigenschaftsbeschreibungen von Services hier als Qualitätsmodell bezeichnet. Das Qualitätsmodell baut auf dem Service-Modell des vorherigen Kapitels auf und unterscheidet insbesondere auch zwischen Service-Typ und Service.

In Abschnitt 4.1 werden für zwei der bereits vorgestellten Beispiel-Szenarien Qualitätsmerkmale skizziert, die im weiteren Verlauf des Kapitels verwendet werden. In Abschnitt 4.2 werden die Annahmen und die Anforderungen an das Modell erarbeitet. Das Qualitätsmodell wird in Abschnitt 4.3 formuliert und in das Service-Modell integriert. Eine mögliche Umsetzung des Qualitätsmodells wird in Abschnitt 4.4 skizziert, wobei die Qualitätsmerkmale für einen Beispiel-Service beschrieben werden (Modellebene in Abbildung 1.2). Mögliche Erweiterungen werden in Abschnitt 4.5 diskutiert und eine Zusammenfassung in Abschnitt 4.6 schließt das Kapitel. Das Qualitätsmodell wird dann im Kapitel 5 im Kontext von Geschäftsprozessen weiter untersucht und durch eine praktische Umsetzung validiert.



## 4.1 Beispiele

Die Bedeutung des Qualitätsmodells für serviceorientierte Umgebungen wird zunächst durch zwei Beispiel-Szenarien hervorgehoben. Abschnitt 4.1.1 beschreibt ein Szenario für digitale Produkte [168], indem die relevanten Qualitätsmerkmale für Video-Services betrachtet werden. Abschnitt 4.1.2 skizziert die Sicht eines Provider, der Transport-Dienstleistungen als Services anbietet und beschreibt, wie die angebotenen Services von Dritten in einem Geschäftsprozess integriert werden können. Um möglichst unterschiedliche Qualitätsmerkmale betrachten zu können und um potentielle Missverständnisse zu illustrieren, sind beide Szenarien nötig. Außerdem werden durch die Beispiele einige elementare Anforderungen an das Qualitätsmodell erarbeitet.

### 4.1.1 Video-Service

Das bereits in der Einleitung in Abschnitt 1.1 skizzierte Beispiel eines *Video-Services* wird hier aufgegriffen und detaillierter betrachtet. Ein Provider bietet in einer serviceorientierten Umgebung einen Service für Video-Streams aktueller Spielfilme in verschiedenen digitalen Formaten an [95]. Kunden können ein Video auswählen und es gegen Bezahlung anschauen.

**Qualitätsmerkmale** Die Qualitätsmerkmale für einen Service sollen in dessen Beschreibung angegeben werden. Es gibt nicht einen Wert für die Qualität, sondern mehrere Qualitätsmerkmale, die der Requestor aus den Daten der Service-Beschreibung als solche erkennen muss. Ebenso muss der Requestor die jeweiligen Ausprägungen erkennen können. Tabelle 4.1 zeigt einen intuitiven Überblick über die relevanten Qualitätsmerkmale dieses Szenarios. Die Tabelle zeigt einen eindeutigen Namen für das Qualitätsmerkmal (hier ein englischer Begriff) und eine Erklärung der Bedeutung. Teilweise sind bereits Maßeinheiten wie € und mögliche Werte angegeben. Neben den Kosten (*Price*) hier vor allem die Darstellungsqualität des Videos betrachtet. Im Folgenden seien hier die Qualitätsmerkmale betrachtet und jeweils auf Probleme bei deren Interpretation hingewiesen. Es wird dabei aufgezeigt, inwiefern die intuitive Beschreibung der Qualitätsmerkmale unpräzise ist und zu Missverständnissen führen kann.

**Datentyp und Maßeinheit** Eines der Qualitätsmerkmale sind die Kosten für den Requestor, die beim Aufruf des Services entstehen. Das Qualitätsmerkmal Kosten ist für viele Arten von Services relevant und findet sich in vielen Beispielen in der Literatur [31, 72, 90, 112, 185].

Wie in der Tabelle beschrieben, werden die Kosten (*Price*) als Zahl mit zwei Nachkommastellen und in der Währung Euro (€) angegeben. Es ist also der Datentyp der Werte spezifiziert und eine Maßeinheit festgelegt. Ebenso kann man aus der Tabelle schließen, dass jedes Video den gleichen Preis hat. Dennoch ist eine Angabe beispielsweise von 4,99 € noch nicht vollständig.

Qualitätsmerkmal	Erklärung der Bedeutung
Price	Preis pro Video, angegeben mit zwei Nachkommastellen in der Währung €
FrameRate	Bildfrequenz, garantierter Mindestwert der Bilder pro Sekunde
ColorDepth	Farbtiefe in Bits oder Anzahl Farben. Mögliche Werte: 8, 16, 32 (Bits) bzw. 256, 65536, ca. 4,3 Mio (Anzahl verschiedenen Farben)
Resolution	Auflösung des Bildes, angegeben in Pixel×Pixel, also Anzahl Bildpunkte pro Spalte (vertikal) und Zeile (horizontal). Beispiele: 1024×768, 1280×720 Alternativ: Name der Auflösung als feste Bezeichner, zum Beispiel wie <i>VGA</i> für 640×480 und <i>SVGA</i> für 800×600
Codec	Standardisierter Name des Verfahrens, in dem das Video kodiert ist; zulässige Werte: <i>WMV</i> , <i>Real</i> und <i>MPEG</i>

Tabelle 4.1: Intuitive Darstellung der relevanten Qualitätsmerkmale für Video-Services

**Semantik der Werte** Neben den Informationen der Tabelle muss spezifiziert werden, dass der Preis jeweils für einen Service-Aufruf, also das einmalige Ansehen des Videos, gilt und nicht etwa für einen wiederholten Aufruf des Videos. Ebenso ist zu spezifizieren, dass dieser Preis inklusive Umsatzsteuer angegeben wird. Der gültige Wertebereich kann eingeschränkt werden, indem nur positive Zahlenwerte zugelassen werden. Auch wenn eine Währung festgelegt wurde, ist für einen Requestor nicht ersichtlich, wie der Euro-Wert in eine andere Währung umzurechnen ist. Aus den bisherigen Überlegungen lassen sich die folgenden Anforderungen an die Modellierung von Qualitätsmerkmalen ableiten:

- Für einen Service sind Ausprägungen mehrerer Qualitätsmerkmale anzugeben, die für einen Requestor erkennbar sein müssen.
- Der Datentyp beziehungsweise der gültige Wertebereich ist für jedes Qualitätsmerkmal zu spezifizieren.
- Für die Ausprägungen kann eine Maßeinheit definiert werden. Dazu sind unter Umständen zusätzliche Informationen notwendig, beispielsweise Wechselkurse.
- Die Bedeutung eines Qualitätsmerkmals ist eindeutig und präzise zu formulieren, so dass es nicht zu Missverständnissen kommen kann, beispielsweise ist zu spezifizieren, dass die Kosten die Umsatzsteuer enthalten.

**Alternative Angabe der Werte und Definition des Wertebereiches** In Tabelle 4.1 ist die Bildauflösung (*FrameRate*) des Videos durch standardisierte Namen angegeben, wie beispielsweise *VGA*, *SVGA* und *XGA*. Die Namen repräsentieren eine Bildauflösung in Pixel×Pixel von 640×480, 800×600 beziehungsweise

1024×768. Problematisch ist hierbei, dass diese Zuordnung nicht eindeutig ist, denn XGA kann auch als 1152×870 interpretiert werden. Somit kann es bei der Verwendung von Namen zu Missverständnissen kommen, so dass das angeforderte Video nicht den Anforderungen des Requestors entspricht.

Als Alternative kann die Bildauflösung auch direkt in Pixels angegeben werden. Allerdings sind hier nicht alle Angaben zulässig, da das Seiten-Verhältnis 4:3 oder 16:9 sein sollte. Es kann also sinnvoll sein, den Wertebereich durch zusätzliche Regeln einzuschränken. Sollen auch abweichende Bildauflösungen und Seitenverhältnisse beispielsweise für kleinere, mobile Abspielgeräte angeboten werden, ist eine komplette Angabe aller unterstützten Bildauflösungen notwendig. Je nach Anwendungsgebiet ist hier eine geeignete Definition festzulegen.

Das Qualitätsmerkmal Farbtiefe (*ColorDepth*) gibt die Anzahl der unterschiedlichen Farben des Video-Streams an. Angegeben wird die Farbtiefe in Anzahl der Bits zur Farbcodierung, übliche Werte sind 8, 16 oder 32 Bits. Alternativ kann die Farbtiefe auch direkt angegeben werden, beispielsweise 256, 65536 oder 4,3 Millionen verschiedene Farben. In der Tabelle 4.1 sind alternative Möglichkeiten dargestellt, um dasselbe auszudrücken, wobei der Datentyp (hier ganzzahlige Werte) derselbe ist. Ebenso ist zu bedenken, dass für digitale Videos nur wenige Werte überhaupt sinnvoll sind. Es ergeben sich folgende weitere Erkenntnisse:

- Es kann verschiedene Werte für ein Qualitätsmerkmal geben, die dasselbe auszudrücken. Die Semantik der Qualitätsmerkmale ist diesbezüglich eindeutig zu definieren.
- Neben einer Definition des Wertebereichs durch einen Datentyp oder Einschränkungen (nur positive Werte) können auch direkt die erlaubten Werte aufgezählt werden.
- Wenn als Werte Namen verwendet werden, so ist die Bedeutung präzise und eindeutig zu spezifizieren, für die Bildauflösung beispielsweise durch eine Zuordnung der Namen zu einer Angabe Pixel×Pixel.
- Wenn die Ausprägungen direkt in Pixel×Pixel angegeben werden (also zwei Zahlenwerte), dann ist nicht jede Kombination zulässig. Es sind dann entweder Regeln für den Wertebereich zu definieren oder alle zulässigen Werte anzugeben.

**Richtung der Qualitätssteigerung** Betrachtet man die Kosten, so ist offensichtlich, dass ein Requestor niedrigere Kosten bevorzugt. Aus Sicht eines Requestors ist es prinzipiell ebenfalls vorteilhaft, das Video in einer möglichst hohen Bildauflösung zu erhalten. Somit steigt die Qualität in Richtung der steigenden Bildauflösung. Allerdings ist die Obergrenze seine eigene Darstellungsmöglichkeit. Ebenso ist das Seitenverhältnis seines Abspielgerätes zu berücksichtigen, so dass nicht jede größere Auflösung eine bessere Darstellung ergibt.

Aus Sicht eines Requestors ist eine höhere Farbtiefe besser. Somit ist ein Service, der Videos mit 32 Bit Farbtiefe anbietet, einem Service mit 16 Bit vorzuziehen. Beim Wert 32 ist davon auszugehen, dass die Angabe in Bits erfolgt, beim Wert 256 ist die Anzahl verschiedener Farben gemeint. Somit ist der Wert 256 „schlechter“ also der Wert 32. Das Potential für Missverständnisse ist hier offensichtlich, daher ist eine derartige Spezifikation eines Qualitätsmerkmals nicht vorteilhaft.

- Es kann eine Richtung geben, in der die Qualität für alle Requestoren steigt: niedrigere Kosten sind besser.
- Diese Richtung kann Grenzwerte aufweisen: Beispielsweise ist eine größere Bildauflösung besser, aber für einen Requestor kann es Grenzen der Darstellungsmöglichkeit geben, so dass eine eigentlich höhere Qualität für ihn schlechter oder sogar unbrauchbar ist.
- Es kann also von den Anforderungen des Requestors abhängen, in welcher Richtung die Qualität steigt beziehungsweise welche Werte die höchste Qualität aufweisen.

**Qualitätsmerkmale ohne Ordnung** Die Eigenschaft *Codec* steht für das Format in dem die Videos codiert sind. In Tabelle 4.1 sind drei Namen als Standard festgelegt. Während Farbtiefe und Auflösung bei steigenden Werten eine bessere Qualität bedeuten, kann eine solche Aussage bei den Video-Codecs nicht getroffen werden. Entgegen den anderen Merkmalen ist der Codec nicht nur ohne besondere Richtung in der die Qualität steigt, sondern auch ohne Ordnung der Werte. Dies führt zu den folgenden Erkenntnissen:

- Es gibt Qualitätsmerkmale, für die kein metrischer Wert angegeben werden kann, sondern nur eine Vergabe von Namen möglich ist.
- Es gibt Qualitätsmerkmale, auf deren Werte keine Ordnung definiert ist; daraus folgt auch:
- Es gibt Qualitätsmerkmale, für die es per se keine Richtung der Qualitätssteigerung gibt.

**Feste Werte, Grenzwerte oder Erwartungswerte** Die *Bildfrequenz* gibt die Anzahl von Bildern pro Sekunde an. Hier gilt: Je höher der Wert, desto besser die Qualität. Allerdings ist hierbei ähnlich wie bei der Auflösung Obergrenze für einen Requestor zu beachten. Zusätzlich ist dieser Wert kein fester Wert: Während Bildauflösung und Farbtiefe feste Werte darstellen, die für alle Service-Aufrufe gleich bleiben, kann die Bildfrequenz schwanken. Hier ist zu spezifizieren, ob ein angegebener Wert einen garantierten Mindestwert darstellt, die im Idealfall erreichbare maximale Bildfrequenz oder einen mittleren Erwartungswert.

- Die angegebene Ausprägung des Qualitätsmerkmals kann einen garantierten Maximal- oder Minimalwert repräsentieren oder einen festen Wert.
- Wird das Qualitätsmerkmal als Grenzwert spezifiziert, weicht die beobachtete Ausprägung in der Regel bei der Leistungserbringung vom veröffentlichten Wert ab.
- Wird ein erwarteter Mittelwert angegeben, ist die genaue Bedeutung des Wertes zu spezifizieren. Es kann beispielsweise eine durchschnittliche Abweichung von dem Wert angegeben werden (Streuung) oder die maximale Abweichung.

**Zusammenfassung und Diskussion** Es ist gezeigt worden, dass die intuitive Darstellung des Qualitätsmerkmals in Tabelle 4.1 nicht ausreichend ist. Weiterhin wurde aufgezeigt, dass die Qualitätsmerkmale sehr unterschiedlicher Art und die Werte unterschiedlich zu interpretieren sind. Nur wenn die Qualitätsmerkmale präzise und vollständig beschrieben werden, können Mehrdeutigkeiten und Missverständnisse verhindert werden. Eine konzeptionelle Grundlage für die Modellierung von Qualitätsmerkmalen kann eine präzise und vollständige Beschreibung der Semantik der Qualitätsmerkmale und der Werte unterstützen.

Der Provider möchte eine Service-Beschreibung veröffentlichen und dabei angeben, welche Ausprägungen der Qualitätsmerkmale bei der Leistungserbringung möglich sind. Es sollen beispielsweise die unterstützten Codecs angegeben werden, mögliche Auflösungen sowie eine garantierte Bildfrequenz und die maximale Farbtiefe. Ein Requestor soll dann anhand der veröffentlichten Informationen entscheiden, ob er den angebotenen Service nutzen kann. Wenn mehrere Video-Services von verschiedenen Providern angeboten werden, muss der Requestor entscheiden können, welcher Service aus seiner Sicht die beste Qualität hat.

In einem auf Web Services basierenden IT-System gibt es bislang keine konzeptionelle Grundlage, diese unterschiedlichen Arten der Qualitätsmerkmale als Teil der Service-Beschreibung zu veröffentlichen. Wie bereits in Abschnitt 2.3 angesprochen, besteht nur eine unzureichende Möglichkeit, bei einem UDDI-konformen Service Broker Informationen zu hinterlegen. Ohne eine konzeptionelle Grundlage gleicht eine Umsetzung in UDDI einer textuellen Beschreibung, die nur durch menschliche Experten ausgewertet werden kann. Wie hier illustriert wurde, kann es dabei zu Missverständnissen kommen. Wenn jeder Web Service eine eigene, proprietäre Beschreibung der Qualität veröffentlichen, ist ein automatisierter Vergleich der Qualität nicht möglich. Ein Requestor hat dann nicht die Möglichkeit, vom Broker zu erfahren, welcher der verfügbaren Video-Services eine von ihm benötigte Qualität bereitstellen kann. Ein Requestor muss während der Service-Auswahl mit den einzelnen Providern Kontakt aufnehmen und sich über die Ausprägungen der Qualitätsmerkmale informieren und diese mit seinen Anforderungen vergleichen.

### 4.1.2 Transport-Service

Das Transport-Szenario ist ebenfalls in Abschnitt 1.1 eingeführt worden und betrachtet einen Transport-Dienstleister. Die Dienstleistung unterschiedliche Güter zu transportieren wird als Web Service angeboten.

**Serviceorientierte Umgebung** Die Kommunikation in der serviceorientierten Umgebung wird durch Web Services realisiert. Dazu wird vom Provider je angebotenen Service ein Web Service auf einem Application Server angemeldet und aktiviert. Auf das Service-Modell übertragen, ist der Endpunkt die beim Application Server aktivierte Web Service Implementierung. Dieser Web Service realisiert eine (standardisierte) Schnittstelle und empfängt die entsprechenden Nachrichten. Die konkreten WSDL-Dokumente des aktivierten Web Services und eine genauere Beschreibung der Kommunikation seien hier nicht betrachtet.

Das Transportunternehmen bietet sechs unterschiedliche Services ( $S_1$  bis  $S_6$ ) als sechs Web Services an. Der Service-Typ  $T_{shipping}$  sei in der Transport-Branche standardisiert worden. Im Sinne des Service-Modells implementieren die sechs Services alle diesen Service-Typ. Die sechs Web Services unterscheiden sich in den Qualitätsmerkmalen, beispielsweise schnelle oder langsame Lieferung, unterschiedliche Kosten oder verschlüsselte Kommunikation. Tabelle 4.2 zeigt eine informale Übersicht über die Services  $S_1$  bis  $S_6$  mit Ausprägungen der Qualitätsmerkmale.

**Überblick über die Qualitätsmerkmale** Die *Temperatur* wird angegeben in °C und stellt eine Garantie Obergrenze für den Transport dar. Die *Kosten* stellen einen festen Betrag dar, für den der Transport durchgeführt wird. Kosten werden als Zahlenwert in der Währung € angegeben. Die *Lieferzeit* ist in vollen Stunden angegeben. Die Bedeutung sei hier so definiert, dass die Lieferzeit eine vom Provider garantierte höchste Zeitdauer darstellt. Bei *Sicherheit* ist für die Services  $S_3$ ,  $S_4$ ,  $S_5$  und  $S_6$  jeweils die Länge des verwendeten Schlüssels für die Kommunikation angegeben. Dabei wird neben dem Wert als Maßeinheit „bit key“ angegeben. Die *Statusüberwachung* gibt an, wie der Zustand eines beauftragten Transports ermittelt werden kann. Die Werte in der Spalte Statusüberwachung der Tabelle 4.2 stehen als Namen für die verfügbaren Techniken.

**Qualitätsmerkmale ohne Ausprägung** Bei der Temperatur scheinen Maßeinheit und Semantik festgelegt zu sein. In Tabelle 4.2 ist dies ausgedrückt, indem die Maßeinheit angegeben wird und den Werten jeweils ein  $\leq$  vorangestellt wird. Die Temperatur ist aber nicht für alle Services angegeben, denn für  $S_1$  und  $S_3$  sind keine Werte angegeben. Daher sei zusätzlich definiert, dass ein fehlender Wert bei einem nicht klimatisierten Transport verwendet wird. Daher kann der Provider keine Ober- oder Untergrenze für die Temperatur garantieren. Wenn eine Rangfolge der Services bezüglich der Temperatur erstellt werden soll, so können diese Services

Service	Temperatur	Kosten	Lieferzeit	Sicherheit	Statusüberwachung
$S_1$	-	100 €	48 Std.	-	Telefon
$S_2$	$\leq -8^\circ\text{C}$	275 €	24 Std.	-	Telefon
$S_3$	-	150 €	24 Std.	128 bit key	E-Mail
$S_4$	$\leq +20^\circ\text{C}$	320 €	18 Std.	128 bit key	E-Mail
$S_5$	$\leq +4^\circ\text{C}$	400 €	36 Std.	512 bit key	Web-Portal
$S_6$	$\leq -4^\circ\text{C}$	450 €	48 Std.	512 bit key	Web-Portal

Tabelle 4.2: Übersicht über die angebotenen Transport-Services und die Ausprägungen der Eigenschaften

nicht eingeordnet werden. Hat ein Requestor Anforderungen an die Temperatur, so kann ein Service ohne Temperatur-Angabe nicht berücksichtigt werden.

**Requestor-Anforderungen** Ein Requestor benötige einen Service, der verderbliche Lebensmittel transportiert. Seine Anforderung sei, dass die Ware immer gefroren bleibt, also unter  $0^\circ\text{C}$ . Selbst kurzfristiges Überschreiten der  $0^\circ\text{C}$ -Grenze würde die Lebensmittel verderben lassen. Für ihn ist es also wichtig, dass die garantierte Temperatur bei  $S_2$  und  $S_6$  dieses gewährleistet und nicht etwa eine kurzfristige Verletzung des Maximalwertes toleriert wird.

Ein anderer Requestor möchte Ware transportieren, die zwar gekühlt werden soll, aber nicht eingefroren werden darf. Diese Anforderung kann durch die Qualitätsmerkmale nicht abgedeckt werden, da die Temperatur einen Maximalwert darstellt. Eine garantierte Untergrenze der Temperatur ist nicht als Qualitätsmerkmal vorgesehen. Welche Informationen als Qualitätsmerkmal zur Verfügung gestellt werden hängt vom jeweiligen Anwendungsgebiet ab und ist durch die Teilnehmer der serviceorientierten Architektur zu spezifizieren. Folgende Fragestellungen können beispielsweise beachtet werden:

- Ist die Angabe der Ausprägung für alle Qualitätsmerkmale Pflicht oder soll es möglich sein, für einzelne Qualitätsmerkmale keine Ausprägungen anzugeben? Falls dies erlaubt ist: Wie werden diese unspezifizierten Ausprägungen mit spezifizierten Ausprägungen anderer Services verglichen? Ist ein Standardwert anzunehmen?
- Stellen die spezifizierten Werte feste Werte, Obergrenzen, Untergrenzen oder Mittelwerte dar? Falls es sich um Mittelwerte handelt: Sind die möglichen (maximalen) Abweichungen relevant und müssen angegeben beziehungsweise eingeschränkt werden?
- Mit welchen Verfahren und zu welchen Zeitpunkten werden die Werte der Qualitätsmerkmale gemessen beziehungsweise überprüft?

Wenn eine Ordnung der Ausprägungen für ein Qualitätsmerkmal angegeben werden kann, so ist die Angabe von minimalen Anforderungen unter Umständen

nicht ausreichend. Sei von einem Requestor gefordert, dass die Kommunikation mindestens mit einem 128-bit Schlüssel verschlüsselt wird. Der Requestor unterstütze sowohl 128-bit Schlüssel, als auch 256-bit Schlüssel. Die Services  $S_5$  und  $S_6$  kann er allerdings nicht nutzen. Somit muss seine Anforderung sich explizit auf die unterstützten Ausprägungen dieses Qualitätsmerkmals beziehen. In der Regel muss ihm dazu der gesamte Wertebereich bekannt sein.

**Alternative Maßeinheiten** Da Transport-Services international angeboten werden sollen, seien auch andere Währungen für die Kosten sowie andere Temperaturskalen erlaubt. Es sind also für die Qualitätsmerkmale Temperatur und die Kosten andere Maßeinheiten wie Grad Fahrenheit ( $^{\circ}\text{F}$ ) beziehungsweise Britisches Pfund (£) oder US-Dollar (\$) denkbar. Entweder sind daher die Maßeinheiten bei jeder spezifizierten Ausprägung eines Service anzugeben, oder eine Maßeinheit ist als Standard zu definieren, so dass Angaben beispielsweise in € umgerechnet werden müssen.

**Zahlenwerte ohne Verhältnis** Beim Qualitätsmerkmal Sicherheit kann ein 512-bit Schlüssel nicht als vier Mal sicherer angesehen werden kann als ein 128-bit Schlüssel, das heißt, das Verhältnis der Sicherheit entspricht nicht dem Verhältnis der Zahlenwerte. Dennoch ist ein längerer Schlüssel prinzipiell sicherer, so dass eine Ordnung für die Werte der Tabelle angegeben werden kann. Wird kein Wert angegeben ( $S_1$  und  $S_2$ ), so wird keine Verschlüsselung verwendet. Im Vergleich zu den anderen Services sind diese von geringerer Qualität. Hier können fehlende Werte also mit anderen Ausprägungen verglichen werden.

**Einbindung des Services in einen Geschäftsprozess** Der Transport-Service soll nun in einem Geschäftsprozess integriert werden, der aus vier Prozessschritten besteht. Dieser Prozess ist in Abbildung 4.1 in der graphischen Beschreibungssprache Business Process Modeling Notation (BPMN) [107] dargestellt. Als erstes wird der Auftrag eines Kunden entgegengenommen (*Auftragsannahme*). Anschließend wird der Auftrag ausgeführt, in dem die Produktion durchgeführt wird (*Produktion*). Zum Abschluss wird dann parallel die Abrechnung durchgeführt (*Abrechnung*) sowie das Produkt zum Kunden transportiert (*Transport*).

Als Service-Typ für den Prozessschritt *Transport* sei  $T_{shipping}$  festgelegt worden. Nun wird aus den Transport-Services  $S_1$  bis  $S_6$  ein geeigneter Service ausgewählt, der während der Prozessausführung aufgerufen werden soll. Der Geschäftsprozess soll wiederum als zusammengesetzter Service angeboten werden (siehe Abschnitt 3.4.3). Die Gesamtkosten dieses zusammengesetzten Services hängen auch vom gewählten Transport-Service ab, dessen Ausprägungen in Tabelle 4.2 zwischen 100 und 450 € schwanken. Somit beeinflussen die Ausprägungen der Qualitätsmerkmale den Geschäftsprozess. Als weiteres Beispiel sei die Ausführungsdauer des Geschäftsprozesses betrachtet: Der Prozess kann erst beendet werden, wenn alle Prozessschritte beendet sind. Da der Transport-Service parallel mit



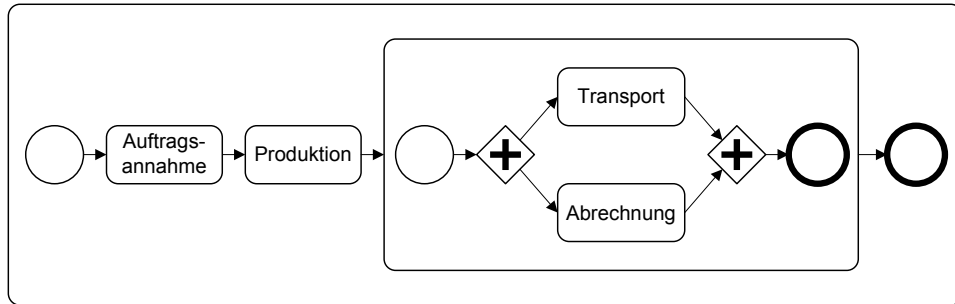


Abbildung 4.1: Beispiel-Geschäftsprozess: Produktionprozess, dargestellt in BPMN

der Abrechnung der letzte Prozessschritt ist, wirkt sich die Lieferzeit des Transport-Services auf die Ausführungsdauer aus, wenn er länger dauert als die Abrechnung. Dazu müssen die Informationen über die beiden Prozessschritte verglichen werden und ermittelt werden, welcher der Prozessschritte länger dauert. Dieser Wert ist dann die Ausführungsdauer der beiden Prozessschritte. Für derartige Überlegungen ist eine konzeptionelle Grundlage für Qualitätsmerkmale Voraussetzung.

Zusammengesetzte Services, die Geschäftsprozesse realisieren, werden in Kapitel 5 detailliert betrachtet. An dieser Stelle sei darauf hingewiesen, dass sich die Auswahl des Services auf den Ablauf des Geschäftsprozesses auswirkt. Um die Auswirkung zu ermitteln, wird eine konzeptionelle Grundlage für Qualitätsmerkmale von zusammengesetzten Services benötigt.

**Zusammenfassung und Diskussion** Es ist in diesem Szenario aufgezeigt worden, dass entweder unterschiedliche Maßeinheiten berücksichtigt werden müssen oder dass eine Maßeinheit als Standard definiert werden sollte. Ferner muss die Bedeutung der Werte genau spezifiziert werden und dabei muss beachtet werden, was der Requestor in diesem Anwendungsgebiet für Anforderungen stellen kann. Die Anfragen eines Requestors müssen so exakt spezifiziert werden, dass keine Services ermittelt werden, die er nicht nutzen kann. Weiterhin ist anhand des Beispiels Schlüssellänge erläutert worden, dass die Bedeutung von Ziffern als Ausprägung nicht immer dem Zahlenwert entsprechen muss. Es ist weiterhin aufgezeigt worden, dass sich die Qualitätsmerkmale eines Services auf zusammengesetzte Services auswirken können. Dazu sind die Informationen über die einzelnen Prozessschritte unter Umständen geeignet zu verdichten. Es muss dazu mit den Ausprägungen der Qualitätsmerkmale gerechnet werden.

## 4.2 Annahmen und Anforderungen

Die Notwendigkeit einer konzeptionellen Grundlage für ein Qualitätsmodell wurde in den vorherigen Kapiteln und durch die Beispiele herausgearbeitet. In diesem

Abschnitt werden diese Anforderung zusammengefasst und typische Anwendungsszenarien abgeleitet.

Grundsätzlich wird das Ziel verfolgt, Angebot und Nachfrage in einer serviceorientierten Architektur geeignet zusammenzuführen [161, 42]. Dazu soll ein einheitliches Konzept auf der Basis eines formalen Meta-Modells für Qualitätsmerkmale geschaffen werden. Mit dem Meta-Modell kann jeweils für ein Anwendungsgebiet ein Modell der Qualitätsmerkmale spezifiziert werden. Die Qualitätsmerkmale können dann als Standard für das Anwendungsgebiet angesehen werden. Das Qualitätsmodell soll auf dem Service-Modell aus Kapitel 3 aufbauen. Der Service eines Providers kann dann präzise beschrieben und angeboten werden und der Requestor kann seine Anfragen präzise formulieren. Es ist darüber hinaus möglich, unter den verfügbaren Services automatisiert den Besten auszuwählen. Dabei wird von folgenden Annahmen ausgegangen:

1. Es gelten die Annahmen, die für das Service-Modell in Abschnitt 3.1 spezifiziert wurden. Insbesondere ist hier hervorzuheben, dass ein Provider Services anbietet und dazu Beschreibungen der Services bei einem Broker veröffentlicht. Außerdem ist die Funktionalität für einen Requestor erkennbar, so dass er Service-Typen und Services finden kann, die seinen funktionalen Anforderungen entsprechen.
2. Ein Provider will seinen Service mit Informationen über die Ausprägungen definierter Qualitätsmerkmale veröffentlichen. Er kann dazu die tatsächlichen Eigenschaften ermitteln, die bei einer Leistungserbringung auftreten und die Ausprägungen entsprechend formulieren.
3. Ein Requestor kann in seinen Anfragen an den Broker seine gewünschten Ausprägungen für Qualitätsmerkmale angeben und will den am besten geeigneten Service aufrufen.

Zusammenfassend lässt sich festhalten: Das Qualitätsmodell muss gewährleisten, dass Requestor und Provider dasselbe Verständnis der Bedeutung der Qualitätsmerkmale haben und Missverständnisse verhindern. Ebenso müssen die Bedeutungen der Werte, also der Ausprägungen der Qualitätsmerkmale, übereinstimmen. Es muss also spezifiziert werden, wie und wann die Ausprägungen der Eigenschaften beobachtet oder gemessen werden. Dazu ist es notwendig, dass die Eigenschaften objektiv messbar sind und inhaltlich gleich verstanden werden, also die gleiche Bedeutung bei allen Beteiligten haben [91, 92].

### 4.2.1 Allgemeine Anforderungen

#### **Provider**

Grundsätzlich soll es möglich sein, mehrere Eigenschaften für einen Service anzugeben. Der Provider gibt diese Eigenschaften bei der Veröffentlichung der Service-Beschreibung bekannt. Er sollte dazu jeweils einen Namen für die Eigenschaft und

einen Wert angeben. In Anlehnung an den aus Programmiersprachen bekannten Konzepten, sind die Datenstruktur und der Datentyp zu definieren. Der Provider sollte auch Wertebereiche angeben können oder Grenzwerte festlegen können, die er zusichert. Die Angaben des Providers stellen zunächst ein Angebot dar. Weiterhin ist es denkbar, dass ein Provider funktional äquivalente Services mit jeweils unterschiedlichen Ausprägungen der nicht-funktionalen Eigenschaften anbietet. Dieses ist in dem Transport-Beispiel durch die Services  $S_1$  bis  $S_6$  gegeben.

### Requestor

Der Requestor muss eine Anfrage so formulieren können, dass er seine Anforderungen an die Eigenschaften abbilden kann. Dabei können Einschränkungen des Wertebereichs (beispielsweise Mindest- und/oder Maximalwerte) relevant sein. Zusätzlich kann ein Optimierungskriterium, beispielsweise minimale Kosten, angegeben werden. Er muss seine Restriktionen und Beschränkungen präzise und eindeutig formulieren können. Die Anforderungen an verschiedene Qualitätsmerkmale sollen beliebig kombinierbar sein; bezogen auf den Transport-Service kann eine Anfrage beispielsweise wie folgt aussehen:

$$R_1 = \text{Gesucht wird ein Service, der einen Transport anbietet,} \quad (4.1)$$

$$\text{mit möglichst geringen Kosten bis maximal 200 €} \quad (4.2)$$

$$\text{und Lieferzeit maximal 48 Stunden.} \quad (4.3)$$

In dem ersten Teil der Anfrage (4.1) ist die Funktionalität angegeben, im zweiten (4.2) und dritten Teil (4.3) Anforderungen an die Qualität. Es liegen dabei folgende Einschränkungen des Wertebereichs vor: Kosten unter 200 € (4.2) und Lieferzeit  $\leq 48$  Stunden (4.3). Als Optimierungskriterium ist „möglichst geringe Kosten“ angegeben. Das Ziel ist es, eine Rangfolge zu ermitteln und den besten Service auszuwählen.

### Broker

Die Eigenschaften eines angebotenen Services sollen vom Service Broker zusammen mit der Service-Beschreibung verwaltet werden. Sie können dabei auch direkt als Bestandteil der Service-Beschreibung in einem Dokument angegeben werden. Bei entsprechenden Anfragen eines Requestors muss der Broker dessen Anforderungen berücksichtigen und passende Services finden:

1. Wenn die Funktionalität in der Anfrage festgelegt ist, so sollen in einem ersten Schritt die passenden Service-Typen ermittelt werden. Im Folgenden dürfen nur Services eines passenden Service-Typs berücksichtigt werden.
2. Der Broker soll dann eine Rangfolge der passenden Services erstellen und diese dem Requestor zusenden. Der Broker muss dazu die Werte der Qualitätsmerkmale vergleichen können.

### 4.2.2 Anwendungsfälle

Das zu formulierende Qualitätsmodell muss die folgenden Anwendungsfälle unterstützen. Es wird aufgezeigt, dass sich eine konzeptionelle Grundlage zur Abbildung der Qualitätsmerkmale besonders positiv auswirkt auf diese Anwendungsfälle auswirkt [111].

#### Suche und Auswahl

Zu Beginn dieser Arbeit wurde in Kapitel 2 in Abbildung 1.1 die Interaktion *find* dargestellt. Hierbei handelt es sich um einen Vorgang, bei dem ein Service entdeckt werden soll; in der englischsprachigen Literatur wird dieser Vorgang daher als *Service Discovery* bezeichnet [116]. Der Vorgang beginnt mit der Anfrage eines Requestors, der einen Service sucht. Dazu wird eine Anfrage in der Sprache des relevanten Brokers formuliert. Grundsätzlich sucht ein Requestor zunächst nach einer bestimmten Funktionalität. Über die Service-Typen, denen die Funktionalität zugeordnet ist, können vom Broker entsprechende Services ermittelt werden. Ist ein Service-Typ bekannt, so kann ein Requestor direkt nach Services dieses Typs suchen.

Bei einer Anfrage nach dem Service-Typ  $T_{shipping}$  (bezogen auf das Transport-Beispiel in Abschnitt 4.1.2), seien alle Services von  $S_1$  bis  $S_6$  in der Ergebnismenge. Der Requestor muss entscheiden, welchen der Services er nutzen möchte. In einem ersten Schritt sind feststehende Einschränkungen zu berücksichtigen; dies können Budget-Restriktionen sein oder terminliche Einschränkungen. Wenn beispielsweise gefordert ist, dass der Service nicht mehr als 420 € Kosten darf und eine verschlüsselte Kommunikation gefordert ist, lautet die (informale) Anfrage:

$$R_2 = \text{Ein Service, der den Service-Typs } T_{shipping} \text{ implementiert,} \\ \text{Kosten} \leq 420 \text{ € hat und einen Wert für Sicherheit hat.}$$

Dadurch bleiben von den hier betrachteten Services nur die Services  $S_3$ ,  $S_4$  und  $S_5$  übrig, da  $S_1$  und  $S_2$  keinen Wert für das Qualitätsmerkmal Sicherheit haben und  $S_6$  zu hohe Kosten hat. Es scheint nun nahe liegend, dass der günstigste der Services gewählt ( $S_3$ ) wird. Allerdings setzt dieses voraus, dass es ein gewünschtes Optimierungsziel des Requestors ist, die Kosten so gering wie möglich zu halten. Service  $S_3$  ist zwar mit 150 € der mit den geringsten Kosten, allerdings verfügt  $S_5$  (400 €) über eine bessere Verschlüsselung durch einen längeren Schlüssel (512-bit gegenüber 128-bit). Wie Tabelle 4.2 zeigt, unterscheiden sich die beiden Services auch in anderen Qualitätsmerkmalen. Ohne weitere Informationen über die Anforderungen und Optimierungsziele des Requestors ist eine Entscheidung also nicht zu treffen. Durch die Einbeziehung der Qualitätsmerkmale ist dennoch grundsätzlich eine bessere Entscheidung möglich. Liegen durch die Anfrage ausreichende Informationen vor, so kann direkt der ideale Service durch den Broker vermittelt werden. In der Anfrage  $R_2$  reicht der Zusatz der Art *minimiere die Kosten* aus, um eine Rangfolge zu erstellen:  $S_3$ ,  $S_4$ ,  $S_5$ .

### Verhandlungen

Aus betriebswirtschaftlicher Sicht ist es denkbar, dass vor dem Aufruf eines Services zwischen Provider und Requestor die Konditionen der Ausführung wie beispielsweise Preis und Zeitdauer verhandelt werden. Verhandlungen wurden bereits im Rahmen der Kommunikationsprinzipien in Abschnitt 3.4.2 erwähnt. Allgemein gilt dieses für alle Qualitätsmerkmale. Es startet somit eine Verhandlungsphase, in der ein Requestor direkt mit dem Anbieter des Services über variable Eigenschaften verhandelt. Eine automatisierte Durchführung dieser Verhandlungsphase wird in der Literatur als *Negotiation* bezeichnet und ist vor allem im Umfeld autonomer Software-Agenten bereits ausführlich untersucht [42]. Bei diesen Verhandlungen können Eigenschaften für alle Service-Aufrufe eines Requestors vereinbart werden oder für einen einzelnen Aufruf. Die Verhandlung finden in der Regel zwischen den beiden Rollen direkt statt.

Beispielsweise vereinbart der Provider des Transport-Services mit einem Requestor, dass die Kosten für die Leistungserbringung ähnlich wie in  $S_5$  angeboten seien, allerdings mit geringeren Kosten von nur 300 €. Da es sich um einen wichtigen Kunden handelt, der seinerseits regelmäßige Service-Aufrufe garantiert, sei der Provider bereit, diesen Service günstiger anzubieten. Das zu entwickelnde Qualitätsmodell soll eine derartige, von der Veröffentlichung abweichende, Vereinbarung konzeptionell unterstützen.

### Ersetzung

Die Services desselben Service-Typs, die eine bestimmte Anfrage erfüllen, sind aus Sicht eines Requestors prinzipiell austauschbar. Da beispielsweise in der Anfrage  $R_3$  kein Optimierungsziel angegeben wurde, können alle Services des Ergebnisses der Anfrage benutzt werden. Fällt beispielsweise der gewählte Service  $S_3$  aus, so kann ebenfalls der Service  $S_4$  genutzt werden, der ebenfalls  $R_3$  erfüllt.

In einem Unternehmen kann ein Service-Aufruf oftmals ein Schritt in einem größeren, unternehmensübergreifenden Geschäftsprozess sein, wie bereits in Abschnitt 3.4.3 diskutiert. In unvorhergesehenen Ausnahmefällen, kann es notwendig werden, einen normalerweise genutzten Service (beispielsweise  $S_3$ ) durch einen anderen zu ersetzen, beispielsweise  $S_4$ , der eine geringere Lieferzeit hat. Diese dynamische Modifikation ist im Kontext von Workflow-Management-Systemen bereits eingehend untersucht worden [43, 65, 127, 166, 163, 165]. Es kann also ein ursprünglich gewählter Service aus der Ergebnismenge durch einen anderen ersetzt werden, der ebenfalls die Anfrage erfüllt. Dadurch kann der Geschäftsprozesse effizienter ausgeführt werden und die Flexibilität des Unternehmens wird erhöht.

Allgemein lassen sich Geschäftsprozesse analysieren und optimieren [60], indem einzelne Schritte durch bessere Services ausgeführt werden. Weiterhin können im Rahmen einer Geschäftsprozessanalyse beispielsweise Grenzwerte für die Ausführungsdauer eines Prozessschrittes erkannt werden. Diese können dann als Grenzwerte für die oben erwähnten Verhandlungen mit Providern genutzt werden.

Durch eine Analyse können beispielsweise auch zeitkritische Ausführungspfade in Geschäftsprozessen erkannt werden [60]. Hier kann durch den Austausch einzelner Services beispielsweise durch schnellere Services, kann der gesamte Geschäftsprozess unter Umständen ebenfalls schneller ausgeführt werden.

Ein Beispiel verdeutlicht den Vorteil eines Qualitätsmodells bei der Ersetzung eines Services: Normalerweise sei der Service  $S_1$  von einem Requestor am Ende eines Geschäftsprozess aufgerufen worden um eine produzierte Ware auszuliefern. Bei der Ausführung einer Geschäftsprozessinstanz kommt es nun zu Verzögerungen während der Produktion, so dass er den vereinbarten Termin nicht einhalten kann. Nun kann dynamisch der Service  $S_4$  für den Transport für diese Geschäftsprozessinstanz genutzt werden. Dadurch sei der Termin doch noch einzuhalten.

Die Besonderheit der Ersetzung ist dadurch gegeben, dass bereits ein Service verwendet wird und dessen Werte der Qualitätsmerkmale bekannt sind. Es ist nun das Ziel, einen zweiten Service mit besseren Werten in bestimmten Qualitätsmerkmalen zu finden. Dies wird dann möglich, wenn die (funktional äquivalenten) Services desselben Service-Typs bezüglich der Qualitätsmerkmale vergleichbar sind. Die Vergleichbarkeit der Werte ist also eine zentrale Anforderung an das Qualitätsmodell.

### **Komposition**

Die Vorteile der Service-Orientierung werden vor allem deutlich, wenn man zusammengesetzte Services betrachtet [84] (siehe auch Abschnitt 3.4.3). Dabei wird die Geschäftslogik in einer Prozessdefinition manifestiert, die von menschlichen Experten erarbeitet und optimiert wird. Hier wird bereits bei der Auswahl der einzelnen Services auf die Qualitätsmerkmale geachtet [186] und daraus die Qualität des zusammengesetzten Services berechnet.

Die Qualitätsmerkmale des neuen Service sind dabei abhängig von den einzelnen Services [30]. Dies wird in Kapitel 5 detaillierter betrachtet. Es soll hier die Anforderung berücksichtigt werden, dass die Qualitätsmerkmale beziehungsweise deren Werte unter Umständen aggregierbar sein sollen. Beispielsweise sollen die Kosten zweier Services so angegeben werden, dass die Werte addierbar sind und die Summe der Werte die Kosten der beiden Services darstellen.

### **Prozess (Um-)Planung**

Die Definition eines Prozesses wird üblicherweise von menschlichen Experten vorgenommen, wie bereits erwähnt wurde. In einer integrierten Prozessplanungs- und Prozessausführungsumgebung übernehmen beziehungsweise unterstützen Planungsalgorithmen aus dem Bereich der künstlichen Intelligenz diese Aufgabe. Ein Beispiel hierfür stellt das Pløngine-Projekt dar [135], in dem von einem Startzustand aus eine Prozessdefinition anhand der Vor- und Nachbedingungen der verfügbaren Prozessschritte zu einem definierten Prozessziel geplant wird [99]. Die

so für eine individuelle, einzelne Ausführung geplante Prozessdefinition wird dann von einer Prozesssteuerungskomponente ausgeführt [113].

Auch hier ist es neben einer Definition der Funktionalität notwendig, ein gemeinsames Modell der nicht-funktionalen Eigenschaften und ein Prozess Meta-Modell zu benutzen. Erst dann können Qualitätsmerkmale durch den Planungsalgorithmus berücksichtigt werden und eine Bewertung der optimierten Pläne ist möglich [136]. Weiterhin spielt gerade in einer dynamischen und unbeständigen serviceorientierten Architektur die Umplanung eine große Rolle. Dabei wird zur Laufzeit unter Benutzung der Planungsalgorithmen eine veränderte Prozessdefinition erzeugt, wenn unvorhergesehene Ereignisse auftreten, wie beispielsweise der Ausfall eines Services. Unter Umständen können dabei neue Prozessdefinitionen entstehen, die sich von den ursprünglichen Plänen auch darin unterscheiden, welche Services im weiteren Verlauf aufgerufen werden. Eine ausführliche Untersuchung findet sich beispielsweise in [137].

Aus der Sicht der Anwendungsszenarien sind Komposition und Ersetzung sehr ähnlich wie Prozess Planung und Umplanung. Allerdings werden in der Planung Algorithmen der künstlichen Intelligenz eingesetzt, um mit möglichst geringem menschlichen Einsatz eine Prozessdefinition zu entwerfen. Dazu sind mehr Informationen über die Services, insbesondere die Funktionalität und die Qualitätsmerkmale, nötig.

### **Service Management**

Service Management wird von den Providern der Services betrieben [118]. In der Regel bieten Provider eine ganze Reihe unterschiedlicher Services an. Einerseits können diese Services unterschiedliche Service-Typen implementieren, andererseits kann ein Provider auch mehrere Services desselben Typs mit unterschiedlicher Qualität anbieten. Somit unterscheiden sich die Services durch unterschiedliche Ausprägungen der Qualitätsmerkmale.

Wie in Kapitel 3 beschrieben, werden bei der Leistungserbringung Ressourcen eingesetzt. Während der Leistungserbringung kann ein Provider das Verhalten der Services beobachten und die Verwendung der Ressourcen anpassen. Das Verhalten kann auch durch Auswertungen der historischen Ausführungsdaten im nachhinein analysiert werden und eine Anpassung der eingesetzten Ressourcen vorgenommen werden [32]. Bezogen auf das Transport-Beispiel kann dieses beispielsweise durch neue Fahrer und Fahrzeuge geschehen. Ein Provider muss dazu flexibel auf Anfragen von Requestoren reagieren und den Markt beobachten. Dies ist besonders dann wichtig, wenn ein standardisierter Service-Typ implementiert wird, der auch von konkurrierenden Providern angeboten wird.

Das Qualitätsmodell liefert dem Provider eine konzeptionelle Grundlage, für einen Service-Typ mehrere Services mit unterschiedlichen Konditionen anzubieten. Der Service  $S_2$  wird beispielsweise mit 275 € wesentlich teurer angeboten als  $S_1$ . Dafür wird die maximale Temperatur, sowie eine Lieferzeit von 24 Stunden garantiert, wohingegen  $S_1$  nur eine Lieferzeit von 48 Stunden garantiert. Durch ein

aktives Service Management hat der Provider auch bei Verhandlungen mit einem Requestor mehr Eingriffsmöglichkeiten und einen besseren Überblick. So kann ein Provider bei sehr hoher Auslastung nur auf Verhandlungen eingehen, wenn ein sehr hoher Preis geboten wird.

## 4.3 Entwurf des Modells

Ausgehend von Begriffsdefinitionen und der Abgrenzung der Diskurswelt wird das Meta-Modell für Qualitätsmerkmale formuliert. Anschließend werden die Elemente des Meta-Modells in das Service-Modell integriert. Das Meta-Modell wird danach untersucht und ausgewählte Aspekte und ihre Auswirkungen auf die serviceorientierte Architektur herausgearbeitet.

### 4.3.1 Begriffsdefinition und Abgrenzung

Die Untersuchung der grundlegenden Konzepte der Service-Orientierung lieferte bereits in Abschnitt 2.4 zwei Arten von Service-Eigenschaften: funktionale Eigenschaften und Qualitätsmerkmale. Die funktionalen Eigenschaften wurden im Service-Modell in Kapitel 3 einem Service-Typ zugeordnet. Dasselbe Prinzip soll hier auf die Qualitätsmerkmale angewandt werden.

Ähnlich wie in den Abschnitten 2.2.1 und 3.2 werden hier zentrale Begriffe definiert, wie in einem konstruktivistischen Ansatz üblich. Ziel ist es, die Vielzahl der in der Literatur verwendeten Begriffe aus dem Kontext der Service-Eigenschaften zu definieren. Dieses ist der erste Schritt zur Konstruktion eines formalen Modells für die Eigenschaftsbeschreibung von Services.

#### Qualität

Der Begriff der Qualität stammt vom lateinischen *qualitas* (Eigenschaft, Beschaffenheit, Zustand), sowie *qualis* (wie konstituiert, von welcher Art oder Natur) ab. Qualität beschreibt die Beschaffenheit einer Sache. Die Bedeutung des Begriffs Qualität ist allerdings vom Kontext abhängig und variiert stark. So wird Qualität im ingenieurwissenschaftlichen Kontext als Maß verstanden, dass nur bezogen auf konkrete Problemstellungen in der industriellen Praxis zwischen Hersteller und Kunden definiert ist [18].

In der Wirtschaftslehre gibt es theoretische Ansätze für eine allgemeine Definition der Qualität, die auf Marketing-Sichtweisen beruhen [98]. Dort wird die Qualität eines Produktes als Maß für den Grad der Eignung eines Produktes für seinen jeweiligen Verwendungszweck verstanden [18]. Diese Definition ist ungenau, denn die Eignung hängt von Wahrnehmung und Empfindung der Qualität des Produktes ab. Allerdings zielt der Grad der Eignung ebenfalls auf die Benutzbarkeit eines Produktes ab. Die Eignung lässt sich hier aus Hersteller- und aus Kundensicht beschreiben. Betrachtet werden hierbei unter anderem die physische, reale Beschaffenheit des Produktes, sowie das Kundenverhalten. Man unterscheidet



auch hier das objektive und subjektive Qualitätsverständnis [53]. Objektive Qualität berücksichtigt die Beschaffenheit des Objektes und somit die Gesamtheit der Produkteigenschaften, wohingegen die subjektive Qualität eines Produktes durch die Gesamtheit aller *vom Kunden wahrgenommenen* Eigenschaften abhängt. Subjektive Qualität ist bestimmt durch den subjektiv wahrgenommenen Nutzen des Produktes für die Befriedigung der individuellen Bedürfnisse.

Eine Zusammenfassung dieser Sichtweisen auf die objektive und subjektive Qualität versucht die so genannte teleologische Auffassung der Qualität<sup>1</sup>. Das teleologische Qualitätsverständnis beruht auf einer Subjekt-Objekt-Beziehung und drückt ein Qualitätsurteil aus. Dieses Urteil gibt an, in wie weit das Produkt mit seinen verschiedenen Eigenschaften die aus den Bedürfnissen des Kunden resultierenden Anforderungen erfüllt. Die teleologische Qualität stellt somit die Eignung des Produktes für einen bestimmten Zweck dar. [75]

Festzuhalten sei an dieser Stelle, dass sich Qualität auf ein konkretes Objekt bezieht und die Ausprägung einer Qualität abhängig ist vom Subjekt, also dem Zweck. Weiterhin ist Qualität vor allem zum Vergleich verschiedener Objekte sinnvoll. Allerdings sind umfangreiche Kenntnisse in der Sichtweise und dem Kontext notwendig, um die Aussage über einer bestimmten Qualität einordnen zu können. Wie bereits erwähnt, wird Qualität oftmals zur Kommunikation benutzt, um die Eigenschaften eines Objektes zu beschreiben. Bezogen auf die Service-Orientierung schließt dies die Kommunikation zwischen Provider und Requestor über eine gewünschte oder verfügbare Qualität ein. Es ist also genau zu spezifizieren, welche Aspekte bei der Definition einer Qualität eines Services betrachtet werden und was die Ausprägung bedeuten. Weiterhin bleibt festzuhalten, dass Qualität aus mehreren Eigenschaften besteht. Somit setzt sich eine bestimmte Qualität aus Bewertungen oder Kennzahlen für einzelne Qualitätsmerkmale zusammen. Diese Beschreibung ist der Qualität ähnlich, die im EN/ISO-Standard EN ISO 9000:2000 definiert ist [68]: Qualität ist dort als Grad definiert, in dem ein Satz inhärenter Merkmale Anforderungen erfüllt.

Die *Qualität* eines Objektes beschreibt also die Gesamtheit der Ausprägungen der für einen definierten Zweck relevanten Qualitätsmerkmale dieses Objektes. Ein *Qualitätsmerkmal* ist eine einzelne, relevante Eigenschaft, für die bei einem konkreten Objekt eine Ausprägung angegeben werden kann. Zwei Objekte sind in der Regel bezüglich ihrer Ausprägungen eines Qualitätsmerkmals vergleichbar.

Für den Zweck des Qualitätsmodells für Services sind die Objekte Services. Die Qualität der Services in einer serviceorientierten Umgebung wird als *Service-Qualität* bezeichnet. In der englischen Literatur ist der Begriff *Quality of Services (QoS)* gebräuchlich. Zwei Services sind bezüglich ihrer Qualität vergleichbar, indem die Ausprägungen der einzelnen Qualitätsmerkmale verglichen werden.

---

<sup>1</sup>Teleologisch: Stammt vom griechischen *telos*, das „Ziel“ bedeutet; somit „zielgerichtete Qualität“ oder „zweckgebundene Qualität“

### Nicht-funktionale Eigenschaften

In der Literatur zur Service-Orientierung und Web Services ist der Begriff der *nicht-funktionalen Eigenschaften* stark verbreitet [16, 100, 112, 161] und wird auch als Synonym für Servicequalität beziehungsweise Quality of Services [111] angesehen. Dieser Begriff erlaubt eine deutliche Abgrenzung zur Funktionalität. Die funktionalen Eigenschaften beschreiben dann, *was* ein Service macht und die nicht-funktionalen beschreiben, *wie* er es macht. Diese Aufteilung erscheint intuitiv und nützlich, da es möglich ist, *funktionale Äquivalenz* zwischen zwei Services zu definieren, wenn beide dieselben funktionalen Eigenschaften aufweisen. Man kann dann die Aussage treffen, dass ein Service durch einen funktional äquivalenten Service ersetzt werden kann. Dieses lohnt sich dann, wenn der neue Service bessere nicht-funktionale Eigenschaften hat, wie beispielsweise einen niedrigeren Preis, schnellere Leistungserbringung oder höhere Sicherheit.

In einer serviceorientierten Architektur ist es allerdings schwer zu entscheiden, wann eine bestimmte Eigenschaft funktional ist und wann sie nicht-funktional ist. Als Beispiel sei die Ausführungsdauer eines Services gewählt, die in der Literatur als typische nicht-funktionale Eigenschaft angesehen wird. Folgende (informale) Anfrage an einen Service liegt nun vor:

$R_3$  := „Ein Transport Service, in 24 Stunden und für unter 200 € transportiert.“

Die Funktionalität des gesuchten Services ist die Transportdienstleistung, wie im Beispiel in Abschnitt 4.1.2 beschrieben. Im Service-Modell wurde die Funktionalität einem Service-Typ zugeordnet. In der Anfrage ist aber auch angegeben, dass der Transport innerhalb von 24 Stunden und für unter 200 € zu erfolgen hat. Dieses sind Anforderungen, die Wertebereichseinschränkungen darstellen. Ein Service, der nicht innerhalb von 24 Stunden transportieren kann, ist somit nicht geeignet, einen anderen Service zu ersetzen. Das Ergebnis ist also funktional korrekt, aber dennoch nicht akzeptabel, da ein weiteres Kriterium verletzt wurde [106]. Ähnliche Beispiele können für andere Qualitätsmerkmale angegeben werden. Wenn in der Anfrage eines Requestors für ein Qualitätsmerkmal Wertebereichseinschränkungen vorliegen, so kann die Bezeichnung nicht-funktional hier zu Missverständnissen führen und ist nicht angemessen.

#### 4.3.2 Elemente des Qualitätsmodells

Eine grundsätzliche Idee des Qualitätsmodells ist die Umsetzung der beiden Abstraktionsebenen aus dem Service-Modell (siehe auch Abschnitt 3.3). Diese Übertragung der Abstraktionsebenen auf das Qualitätsmodell soll hier zunächst kurz vorgestellt werden, indem das Vorgehen zur Umsetzung geschildert wird:

1. Für einen Service-Typ sind jeweils die relevanten Qualitätsmerkmale zu spezifizieren. Dies wird ähnlich wie im Kontext des Service-Modells beschrie-

ben als Standardisierung angesehen und gehört ebenso zum Service-Typ, wie die Spezifikation der Funktionalität und der Schnittstelle.

2. Für einen Service ist anzugeben, welcher Service-Typ implementiert wird und welche Ausprägungen für die spezifizierten Qualitätsmerkmale realisiert werden.

Daraus resultieren die folgenden Vorteile:

- Wird eine neue Service-Beschreibung inklusive Ausprägungen für die Qualitätsmerkmale veröffentlicht, lässt sich eindeutig feststellen, ob die angegebenen Qualitätsmerkmale für den entsprechenden Service-Typ zulässig sind und ob alle (notwendigen) spezifizierten Qualitätsmerkmale berücksichtigt wurden.
- Ist der zulässige Wertebereich definiert, kann überprüft werden, ob die angegebenen Ausprägungen gültige Werte darstellen.
- Die Vergleichbarkeit mit bereits veröffentlichten Services desselben Service-Typs ist gewährleistet, da dieselben Qualitätsmerkmale von allen Service desselben Service-Typs verwendet werden.

**Qualitätsmerkmal** Die Beispiele des Video-Services und des Transport-Services haben aufgezeigt, welche Aspekte zur Spezifikation der Qualitätsmerkmale dabei zu berücksichtigen sind. Um nun diese Komplexität und die Vielfalt der Qualitätsmerkmale handhaben zu können, werden unterschiedliche Aspekte zur Charakterisierung formalisiert. Um die Ausprägungen der Qualitätsmerkmale miteinander vergleichen zu können und zur Ermittlung der Rangfolge macht es zunächst Sinn, die Werte zu betrachten.

**Definition 18 Domäne:** Eine Domäne  $dom$  ist eine Menge von Werten und beschreibt einen festgelegten Wertebereich.

Die Angabe der Menge erfolgt entweder durch die direkte Angabe der Werte, durch Angabe einer standardisierten, vordefinierten Menge oder durch Angabe eines Ausdrucks, dessen Auswertung die Menge ergibt. Die Menge kann also direkt angegeben werden, indem die Werte aufgezählt werden, beispielsweise in der Form  $dom = \{\text{„Web-Portal“}, \text{„Email“}, \text{„Telefon“}\}$  für die Statusüberwachung. Wie in Abschnitt 4.1.2 diskutiert, kann ein zusätzlicher Wert aufgenommen werden, mit dem ausgedrückt wird, dass die Ausprägung nicht bekannt ist, zum Beispiel „unbekannt“, „NULL“ [160] oder „nicht definiert“. Alternativ können die Werte bestimmt werden, indem auf eine andere, spezifizierte Menge verwiesen wird, wie beispielsweise in der Mathematik üblich durch Symbole wie die Menge der ganzen Zahlen  $\mathbb{Z}$ ; ebenso  $\mathbb{N}$ ,  $\mathbb{R}$ ,  $\mathbb{Q}$  [24]. Außerdem ist es möglich sein, die Domäne mit Hilfe eines Ausdrucks zu spezifizieren, zum Beispiel für die Temperatur  $dom = \{v \in \mathbb{Z} \mid -40 < v \leq 100\}$ .

Als weiterer zentraler Aspekt wurde bereits die Semantik der Werte herausgearbeitet. Die Bedeutung der Werte kann entweder für jeden einzelnen Wert oder durch Maßeinheit und Messverfahren spezifiziert werden. Durch Maßeinheiten wie Grad Celsius ist die Bedeutung für ein Qualitätsmerkmal festgelegt, aber nicht für jedes Qualitätsmerkmal kann eine Maßeinheit angegeben werden: beispielsweise für die Statusüberwachung. Es wird für die Zuordnung von Werten zu Bedeutungen der Begriff Metrik verwendet:

**Definition 19 Metrik:** Eine Metrik *metric* beschreibt die Abbildung von Dingen oder Beobachtungen der realen Welt auf die Werte einer Domäne. Es wird somit die Semantik der Werte definiert.

Eine Metrik gibt also an, wann, wo und mit welchem Verfahren der Wert zu messen ist beziehungsweise wie der Wert feststellbar oder beobachtbar ist. Wenn zutreffend, kann also das Messverfahren spezifiziert und die Maßeinheit festgelegt werden. Die Beschreibung kann in einer formalen Sprache spezifiziert oder informell beschrieben werden. Hier wird eine ähnliche Technik benötigt, wie für die Funktionalität im Kontext des Service-Modells beschrieben. Zusätzlich definiert die Metrik die Maßeinheit, wie beispielsweise €, °C oder *kg* (Kilogramm) und das Verfahren zur Messung der Werte. Im Rahmen der Beispiele in Abschnitt 4.1 wurden bereits ausführlich Aspekte der Bedeutung der Werte eines Qualitätsmerkmals diskutiert und anhand von Beispielen illustriert. Die Bedeutung der Metrik wird in Abschnitt 4.3.4 genauer beschrieben. Neben Domäne und Metrik ist noch folgender Aspekt zur Charakterisierung eines Qualitätsmerkmals notwendig:

**Definition 20 Skalenniveau:** Das Skalenniveau ist eine Klassifikation eines Qualitätsmerkmals hinsichtlich mathematischer Eigenschaften der Werte. Die Menge der möglichen Skalenniveaus *Scale* ist definiert als

$$Scale = \{nominal, ordinal, interval, ratio\}$$

◇

In Abhängigkeit vom Messverfahren und der Art der Qualitätsmerkmale sind vier Stufen des Skalenniveaus unterscheidbar. Je nach Skalenniveau sind unterschiedliche mathematische Operationen mit den Werten sinnvoll. Nominal skalierte Werte sind lediglich Namen, wie zum Beispiel die *Statusüberwachung* oder der *Codec*. Rational skalierte Werte sind beispielsweise die *Kosten*. Aus der Skala ergibt sich, welche arithmetischen Operationen, insbesondere welche Vergleichsoperationen, zugelassen sind. Die Skalen und ihre Auswirkungen werden in Abschnitt 4.3.5 ausführlich betrachtet. Es ergibt sich somit für ein Qualitätsmerkmal die folgende Definition:

**Definition 21 Qualitätsmerkmal:** Ein Qualitätsmerkmal *q* ist ein Tripel  $(dom_q, metric_q, scale_q)$  mit

- der Domäne des Qualitätsmerkmals  $q : dom_q$ ,
- der Metrik des Qualitätsmerkmals  $q : metric_q$  und
- dem Skalenniveau des Qualitätsmerkmals  $q : scale_q$ , mit  $scale_q \in Scale$

◇

**Service-Typ und Service Qualität** Es ist nun möglich, verschiedene Qualitätsmerkmale zu modellieren, indem Domäne, Metrik und Skalenniveau spezifiziert werden. Um das Ziel zu erreichen, Services untereinander vergleichen zu können, sind jeweils Ausprägungen für dieselben Qualitätsmerkmale anzugeben. Daher wird einem Service-Typ  $T$  eine Menge von Qualitätsmerkmalen zugeordnet (siehe auch Definition 14).

**Definition 22 Service-Typ Qualität:** Sei  $Q_T$  die Menge der Qualitätsmerkmale eines Service-Typs  $T$ , oder kurz: die Service-Typ Qualität. Seien  $q_i, 1 \leq i \leq n$  Qualitätsmerkmale von  $T$ , dann ist

$$Q_T = \{q_1, \dots, q_n\}.$$

◇

Zur Illustration soll hier ein Teil der Qualitätsmerkmale des Transport-Beispiels aus Abschnitt 4.1.2 angegeben werden. Betrachtet seien die zwei Qualitätsmerkmale Kosten und Temperatur, die wie folgt definiert werden:

- Das Qualitätsmerkmal Kosten ist als positive, rationale Zahl einer Ratioskala definiert:  $q_{cost} = (\mathbb{Q}^+, EuroPreis, ratio)$ . Die Metrik ist dargestellt durch den Namen „EuroPreis“ und spezifiziert als Währungswert in €, inklusive Umsatzsteuer.
- Das Qualitätsmerkmal Temperatur ist das garantierte maximale Temperatur und wird in ganzen Grad Celsius, also einer Intervallskala, angegeben:  $q_{temp} = (\mathbb{Z}, MaxGradCelsius, interval)$ . Die Metrik ist dargestellt durch den Namen „MaxGradCelsius“ und sei wie beschrieben präzise und vollständig spezifiziert.
- Für die Menge der bisher betrachteten Qualitätsmerkmale des Service-Typen  $T_{shipping}$  gilt folglich:  $Q_{T_{shipping}} \supset \{q_{cost}, q_{temp}\}$ .

Somit sind Qualitätsmerkmale für einen Service-Typ definierbar. Wie auch im Service-Modell beschrieben, wird ein Service-Typ und die Beschreibung seiner Qualitätsmerkmale als standardisiert angesehen. Dies ist die Grundlage für einen Provider, der einen Service dieses Service-Typs anbieten möchte. Im Folgenden wird nun betrachtet, wie ein Service angeboten wird, also wie die Ausprägungen für die standardisierten Qualitätsmerkmale des Service-Typs definiert sind.

**Definition 23 Service Qualität:** Sei  $Q_T$  die Service-Typ Qualität und  $S$  ein Service, der den Service-Typen  $T$  implementiert. Sei weiterhin eine Gesamtdomäne definiert als  $dom_{Q_T} = \bigcup_{q \in Q_T} dom_q$ . Dann ist die Qualität des Services  $S$  eine Abbildung  $\lambda_S : Q_T \rightarrow dom_{Q_T}$ , so dass

$$(\forall q \in Q_T) : \lambda_S(q) \in dom_q$$

◇

Das bedeutet, es existiert eine Abbildung  $\lambda_S$ , die jedem Qualitätsmerkmal aus  $Q_T$  einen Wert zuweist, wobei jeder Wert jeweils aus der Menge der definierten Domäne des Merkmals stammt<sup>2</sup>. Das griechische  $\lambda$  steht für *Level*, also das Niveau der Qualität beziehungsweise die Ausprägung der Merkmale eines Services an. In der englischsprachigen Literatur hat sich der Begriff „Quality of Service Level“ etabliert [92] (siehe dazu auch Kapitel 6 und Definition 24). Die Qualität eines Services ist also die Menge der Ausprägungen aller Qualitätsmerkmale des Services. Für den Service  $S_2$  aus dem Transport Beispiel ergibt sich nun die folgende Ausprägung der Qualitätsmerkmale für die beiden Eigenschaften Kosten und Temperatur (siehe Tabelle 4.2):

$$\begin{aligned} \lambda_{S_2}(q_{cost}) &= 275 \\ \lambda_{S_2}(q_{temp}) &= -8 \end{aligned}$$

Somit kann ein Provider für jeden Service die Qualität durch die jeweiligen Ausprägungen der Qualitätsmerkmale angeben. Diese Informationen können als Teil der Service-Beschreibung angegeben werden, die beim Broker veröffentlicht wird. Durch eine allgemein akzeptierte oder standardisierte Spezifikation eines Service-Typs kann ein Requestor Anfragen stellen, die auf die spezifizierten Qualitätsmerkmale eingehen und seine Anforderungen (Wertebereichseinschränkungen und Optimierungsziel) in die Anfrage einbauen. So werden beispielsweise Anfragen möglich, die Einschränkungen der Qualitätsmerkmale angeben, wie Kosten maximal 400 € und Temperatur unter +25°C:

$$R_4 = \{S \mid \lambda_S(q_{cost}) \leq 400 \wedge \lambda_S(q_{temp}) < 25\}$$

In Bezug auf die in Tabelle 4.2 angebotenen Services, kommen hier die Services  $S_2$ ,  $S_4$  und  $S_5$  in Frage, also  $R_4 = \{S_2, S_4, S_5\}$ . Da für die Services  $S_1$  und  $S_3$  keine Angabe der Temperatur erfolgt, können sie ebenso wenig berücksichtigt werden wie  $S_6$ , der mit 450 € zu teuer ist.

<sup>2</sup>Hier ist zu beachten, dass die Domäne um Nullwerte erweitert werden kann, siehe Erläuterungen zu Definition 18.

**Service Level Agreement** Die Menge der spezifizierten Qualitätsmerkmale eines Services liefert eine Grundlage für Vereinbarungen zwischen Provider und Requestor. Diese können sich auf Ausprägungen für Qualitätsmerkmale einigen oder diese verhandeln. Hierbei müssen die veröffentlichten Werte nicht zwingend eingehalten werden. Die individuell vereinbarten Werte werden vertraglich dokumentiert und gelten für einen oder mehrere Service-Aufrufe. In der Regel wird auch festgehalten, was geschieht, wenn die vereinbarte Qualität nicht eingehalten werden kann, beispielsweise indem Konventionalstrafen oder Ausgleichsleistungen vereinbart werden. In der folgenden Definition werden allerdings nur die vereinbarten Werte betrachtet:

**Definition 24 Service Level Agreement:** Ein Service Level Agreement (SLA) ist eine vertragliche Grundlage zwischen Requestor und Provider für die Ausführung eines Services  $S$ , die Ausprägungen für die relevanten Qualitätsmerkmale  $q \in Q^{SLA}$  definiert. Implementiere  $S$  einen Service-Typ  $T$  und sei  $Q^{SLA} \subseteq Q_T$ , dann ist ein  $\lambda^{SLA}$  eine Abbildung  $\lambda^{SLA} : Q^{SLA} \rightarrow dom(Q_T)$ , so dass gilt:

$$(\forall q \in Q^{SLA}) : \lambda^{SLA}(q) \in dom_q$$

◇

Die vereinbarten Ausprägungen der Qualitätsmerkmale werden vom Provider für einen Requestor für dessen Service-Aufrufe garantiert. Die Definition bedeutet, dass ein Service Level Agreement [73] zunächst auf einen Teil der standardisierten Qualitätsmerkmale bezieht. Hierbei handelt es sich um die Qualitätsmerkmale, die von Requestor und Provider als relevant angesehen werden. Die Ausprägungen der Merkmale werden dann durch die Abbildung  $\lambda^{SLA}$  festgelegt, wobei die Werte aus der jeweiligen Domäne des Merkmals stammen. Die Werte eines SLA sind also unabhängig von den Werten des ursprünglich angebotenen Services.

**Zusammenfassung** Nun sind die Elemente des Qualitätsmodells definiert: ausgehend von einem Qualitätsmerkmal, dass Metrik, Domäne und Skalenniveau spezifiziert, kann die Service-Typ Qualität definiert werden. Dann können für jeden Service die Ausprägungen der Qualitätsmerkmale angegeben werden und zusammen mit den anderen Bestandteilen einer Service-Beschreibung veröffentlicht werden. Zusätzlich können sich Provider und Requestor in einem Service Level Agreement auf abweichende Ausprägungen der Qualitätsmerkmale einigen. Das Qualitätsmodell erlaubt die vom Provider angegebene Ausprägungen ebenso auf Zulässigkeit zu überprüfen, wie die Anfragen eines Requestors. Weiterhin sind Anfragen auf alle Services eines Service-Typs anwendbar und Services desselben Typs vergleichbar. Somit lassen sich die Anwendungsfälle realisieren und die Anforderungen sind erfüllt.

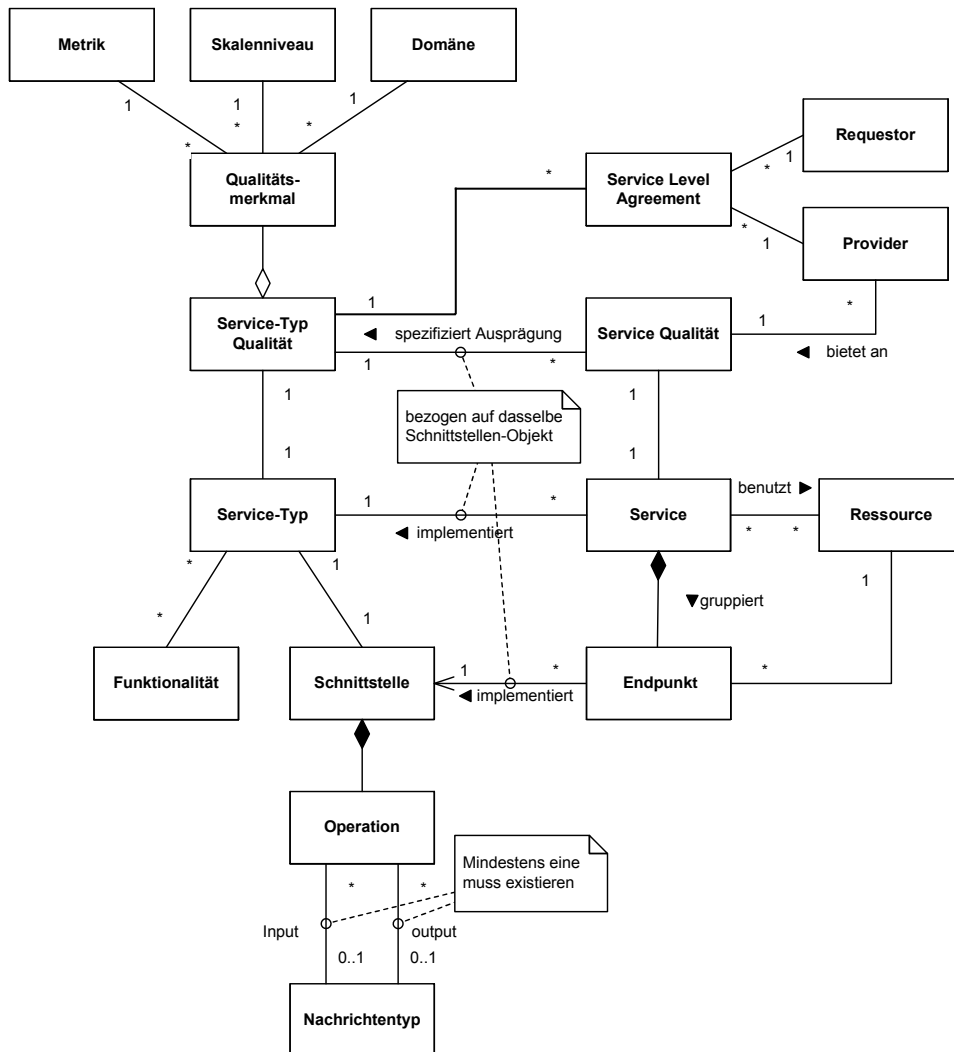


Abbildung 4.2: Integration des Qualitätsmodells und des Service-Modells



### 4.3.3 Integration des Qualitätsmodells in das Service-Modell

Das Qualitätsmodell ist in Abbildung 4.2 als UML Klassendiagramm dargestellt und zeigt die Bestandteile des Qualitätsmodells ebenso wie die des Service-Modells aus Abschnitt 3.2. Zusätzlich werden jeweils die Elemente der Modelle in Beziehung gesetzt. Im Folgenden sind die Beziehungen der Elemente kurz beschrieben und ein Bezug zur jeweiligen Definition wird angegeben:

- Ein Qualitätsmerkmal (Definition 21) setzt sich zusammen aus *Domäne* (Definition 18), *Metrik* (Definition 19) und *Skalenniveau* (Definition 20).
- Die für einen Service-Typen spezifizierten Qualitätsmerkmale werden in der *Service-Typ Qualität* zusammengefasst (Definition 22). Diese ist bijektiv (Kardinalität 1:1) mit dem Service-Typ verbunden, das heißt, es gibt für jeden Service-Typ genau eine Spezifikation seiner Qualität, also genau eine Menge von Qualitätsmerkmalen.
- Entsprechend der Beziehung zwischen Service-Typ und Service, wird die spezifizierte Service-Typ Qualität realisiert durch eine *Service Qualität* (Definition 23). Dies ist dargestellt durch die *spezifiziert Ausprägung*-Beziehung. Die Service Qualität ist bijektiv mit Service verbunden, das heißt für jeden Service ist genau eine Service Qualität spezifiziert. Da jeder Service-Typ durch mehrere Services implementiert werden kann, kann es folglich auch mehrere Elemente der Service Qualität für jede Service-Typ Qualität geben.
- Ein *Provider* bietet einen Service zu einer Service-Qualität an (dargestellt durch die *bietet an*-Beziehung). Diese Service Qualität gehört offensichtlich zu einem Service, den der Provider ebenfalls anbietet (im Service-Modell nicht explizit abgebildet).
- Ein *Provider* und ein *Requestor* einigen sich auf eine konkrete Ausprägung der Service-Qualität in einem *Service Level Agreement*. Dieses Agreement ist nur gültig zwischen einem konkreten Provider und einem konkreten Requestor. Zusätzlich sind noch weitere vertragliche relevante Aspekte in diesem Service Level Agreement angegeben.
- Durch Verhandlungen oder besondere Absprachen zwischen Provider und Requestor können die Werte hier von der angebotenen Service-Qualität abweichen. Daher ist eine Beziehung zwischen Service Level Agreement und Service-Typ Qualität dargestellt.

In den folgenden Abschnitten werden nun die Metrik und die Skala als zentrale Aspekte der Qualitätsmerkmale und des Qualitätsmodells betrachtet, ihre Relevanz herausgestellt und ihre Umsetzung an Beispielen verdeutlicht.

#### 4.3.4 Metrik

In der allgemeinen Wortbedeutung ist eine Metrik ein System von Kennzahlen oder ein Verfahren zur Messung einer quantifizierbaren Größe. Eine Metrik definiert dabei, wie Beobachtungen oder Messungen in der realen Welt übertragen werden auf einen Wert, der dann dokumentiert werden kann. Der Begriff *Metrik* ist aus vielen Bereichen bekannt. Eine Metrik ist beispielsweise in der Mathematik eine Funktion, die je zwei Punkten eines Raums einen reellen Wert zuordnet, der als Abstand angesehen wird. Übertragen auf die Qualitätsmerkmale des Qualitätsmodells sind die Punkte im Raum die Ausprägungen der Qualitätsmerkmale. Dann ist durch eine Metrik anzugeben, wie der Abstand zwischen den Ausprägungen zu berechnen ist. Für das Beispiel des Qualitätsmerkmals Kosten ( $q_{cost}$ ) ist der Abstand der Betrag der Differenz der Werte, also für die beiden Services  $S_2$  und  $S_3$ :

$$d(\lambda_{S_2}(q_{cost}), \lambda_{S_3}(q_{cost})) = |\lambda_{S_2}(q_{cost}) - \lambda_{S_3}(q_{cost})| = 125$$

Eine Metrik im Sinne der Mathematik, also vor allem die Berechnung des Abstandes, fokussiert vor allem diesen einen Aspekt. Für das Qualitätsmodell ist dies nicht ausreichend und nicht immer anwendbar, wie die folgenden Beispiele zeigen, die Bezug nehmen zu den in Abschnitt 4.1 eingeführten Beispielszenarien.

**Metrik des Qualitätsmerkmals Kosten** Neben der bereits angegebenen Abstandsberechnung wird für das Qualitätsmerkmal Kosten ( $q_{cost}$ ) auch festgelegt, dass der angegebene Wert in der Währung € angegeben ist. Zusätzlich wird spezifiziert, welche Wechselkurse zu anderen Währungen anzuwenden sind, zum Beispiel offizieller Wechselkurs der Europäischen Zentralbank vom Vortag. Dadurch ist es einem Requestor aus einem anderen Währungsgebiet möglich, die Kosten des Services für ihn korrekt zu interpretieren. Außerdem ist zu definieren, wie viele Nachkommastellen berücksichtigt werden, beispielsweise werden die Kosten auf Basis von vier Nachkommastellen genau berechnet und dann aufgerundet auf zwei Nachkommastellen angegeben. Weitere Aspekte und potentielle Missverständnisse, die verhindert werden müssen, sind bereits in Abschnitt 4.1 angegeben worden.

**Metrik des Qualitätsmerkmals Temperatur** Im Beispiel der Temperatur ist die Einheit °C und wird zunächst als normaler Zahlenwert mit dieser Maßeinheit angesehen. Dahinter steht jedoch die Definition der Celsius-Temperaturskala. Diese verwendet zwei Fixpunkte: die Temperaturen von Gefrier- und Siedepunkt des Wassers. Diese Fixpunkte gelten bei einem Luftdruck von 1013,25 hPa<sup>3</sup>. Zwischen diesen Fixpunkten findet eine lineare Unterteilung in 100 Einheiten statt. Im Transport-Beispiel ist es sinnvoll, zusätzlich zu definieren, wo und mit welchem Verfahren diese Temperatur theoretisch gemessen werden kann. Die Abstandsberechnung ist der Betrag der Differenz der Werte.

<sup>3</sup>Hektopascal; der mittlere Luftdruck der Atmosphäre auf Meereshöhe wird als Normdruck definiert.

**Metrik des Qualitätsmerkmals Codec** Im Video-Beispiel in Abschnitt 4.1.1 ist der Codec als standardisierter Name angegeben und drei Werte als zulässige Werte definiert worden (siehe Tabelle 4.1, Seite 65). Hier ist als Bestandteil der Metrik keine Reihenfolge angegeben und keine Abstandsberechnung anwendbar. Die Metrik definiert hier, was sich hinter den einzelnen Werten verbirgt, also beispielsweise die Spezifikation wie ein Video im *MPEG*-Codec auszusehen hat, oder ein Verweis auf den Standard *ISO/IEC JTC1/SC29/WG11*.

**Zusammenfassung** Es wurde aufgezeigt, dass die Spezifikation einer Metrik stark vom betrachteten Qualitätsmerkmal und dem Anwendungsgebiet abhängt. Daher kann eine allgemeine Vorgehensweise, wie eine Metrik zu definieren ist, nicht angegeben werden, sondern lediglich aufgezeigt werden, was bei der Spezifikation einer Metrik zu beachten ist und welche Aspekte dabei berücksichtigt werden müssen. Zur Definition einer Metrik können Verfahren der Software-Entwicklung verwendet werden, beispielsweise durch Formulierung von Fragenkatalogen in der Analyse-Phase. In Abschnitt 4.1 sind bereits einige beispielhafte Fragestellungen vorgestellt worden.

Die Metrik ist jeweils für ein Qualitätsmerkmal eines Service-Typs definiert. Somit wird im weiteren Verlauf davon ausgegangen, dass die Metrik standardisiert vorgegeben wird und alle Beteiligten sich auf diese Definition geeinigt haben. Sowohl Provider als auch Requestor können somit die Bedeutung der Werte verstehen. Die folgenden zentralen Aspekte die bei der Spezifikation einer Metrik bedacht werden müssen seien hier festgehalten:

- **Semantik** des Qualitätsmerkmals
- **Maßeinheit** beziehungsweise Semantik der Werte
- **Messverfahren** oder Heuristik zur Ermittlung und Überprüfung der Werte
- **Art des Wertes:**
  - fester Wert
  - Extremwert (maximaler oder minimaler Wert)
  - Erwartungswert beziehungsweise Mittelwert

### 4.3.5 Skalenniveau

Die Definition von Skalenniveaus in der Statistik geht zurück auf Arbeiten von S. Stevens [144], der die vier hier vorgestellten Skalenniveaus und ihre mathematischen Eigenschaften bereits 1946 definierte<sup>4</sup>:

---

<sup>4</sup>In der Statistik sind weitere Skalenniveaus definiert [44]. Hier sollen lediglich die wichtigsten vier betrachtet werden; die anderen vorgeschlagenen Skalenniveaus spielen vor allem im Kontext von statistischen Untersuchungen eine Rolle.

1. Nominalskala
2. Ordinalskala
3. Intervallskala
4. Ratioskala (Synonym: Verhältnisskala)

Die Intervallskala und die Ratioskala werden zusammenfassend auch als Kardinalskala bezeichnet. Merkmale auf einer Kardinalskala werden als *metrisch* und Merkmale der anderen Skalen als *kategorial* bezeichnet. Im Allgemeinen gibt ein Skalenniveau die Art an, wie Variablen oder Nummern definiert oder kategorisiert werden. Das Skalenniveau legt die folgenden Charakteristika fest:

- Das Skalenniveau bestimmt die mathematischen Operationen, die mit einem entsprechend skalierten Qualitätsmerkmal zulässig sind. Dabei können Operationen, die bei Qualitätsmerkmalen eines bestimmten Skalenniveaus zulässig sind, grundsätzlich auch auf Merkmale höherer Skalenniveaus durchgeführt werden.
- Es wird festgelegt, welche Transformationen mit entsprechend skalierten Merkmalen durchgeführt werden können, ohne Information zu verlieren beziehungsweise zu verändern.
- Es wird eingeschränkt, welche Information das entsprechende Merkmal liefert, das heißt, welche Interpretationen zulässig sind für die Ausprägungen des entsprechenden Merkmals.

Die Skalen sind geordnet und die Kategorisierung beginnt mit dem niedrigsten Niveau. Für jedes Skalenniveau wird nach einer kurzen Beschreibung ein Beispiel angegeben und anschließend die möglichen arithmetischen Operationen beschrieben. Diese Operationen sind relevant für den Vergleich zweier Services beziehungsweise deren Ausprägungen und die Aggregation von Ausprägungen (siehe Kapitel 5).

### **Nominalskala**

Die Nominalskala weist den beobachteten Verhalten beziehungsweise den gemessenen Werten Namen zu und kategorisiert so die Werte. Es werden keine Beziehungen oder Abstände (im Sinne einer mathematischen Metrik, siehe Abschnitt 4.3.4) zwischen den Namen definiert. Daher bedeutet dieselbe Ausprägung eines Qualitätsmerkmals zweier Services lediglich, dass die Services bezüglich dieses Merkmals die gleiche Qualität haben, also äquivalent sind. Die Namen haben aber keine numerische Bedeutung, auch wenn es sich um Ziffern handeln sollte.

**Beispiele** Bezogen auf das Transport-Beispiel ist das Qualitätsmerkmal Statusüberwachung als nominal definiert. Es gibt die drei vordefinierten Werte *Web-Portal*, *E-Mail* und *Telefon*, wobei keine Ordnung angegeben werden kann. Wie bereits im Zusammenhang mit der Metrik beschrieben, wird auch das Qualitätsmerkmal *Codec* des Video-Beispiels als nominal definiert. Es sei hier auch darauf hingewiesen werden, dass eine Metrik für ein nominales Qualitätsmerkmal eine Abbildung auf Zahlenwerte vorgeben kann. Sei beispielsweise die Statusüberwachung über ein Web-Portal als 1 codiert und mittels E-Mail als 2, so bedeutet dieses durch die Definition als nominal skaliert dass E-Mail nicht von höherer Qualität ist als Web-Portal. Allerdings kann ein Requestor bei einer Anfrage spezifizieren, dass er eine Ausprägung einer anderen vorzieht.

Beispielsweise suche ein Requestor einen Service vom Typ  $T_{shipping}$ , dessen Kosten unter 200 € liegen (erfüllt von  $S_1$  und  $S_3$ ). Gibt er nun noch an, dass die Ausprägung *E-Mail* für ihn besser sei als die Ausprägung *Web-Portal*, so ist  $S_3$  der beste von den verfügbaren Services, die seine Anforderung erfüllen.

**Operationen** Für Qualitätsmerkmale der Nominalskala sind als einzige arithmetische Operationen der direkte Vergleich (Vergleichsoperatoren  $=$ ,  $\neq$ ) zulässig, sowie Aufzählungen, wie oft eine bestimmte Ausprägung auftaucht. Ein Mittelwert kann nicht berechnet werden, außer der Angabe des am häufigsten auftretenden Wertes (Modus).

Die Werte können nicht addiert werden oder Abstände berechnet werden. Wie bereits im Kontext der Definition 18 erläutert, muss die Wertemenge im Vorfeld eindeutig und vollständig spezifiziert werden, in dem alle möglichen Namen aufgezählt werden. Sind nicht alle beobachtbaren Werte der realen Welt im Voraus bekannt oder die Aufzählung zu umfangreich, so können Namen wie *unbekannt* oder *sonstige* spezifiziert werden (Für den Transport-Service sei „-“ definiert als keine Verschlüsselung, siehe Tabelle 4.2). Dadurch ist auch die Abbildung von Nullwerten möglich.

### Ordinalskala

Für Qualitätsmerkmale der Ordinalskala ist eine Ordnung definiert. Dieses Skalenniveau erlaubt zwar eine graduelle Abstandsbestimmung, aber keinen Betragswert des Abstandes. So hat ein Service mit einem höheren Wert eines Qualitätsmerkmals *mehr* von dieser Qualität beziehungsweise eine höhere Qualität (bei einer aufsteigenden Metrik), aber der konkrete Abstand kann nicht angegeben werden beziehungsweise ist nicht deterministisch.

**Beispiele** Das Qualitätsmerkmal Verschlüsselung ( $q_{encrypt}$ ) sei als ordinal skaliert definiert. Dann ist Verschlüsselung mit 128 Bit schlechter als eine mit 512 Bit. Daher ist  $\lambda_{S_4}(q_{encrypt}) < \lambda_{S_5}(q_{encrypt})$ . Der Unterschied zwischen den Werten kann aber nicht als Abstand gewertet werden, zum Beispiel hat die numerische

Differenz der Schlüssellängen von 384 Bits keine Bedeutung. Im Video-Beispiel ist das Merkmal *Resolution* ( $q_{resolution}$ ) als ordinal skaliert definiert.

**Operationen** Für Qualitätsmerkmale der Ordinalskala sind neben  $=$  und  $\neq$  die Vergleichsoperatoren  $<$ ,  $\leq$ ,  $\geq$  und  $>$  zulässig. Für einen Requestor ist es im Falle eines ordinal skalierten Merkmals möglich, seine Anforderungen mit Restriktionen und Vergleichen zu versehen. Beispielsweise kann der Wertebereich für die Bildauflösung mit  $q_{resolution} \geq 1024 \times 768$  eingeschränkt werden. Für eine Menge von Ausprägungen kann auch der Median berechnet werden. Der Median bezeichnet eine Grenze zwischen zwei gleich großen Hälften. Er kann mit dem arithmetischen Mittel zusammenfallen, kann aber auch unterschiedlich sein. Außerdem sind die Spanne der Werte (Minimal- und Maximalwert) und Quantile berechenbar.

### Intervallskala

Bei Qualitätsmerkmalen einer Intervallskala sind die Abstände zwischen benachbarten Werten gleich. Dies bedeutet, dass zwischen 4 und 5 derselbe Abstand ist, wie zwischen 27 und 28. Ansonsten gelten die Eigenschaften wie in der Ordinalskala.

**Beispiele** Ein Beispiel für eine Intervallskala sind Temperaturangaben in Grad Celsius ( $^{\circ}\text{C}$ ), da jeder Grad gleich weit vom benachbarten Grad entfernt ist. Bei der Intervallskala ist der Nullpunkt aber nicht als absoluter, aussagekräftiger Nullpunkt definiert. So wurde beim Service  $S_2$  eine Temperatur garantiert:  $\lambda_{S_2}(q_{temp}) = -8$ . Dies ist  $4^{\circ}\text{C}$  kälter als  $\lambda_{S_6}(q_{temp}) = -4^{\circ}\text{C}$ . Andererseits wird  $S_2$  aber nicht doppelt so kalt angeboten wie  $S_6$ . Dies wird auch deutlich, wenn man in die Überlegung  $S_5$  mit einbezieht:  $\lambda_{S_5}(q_{temp}) = +4^{\circ}\text{C}$ .

**Operationen** Für Qualitätsmerkmale der Intervallskala sind nun Addition und Subtraktion erlaubt. Ebenso ist eine Abstandsfunktion definierbar und für eine Menge von Ausprägungen ist ein (arithmetischer) Mittelwert und die Standardabweichung errechenbar.

### Ratioskala

Gegenüber der Intervallskala ist die Ratioskala dort anwendbar, wo ein sinnvoller Nullpunkt existiert. Ansonsten gelten die Eigenschaften wie in der Intervallskala.

**Beispiele** Wie bereits erwähnt, gilt dieses Skalenniveau für Kosten und Lieferzeit. So beträgt die Lieferzeit von  $S_2$ :  $\lambda_{S_2}(q_{time}) = 24$  Stunden und von  $S_6$   $\lambda_{S_6}(q_{time}) = 48$  Stunden, also tatsächlich doppelt so lange. Im Video-Beispiel sind die Werte des Qualitätsmerkmals *ColorDepth* auf einer Ratioskala. Allerdings sind hier nicht alle Werte zulässig, wie in Tabelle 4.1 dargestellt. Weitere Beispiele

für Qualitätsmerkmale mit einer Ratioskala wären Gewicht (beispielsweise angegeben in Kilogramm) oder Entfernung (beispielsweise angegeben in Meter).

**Operationen** Multiplikation und Division sind bei der Ratioskala zusätzlich erlaubt, so dass auch Transformationen dieser Art zulässig sind. Derartige Transformationen verändern zwar die Werte, aber jeweils für alle Ausprägungen derart, dass sich eine äquivalente Information ableiten lässt. Eine „Verdopplung der Temperatur“ ist nicht zulässig, eine „Verdopplung der Kosten“ ist allerdings zulässig.

### Zusammenhang Skalenniveau und Metrik

Merkmale auf einer Intervallskala oder Ratioskala werden auch als *metrisch* bezeichnet. Für diese Merkmale ist in der Regel eine Maßeinheit spezifizierbar, so dass nicht die Bedeutung jedes einzelnen Wertes definiert werden muss. Ebenso sind bei diesen Merkmalen in der Regel Messverfahren definierbar, beispielsweise mit welchem Verfahren die Temperatur gemessen wird. In der Regel ist bei nicht metrischen Merkmalen die Angabe einer Maßeinheit nicht möglich.

Allerdings wurde in den Beispielen gezeigt, dass auch für ordinal skalierte Merkmale (Sicherheit) unter Umständen Maßeinheiten spezifizierbar sind (Schlüssellänge in Bits). Bei nominalen Werten kann durch einen Verweis auf einen Standard ebenfalls eine Art Maßeinheit angegeben werden. Beispielsweise hat Microsoft die so genannte FourCC-Codierung für Video-Codecs standardisiert, die in vier Zeichen im Header einer Video-Datei angibt, welcher Codec für die Anzeige der Daten verwendet werden muss. Soll dieses als Qualitätsmerkmal spezifiziert werden, ist durch einen Verweis auf diesen Standard eine Semantik definiert, die einer Maßeinheit ähnlich ist, denn nicht jeder einzelne Werte ist semantisch zu spezifizieren.

### Zusammenhang Skalenniveau und Art des Wertes

Bei einem nominal skalierten Qualitätsmerkmal ist die Ausprägung immer ein fester Wert. Ab der Ordinalskala kann eine Ausprägung auch einen Maximal- oder Minimalwert darstellen. Mittelwerte beziehungsweise Erwartungswerte sind ebenfalls vom Skalenniveau abhängig, wie oben bereits erläutert wurde. Bei einem nominal skalierten Wert kann lediglich die häufigste Ausprägung als *Mittelwert* angegeben werden, bei einer Ratioskala kann ein Erwartungswert angegeben werden.

### 4.3.6 Anfragen und Restriktionen

Die Suche eines Requestors nach einem Service wird zunächst durch die Suche über eine Funktionalität vorgenommen. Wird die Funktionalität festgelegt, können verfügbare Service-Typen ermittelt werden. Die Qualitätsmerkmale des Service Typs sind dann für eine Anfrage bekannt. Dann kann ein Requestor seine Restrik-

tionen bezüglich der Ausprägungen der Qualitätsmerkmale in folgenden Formen angeben:

1. Angabe von zulässigen Werten:  $\lambda_S(q) \in \{v_1, \dots, v_i\}$ , mit  $v_i \in \text{dom}_q$
2. Angabe von nicht zulässigen Werten:  $\lambda_S(q) \notin \{v_1, \dots, v_i\}$ , mit  $v_i \in \text{dom}_q$
3. Angabe von Ober- oder Untergrenzen für Werte:  
 $\lambda_S(q) \circ v$ , mit  $\circ \in \{<, \leq, \geq, >\}$ .

Die Restriktionen der 1. und 2. Form sind für alle Qualitätsmerkmale möglich, wohingegen Restriktionen der 3. Form erst ab der Ordinalskala zulässig sind. Die konzeptionelle Entscheidung, das Skalenniveau explizit in die Modellierung der Qualitätsmerkmale einzubeziehen bewirkt hier, dass Anfragen eines Requestors allein anhand der definierten Qualitätsmerkmale auf Korrektheit geprüft werden können.

Neben der einfachen Angabe einer Restriktion sind auch komplexere Restriktionen möglich, die sich beispielsweise aus einer disjunktiven oder konjunktiven Verknüpfung der Restriktionen einzelner Qualitätsmerkmale ergeben (zum Beispiel  $R_4$ , Seite 85). Weiterhin sind folgende Möglichkeiten verfügbar, Optimierungswünsche anzugeben:

- Angabe von Optimierungskriterien: Neben den einzuhaltenden Restriktionen kann ein einzelnes Qualitätsmerkmal angegeben werden, nach dem optimiert werden soll, wie minimale Kosten. Hier ist eine Richtung anzugeben, also minimale oder maximale Werte. Eine derartige Optimierung ist erst ab einer ordinalen Skala möglich.

Wenn  $R_4$  um den Zusatz, die Kosten zu minimieren erweitert wird, so ist  $S_2$  mit 275 € der beste Service (siehe Tabelle 4.2).

- Neben zwingenden Restriktionen, die eingehalten werden müssen, ist es möglich, für ein Qualitätsmerkmal eine optionale, wünschenswerte Ausprägung anzugeben. Ein Service, der zusätzlich diesen optionalen Wert anbietet, wird den anderen Services, die alle Restriktionen einhalten bevorzugt. Diese Form der Optimierung ist für alle Skalenniveaus zulässig.

Wird  $R_4$  erweitert um den Zusatz, eine Verschlüsselung von 512 bit zu bevorzugen, so wird  $S_5$  als bester Service gewählt.

Es ist also möglich, auf der konzeptionellen Grundlage des Qualitätsmodells, Anfragen zu formulieren, die automatisiert ausgewertet werden können. Es kann somit die Menge der Services ermittelt werden, die die Restriktionen eines Requestors erfüllen. Wird in der Anfrage ein Optimierungsziel angegeben, so kann sogar eine Rangfolge bezüglich dieses Qualitätsmerkmals automatisiert zusammengestellt werden, so dass der Requestor den besten Service auswählen kann.



### 4.3.7 Nutzenfunktion

In diesem Abschnitt wird das Konzept der Nutzenfunktion vorgestellt, das zur Bestimmung einer Rangfolge unter Berücksichtigung individueller Präferenzen eingesetzt werden kann. Es wird dazu untersucht, wie eine Nutzenfunktion auf dem Qualitätsmodell aufbauen kann und eine mathematische Abbildung individueller Präferenzen vorgenommen werden kann. Im vorherigen Abschnitt ist untersucht worden, wie Restriktionen und Optimierungsziele formuliert werden können. Dabei ist zunächst definiert worden, dass eine Rangfolge bezüglich *eines* Qualitätsmerkmals aufgestellt werden kann. Diese Einschränkung auf nur ein Qualitätsmerkmal ist aber nicht ausreichend.

Sei beispielsweise die Anfrage  $R_4$  (Seite 85) um das Optimierungsziel „minimiere Kosten“ erweitert, so ist zunächst  $S_2$  der beste Service (siehe Tabelle 4.2). Allerdings hat  $S_4$  eine geringere Lieferzeit ( $\lambda_{S_4}(q_{time}) = 18 < \lambda_{S_2}(q_{time}) = 24$ ). Angenommen, der Requestor bevorzugt  $S_4$  gegenüber  $S_2$ , da er für die Zeitersparnis von 6 Stunden bereit ist, die um 45 € höheren Kosten zu akzeptieren. Dieses kann bisher nicht in einer Anfrage ausgedrückt werden. Daher wird hier das Konzept der Nutzenfunktion vorgestellt, mit dem sich das mehrere Qualitätsmerkmale berücksichtigen lassen.

Bei einer Nutzenfunktion [53] wird für die relevanten Qualitätsmerkmale jeweils spezifiziert, wie sie zum Gesamtnutzen eines Services beitragen. Es wird eine Abbildung von den Ausprägungen eines Qualitätsmerkmals auf eine Zahl vorgenommen<sup>5</sup>. Dadurch kann für jeden in Frage kommenden Service der Gesamtnutzen bestimmt werden. Dadurch kann eine entsprechende Rangfolge aufgestellt werden. Ziel ist es dann, den Service auszuwählen, der den höchsten Gesamtnutzen aufweist. Nutzenfunktionen stammen aus der Wirtschaftslehre [53] und werden auch in der Entscheidungslehre [12] verwendet.

Das Listing 4.1 skizziert, wie der Gesamtnutzen eines Services berechnet wird. Es wird für alle Qualitätsmerkmale jeweils dessen Ausprägung ermittelt. Zu dieser Ausprägung kann dann der Nutzenwert bestimmt werden. Der Gesamtnutzen ist dann die Summe über alle Nutzenwerte der einzelnen Qualitätsmerkmale.

Damit der Nutzen für jeden Service bestimmt werden kann, muss für alle Ausprägungen aus der Domäne der Qualitätsmerkmals  $q$  ( $dom_q$ ) der entsprechende Nutzenwert zugewiesen werden. An dieser Stelle ist zu beachten, dass diese Nutzwertzuweisung im Allgemeinen nur für eine spezielle Anfrage gilt. Weiterhin sind je nach Skalenniveau des betrachteten Qualitätsmerkmals unterschiedliche Arten der Nutzwertzuweisung möglich:

- Für nominal skalierte Werte sind den einzelnen Ausprägungen jeweils ein Nutzenwert zuzuweisen.
- Ab einer ordinalen Skala kann ein Wertebereich angegeben werden, in dem für die Ausprägungen die untere und obere Grenze definiert werden. Für

---

<sup>5</sup>Hier wird von einer kardinalen Nutzenquantifizierung ausgegangen, in der die Nutzendifferenz zweier Alternativen eine Bedeutung hat. Der Nutzen ist quantifizierbar, also auch als Zahl darstellbar.

Listing 4.1: Algorithmus zur Berechnung des Gesamtnutzen eines Service

---

```
Prozedur berechneGesamtnutzen(myService) {  
  
  Eingabe: Service myService  
  Ausgabe: int Gesamtnutzen  
  Variablen: Set alleQ,  
              Qualitätsmerkmal q,  
              int Nutzen  
  
  /* alleQ : Menge der Qualitätsmerkmale */  
  /* q : ein Qualitätsmerkmal */  
  
  Gesamtnutzen := 0  
  alleQ := Menge der Qualitätsmerkmale von myService  
  
  Für jedes Qualitätsmerkmal q aus alleQ {  
    Wert := Ausprägung von q  
    Nutzen := Bestimme Nutzen von Wert  
    Gesamtnutzen := Gesamtnutzen + Nutzen  
  }  
  gebe Gesamtnutzen zurück  
}
```

---

---

einen so definierten Wertebereich kann dann ein Nutzenwert spezifiziert werden.

- Ab intervallskalierten Werten sind auch Nutzenfunktionen anzugeben, wie viel Nutzen ein Intervallschritt beiträgt. Dadurch ist eine direkte Berechnung des Nutzens aus der Ausprägung möglich.

Im Allgemeinen kann ein Requestor für einen Service eines festgelegten Typs eine Nutzenfunktion angeben. Dazu ist zunächst für jedes Qualitätsmerkmal zu spezifizieren, welchen Nutzen die Ausprägungen des Qualitätsmerkmals bedeuten.

**Definition 25 Nutzenfunktion:** Eine Nutzenfunktion  $\nu_q$  ist für ein Qualitätsmerkmal  $q$  definiert als Abbildung  $\nu_q : \text{dom}_q \rightarrow \mathbb{Q}$ .

Somit sei eine Nutzenfunktion  $\nu_q$ <sup>6</sup> so definiert, dass die Ausprägungen des Qualitätsmerkmals  $q$  auf einen Nutzenwert aus der Menge der rationalen Zahlen  $\mathbb{Q}$  abgebildet werden. Beispielsweise kann die Nutzenfunktion eines Requestors für die Kosten aus dem Transport-Beispiel aus Abschnitt 4.1.2 definiert sein als:

$$\nu_{q_{cost}}(\lambda_S(q_{cost})) = 1000 - (\lambda(q_{cost}) * 2)$$

Bei steigenden Kosten sinkt somit der Nutzen. Betrachtet man lediglich das Qualitätsmerkmal Kosten, so ergibt sich für den Service  $S_1$  ein Nutzen von 800, für  $S_2$  450 und so weiter. Somit ist der Service mit den geringsten Kosten ( $S_1$ ) derjenige mit dem höchsten Nutzen. Zusätzlich zu den Kosten sei nun eine weitere Nutzenfunktion definiert, die den Nutzen des Qualitätsmerkmals Temperatur  $q_{temp}$  wie folgt bestimmt:

$$\nu_{q_{temp}}(\lambda_S(q_{temp})) = \begin{cases} 200 & \text{falls } 8 < \lambda_S(q_{temp}) \leq 20, \\ 400 & \text{falls } 0 < \lambda_S(q_{temp}) \leq 8, \\ 540 & \text{falls } \lambda_S(q_{temp}) \leq 0, \\ 0 & \text{sonst} \end{cases}$$

Hier seien also Bereiche definiert durch Definition von Ober- und Untergrenzen. Durch die Angabe des Nutzens 0 für sonstige Werte ist der gesamte Wertebereich abgedeckt, so dass jeder möglichen Ausprägung ein Nutzenwert zugewiesen werden kann. Nun können die Nutzen der Ausprägungen der beiden Qualitätsmerkmale Kosten und Temperatur für jeden Service aggregiert werden. Der Gesamtnutzen ergibt ist dabei die Summe der einzelnen Nutzen der Qualitätsmerkmale:

**Definition 26 Service-Nutzen:** Der Service-Nutzen  $\nu(S)$  eines Services  $S$  ergibt sich aus der Summe der Nutzen der Ausprägungen der Qualitätsmerkmale des Services  $S$  wie folgt:

$$\nu(S) = \sum_{q \in Q_T} \nu_q(\lambda_S(q))$$

---

<sup>6</sup>Der griechische Buchstabe Ny ( $\nu$ ) wird dabei als Symbol für die Nutzenfunktion verwendet.

◇

An dieser Stelle sei darauf hingewiesen, dass der Service-Nutzen die Nutzenfunktionen benötigt, die jeweils durch einen Requestor für eine spezielle Anfrage zu formulieren sind. Der Service-Nutzen kann dann für alle Services des betrachteten Service-Typs ermittelt werden. Betrachtet man die beiden Nutzenfunktionen, die für die Qualitätsmerkmale  $q_{cost}$  und  $q_{temp}$  formuliert wurden, so ergibt sich für die angebotenen Transport-Services der folgende Nutzen (in Klammern jeweils der Rang):

$$\nu(S_1) = 800 + 0 = 800 \quad (2.)$$

$$\nu(S_2) = 450 + 540 = 990 \quad (1.)$$

$$\nu(S_3) = 700 + 0 = 700 \quad (3.)$$

$$\nu(S_4) = 360 + 200 = 560 \quad (6.)$$

$$\nu(S_5) = 200 + 400 = 600 \quad (5.)$$

$$\nu(S_6) = 100 + 540 = 640 \quad (4.)$$

Hier weist der Service  $S_2$  den höchsten Nutzen für den Requestor auf, so dass dieser Service die beste Wahl darstellt. Die Rangfolge der Services ergibt sich aus den jeweiligen Service-Nutzen:  $S_2, S_1, S_3, S_6, S_5, S_4$ .

**Zusammenfassung** Eine Nutzenfunktion kann von einem Requestor so definiert werden, dass den möglichen Ausprägungen der Qualitätsmerkmale eines Service-Typs jeweils ein Nutzenwert zugewiesen wird. Dadurch kann für jeden Service dieses Service-Typs ein Service-Nutzen errechnet werden und eine Rangfolge der Services aufgestellt werden. Eine Nutzenfunktion unterstützt damit die rationale Entscheidungsfindung bei der Service-Auswahl.

Es ist zu bedenken, dass eine Nutzenfunktion für alle Qualitätsmerkmale zu formulieren eine zeitaufwendige Aufgabe ist, für die ein Requestor einen menschlichen Experten einsetzen muss. Allerdings ist es dadurch möglich, eine sehr präzise Auswahl des am Besten geeigneten Services zu ermitteln. Dabei können alle relevanten Qualitätsmerkmale berücksichtigt werden und ein komplexes Optimierungsziel mit mehreren Parametern formuliert werden.

## 4.4 Realisierung des Modells

Anhand des Transport-Beispiels aus Abschnitt 4.1.2 wird hier das Qualitätsmodell erstellt und somit eine Realisierung des Meta-Modells konzeptionell geschildert. Dies ist in Abschnitt 4.4.1 dargestellt. Anschließend wird in Abschnitt 4.4.2 eine Realisierung des Qualitätsmodells als relationales Datenmodell vorgestellt.

#### 4.4.1 Realisierung des Transport-Beispiels

Die Qualitätsmerkmale des Service-Typs  $T_{shipping}$  seien wie folgt definiert, so dass sich die Menge  $Q_{T_{shipping}} := \{q_{temp}, q_{cost}, q_{time}, q_{encrypt}, q_{state}\}$  ergibt:

- $q_{temp} := (\mathbb{Z}, \text{MaxGradCelsius}, \text{interval})$
- $q_{cost} := (\mathbb{Q}^+, \text{EuroPreis}, \text{ratio})$
- $q_{time} := (\mathbb{Q}^+, \text{Zeitdauer}, \text{ratio})$
- $q_{encrypt} := (\text{dom}_{q_{encrypt}}, \text{Verschlüsselung}, \text{ordinal})$
- $q_{state} := (\text{dom}_{q_{state}}, \text{Statusüberwachung}, \text{nominal})$

Die Qualitätsmerkmale  $q_{temp}$  und  $q_{cost}$  sind bereits in Abschnitt 4.3.2 beschrieben worden. Die Metrik *Zeitdauer* des Merkmals  $q_{time}$  definiert dieses Merkmal als Zeitwert mit der Maßeinheit Stunden. „Zeitdauer“ sie hier das Symbol für eine präzise und eindeutige Metrik-Spezifikation. Die Domänen der Merkmale  $q_{encrypt}$  und  $q_{state}$  seien wie folgt definiert:

- $\text{dom}_{q_{encrypt}} := \{„128 \text{ bit key“}, „256 \text{ bit key“}, „512 \text{ bit key“}\}$
- $\text{dom}_{q_{state}} := \{„\text{Web-Portal“}, „\text{E-Mail“}, „\text{Telefon“}\}$

Hinzu kommen die Metriken *Verschlüsselung* und *Statusüberwachung*, mit denen jeweils die Bedeutung dieser Werte definiert wird. Hier sollen ebenfalls die Begriffe als Symbole für eine präzise und eindeutige Metrik verwendet werden. Für das ordinal skaliertes Merkmal  $q_{encrypt}$  ist als Bestandteil der Metrik *Verschlüsselung* auch die folgende Ordnung definiert:

- $128 \text{ bit key} < 256 \text{ bit key} < 512 \text{ bit key}$

Die Ausprägungen der einzelnen Services  $S_1$  bis  $S_6$  entspricht dann den in Tabelle 4.2 angegebenen Werten.

#### 4.4.2 Realisierung als Relationales Datenmodell

Im Rahmen eines Projektes wurde eine prototypische, Web-basierte Anwendung realisiert, die eine Service-Auswahl unter Berücksichtigung von Qualitätsmerkmalen erlaubt. Somit wurden diese zentralen Aspekte des Qualitätsmodells implementiert. Einem Anwender wird es ermöglicht, unter verfügbaren Services denjenigen auszuwählen, dessen Qualität den eigenen Anforderungen am Besten entspricht<sup>7</sup>. Diese zu realisierende Funktionalität ist einem Service Repository zuzuordnen, so

---

<sup>7</sup>Zur Einordnung der Service-Auswahl in die Modellierung von Geschäftsprozessen sei auf den Abschnitt 5.2 verwiesen. Hier wird die Bedeutung der Service-Auswahl in diesem Kontext betrachtet. Die Anforderungen an eine Realisierung der Auswahl sind ebenfalls formuliert.

dass die Informationen über die Services, sowie deren Qualität in einer Datenbank gespeichert wurden. Dazu wurde in einem ersten Schritt das Service-Modell und das Qualitätsmodell in ein relationales Datenbankschema transformiert [160].

Die Abbildung 4.3 zeigt einen Auszug aus dem relationalen Datenmodell. Bei der Transformation des Qualitätsmodells in ein relationales Datenmodell sind zur Implementierung einige Anpassungen vorgenommen worden. Beispielsweise werden  $n:m$ -Beziehungen in eine eigene Relation übersetzt, wie in der Abbildung durch die Tabelle `ST_QUALITIES`, die Informationen enthält, welche Qualitätsaspekte (`QUALITYASPECT`) für welche Service-Typen (`SERVICETYPE`) definiert sind. Abgebildet sind die Datenstrukturen (inklusive Datentypen), die Primärschlüssel (PK), Fremdschlüssel (FK) sowie weitere Bedingungen (Unique Key, UK).

In der Abbildung 4.4 sind jeweils die Relationen für die Anfrage-Restriktionen (`QUERY` und `QUERYVALUES`) und die Nutzenfunktion (`USABILITYFUNCTION` und `USABILITYVALUES`) dargestellt. Die Details der bereits in Abbildung 4.3 gezeigten Relationen sind nicht dargestellt. Neben den hier dargestellten Relationen umfasst das Datenmodell noch diverse Views, Sequenzen und Trigger.

Darauf aufbauend wurde eine Anwendung erstellt, die folgende Funktionalitäten realisiert:

- Masterdaten-Management: Eingabemasken für die Daten über Service-Typen und Services, Qualitätsaspekte (inklusive Domänen, Skalen und Metriken). Sowie dazugehörige Anzeige- und Such-Seiten für diese Daten.
- Querybuilder: Für Service-Anfragen wurden zwei Arten realisiert, die in den Abschnitten 4.3.6 und 4.3.7 beschrieben wurden
  - Anfragen und Restriktionen: Mittels geeigneter Eingabe-Dialoge kann ein Benutzer für einen Service-Typ seine Anforderungen an die Qualität definieren, in dem er Restriktionen für die einzelnen Qualitätsaspekte definiert. Zusätzlich kann ein Sortierungskriterium angegeben werden. Die Anfragen können dann so ausgeführt werden, dass die einzelnen Restriktionen mit `AND` oder `OR` verknüpft werden.
  - Nutzenfunktion: Für die einzelnen Qualitätsaspekte eines Service-Typs kann festgelegt werden, welchen Nutzen die jeweiligen Ausprägungen bringen. Hierbei wird zwischen nominal- und ordinal-skalierten Qualitätsmerkmalen einerseits und intervall- und rational-skalierten Qualitätsmerkmalen andererseits unterschieden.
  - Validierungen: Es sind exemplarische Validierungsregeln definiert worden, welche die Korrektheit einer Service-Anfrage beurteilt.

Zur anschaulichen Service-Auswahl wurde die Anwendung mit einer Web-basierten, grafischen Benutzeroberfläche erstellt (siehe Abbildung 4.7). Bei der Service-Auswahl ist der Service-Typ sowie die notwendige und gewünschte Qualität zu spezifizieren. Der Benutzer wird bei der Service-Auswahl unterstützt, bei-

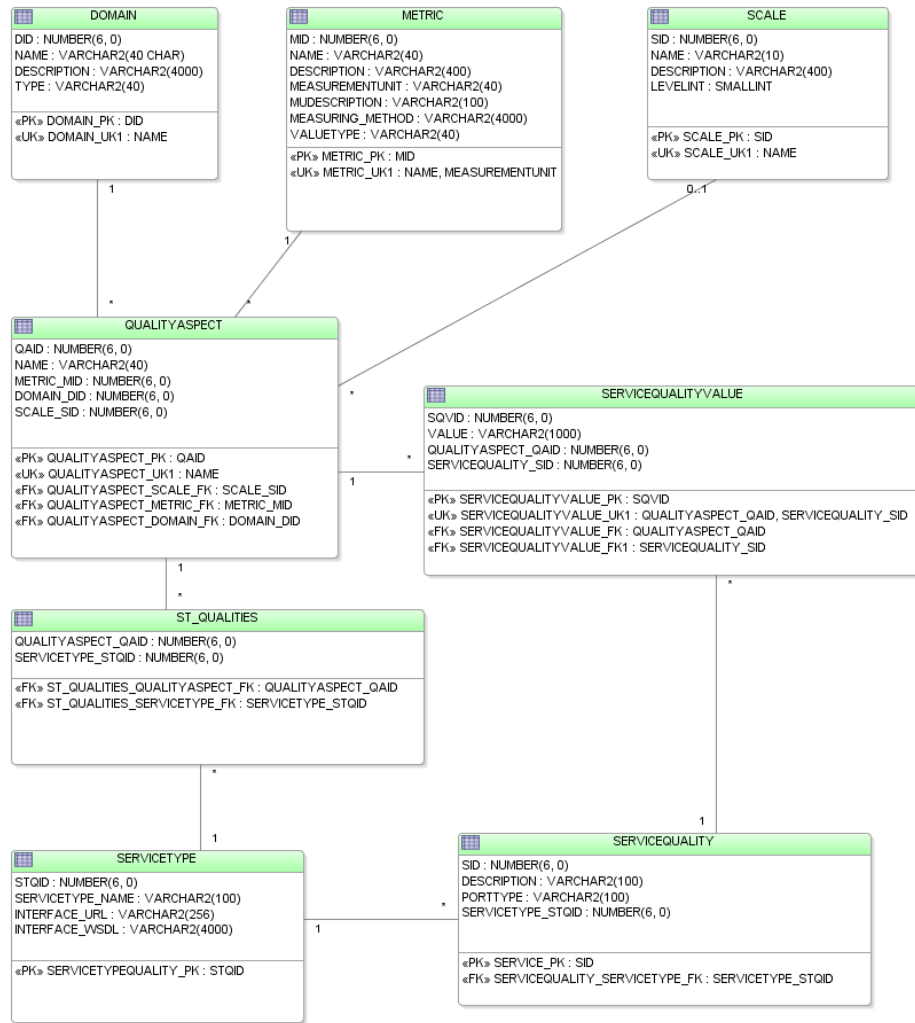


Abbildung 4.3: Relationales Datenmodell (Auszug)

#### 4.4. Realisierung des Modells

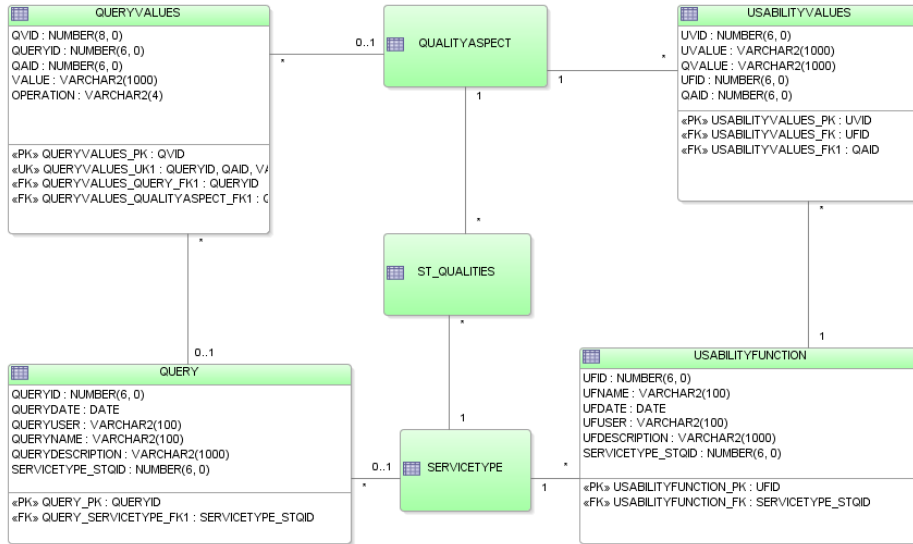


Abbildung 4.4: Relationales Datenmodell der Anfragen und Nutzenfunktion (Auszug)

spielsweise indem nur die definierten Qualitätsmerkmale des Service-Typs angezeigt werden und Validierungsregeln zur Verfügung stehen, um die Korrektheit der Anfrage zu analysieren. Die spezifizierte Anfrage des Benutzers wird ebenfalls in der Datenbank gespeichert. Daraus wird dann eine Datenbank-Anfrage generiert und auf die Datenbasis der verfügbaren Services angewendet.

Die Implementierung wurde durchgeführt mit dem frei verfügbaren Oracle Application Express [108] (APEX), in der Version 3.0.1. Hierbei handelt es sich um ein Entwicklungswerkzeug für Web-Anwendungen. Oracle APEX basiert auf dem relationalen Datenbankmanagementsystem Oracle Database 10g. Weiterhin wurde die integrierte Entwicklungsumgebung Oracle JDeveloper in der Version 10.1.3.2 eingesetzt [109]. Es wird ein HTTP-Server eingesetzt, der die Web-Seiten zur Verfügung stellt. Dieser greift unter Benutzung eines speziellen Moduls (`mod_plsql`) auf das Datenbanksystem zu. Oracle APEX ist dabei zuständig für die Generierung der Web-Seiten und die Behandlung der Anfragen. Abbildung 4.5 skizziert diese Architektur.

Da die Anfragen in Relationen gespeichert werden, sind Sie wiederverwendbar und können angepasst werden. Zusätzlich werden die Service-Anfragen als Verknüpfung der Relationen umgesetzt, so dass als Ergebnis die Reihenfolge der gültigen (dass heißt den Restriktionen entsprechenden beziehungsweise dem Nutzen nach sortierte) Services liefert. Listing 4.2 zeigt eine Service-Anfrage, die auf einer Nutzenfunktion basiert. Eine derartige Verknüpfung von Relationen ist innerhalb der relationalen Datenbank derart möglich, dass Optimierungen relationaler Datenbanken Anfragen effizient eingesetzt werden können.



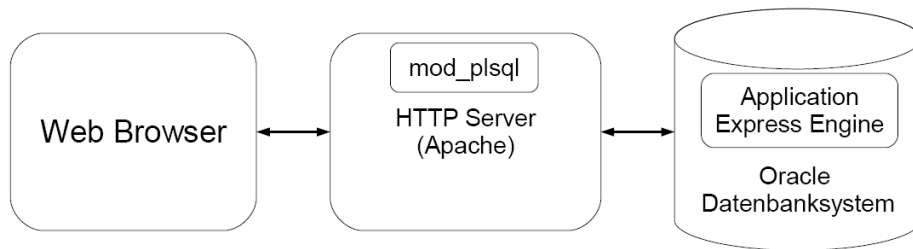


Abbildung 4.5: Architektur-Skizze der Oracle APEX Anwendung [108]

Homepage > Querybuilder Servicetype > UsabilityFunctions for Servicetype > Show Usabilityfunction Details > Tabular Form on Usabilityvalues

### Tabular Form for Usabilityfunction 4

<input type="checkbox"/>	Usabilityfunction ID	Qualityaspect	Quality Value	Usability Value
<input type="checkbox"/>	4	Ausführungsdauer	1	-100
<input type="checkbox"/>	4	Kosten	10	-50
<input type="checkbox"/>	4	Status	E-Mail	1000
<input type="checkbox"/>	4	Status	Web-Portal	5000
<input type="checkbox"/>	4	Status	Telefon	780
<input type="checkbox"/>	4	Sicherheit	128 bit key	10000
<input type="checkbox"/>	4	Sicherheit	512 bit key	15000
<input type="checkbox"/>	4	Temperatur	1	-10

1 - 8

Abbildung 4.6: Eingabe der Nutzenwerte für eine Nutzenfunktion (Screenshot)

Konkrete Werte für einen Nutzenfunktion können über eine Eingabemaske angegeben werden, wie Sie in Abbildung 4.6 dargestellt ist. Dabei werden nur die Qualitätsmerkmale angeboten, die für den gewählten Service-Typ definiert sind. Die Spalte Usability Value enthält den Nutzwert und die Spalte Quality Value enthält die entsprechende Ausprägung des Qualitätsmerkmals. Hier bedeutet der Wert 1 bei Quality Value für das Qualitätsmerkmal Kosten, dass jeder (zusätzliche) Euro einen negativen Einfluss von 100 auf den Gesamtnutzen hat. Hat das Qualitätsmerkmal Sicherheit die Ausprägung `128 bit key`, so trägt dies einen Nutzen von 10000 zum Gesamtnutzen bei.

Das Listing 4.2 zeigt die Ausführung der Nutzenfunktion durch Verknüpfung der oben dargestellten Relationen. Zu erkennen ist, dass die Rangfolge der Skalen ausgenutzt wird, indem zwischen `ratio/interval` (`SCALE.LEVELINT > 2`) und `nominal/ordinal` (`SCALE.LEVELINT <= 2`) unterschieden wird. Für erstere wird jeweils der Nutzen anhand der konkreten Werte ermittelt, indem beispielsweise für jeden Euro beim Qualitätsmerkmal Kosten ein bestimmter Nutzen berechnet wird, während bei letzteren der konkrete Wert des Qualitätsmerkmals nur dann berücksichtigt wird, wenn diese Ausprägung einen definierten Nutzen zugeordnet hat. Abbildung 4.7 zeigt das Ergebnis einer Nutzenfunktions-Anfrage und Abbildung 4.8 erklärt, wie sich der Nutzwert für den Service mit der ID 500 zusammensetzt.

Unabhängig von der konkreten Implementierung der Anwendung, liegt das Qualitätsmodell in einer relationalen Datenbank vor und kann somit für verschiedensten Anwendungsszenarien wiederverwendet werden. Durch einen Zugriff über Standards wie SQL kann somit die Service-Anfrage für anderen Anwendungen genutzt werden [160]. (Siehe hierzu auch Abschnitt 5.2.)

## 4.5 Diskussion möglicher Erweiterungen

In diesem Abschnitt werden mögliche Erweiterungen skizziert und auf weitere Aspekte eines Qualitätsmerkmals eingegangen. Es werden dadurch einige der Annahmen diskutiert beziehungsweise wird beschrieben, welche Änderungen am Modell bei geänderten Annahmen notwendig sind.

### 4.5.1 Maßeinheiten

In dem Qualitätsmodell kann für ein Qualitätsmerkmal eine Maßeinheit als Teil der Metrik spezifiziert werden. Beispielsweise wurde für  $q_{cost}$  als Maßeinheit € festgelegt, für  $q_{temp}$  °C. Die Einschränkung auf *eine* Maßeinheit kann aufgehoben werden, indem mehrere gültige Maßeinheiten spezifiziert werden. Dadurch kann ein Provider jeweils die für ihn relevante Maßeinheit benutzen und ein Requestor andererseits die für ihn interessanten Maßeinheiten verwenden. Somit wird eine komfortablere Vergleichbarkeit der einzelnen Werte ermöglicht. Die Umrechnung der Werte ist dann ebenfalls zu definieren. Grundsätzlich muss eine Einigung

Listing 4.2: SQL-Ausdruck zur Berechnung des Gesamtnutzen aller Services eines Service-Typs

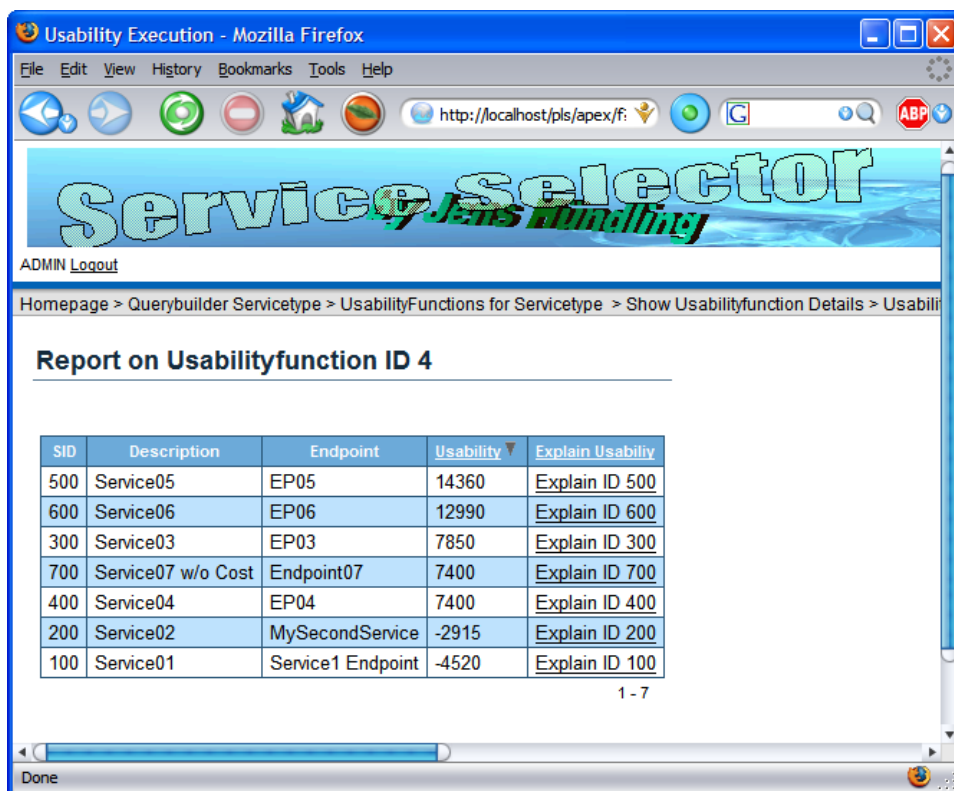
```

SELECT "SERVICEQUALITY"."SID" as "SID", /*Service-ID*/
        "SERVICEQUALITY"."DESCRIPTION" as "DESCRIPTION",
        "SERVICEQUALITY"."PORTTYPE" as "Endpoint",
        /*Aufsummierung des Nutzens */
        /*Intervall- und ratio-skaliert */
        (SUM (CASE WHEN "SCALE".LEVELINT > 2
                /* Berechnung des Nutzwertes */
                THEN (TO_NUMBER("USABILITYVALUES".UVALUE) *
                    (TO_NUMBER("SERVICEQUALITYVALUE".VALUE)
                    / TO_NUMBER("USABILITYVALUES".QVALUE)))
                /* Nominal- und ordinal-skaliert */
                WHEN "SCALE".LEVELINT <= 2
                /* Nutzwert übernehmen, falls definiert(s.u.)*/
                THEN (TO_NUMBER("USABILITYVALUES".UVALUE))
                END ) as "Usability"
FROM
        "SCALE" , "QUALITYASPECT" ,
        "USABILITYFUNCTION" , "SERVICEQUALITY" ,
        "SERVICEQUALITYVALUE" , "USABILITYVALUES"
WHERE
        "USABILITYVALUES"."UFID"="USABILITYFUNCTION"."UFID"
        /* Variable für die Nutzenfunktions-ID */
AND "USABILITYVALUES"."UFID" = :P32_UFID
AND "SERVICEQUALITYVALUE"."SERVICEQUALITY_SID" =
        "SERVICEQUALITY"."SID"
        /* Service-Typ wird durch Nutzenfunktion definiert*/
AND "SERVICEQUALITY"."SERVICETYPE_STQID" =
        "USABILITYFUNCTION"."SERVICETYPE_STQID"
AND "SCALE"."SID"="QUALITYASPECT"."SCALE_SID"
        /* Das Skalenniveau ist entweder interval/ratio...*/
AND ("SCALE".LEVELINT > 2 OR
        /* ... oder für den Wert existiert ein Nutzwert */
        ("SCALE".LEVELINT <= 2
        AND "USABILITYVALUES".QVALUE =
        "SERVICEQUALITYVALUE".VALUE))
AND "USABILITYVALUES"."QAID" = "QUALITYASPECT"."QAID"
AND "USABILITYVALUES"."QAID" =
        "SERVICEQUALITYVALUE"."QUALITYASPECT_QAID"
GROUP BY "SERVICEQUALITY"."SID" ,
        "SERVICEQUALITY"."DESCRIPTION",
        "SERVICEQUALITY"."PORTTYPE"

```

---

## 4.5. Diskussion möglicher Erweiterungen



Usability Execution - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost/pls/apex/f/

# Service Selector

ADMIN Logout

Homepage > Querybuilder Servicetype > UsabilityFunctions for Servicetype > Show Usabilityfunction Details > Usabili

## Report on Usabilityfunction ID 4

SID	Description	Endpoint	Usability	Explain Usability
500	Service05	EP05	14360	<a href="#">Explain ID 500</a>
600	Service06	EP06	12990	<a href="#">Explain ID 600</a>
300	Service03	EP03	7850	<a href="#">Explain ID 300</a>
700	Service07 w/o Cost	Endpoint07	7400	<a href="#">Explain ID 700</a>
400	Service04	EP04	7400	<a href="#">Explain ID 400</a>
200	Service02	MySecondService	-2915	<a href="#">Explain ID 200</a>
100	Service01	Service1 Endpoint	-4520	<a href="#">Explain ID 100</a>

1 - 7

Done

Abbildung 4.7: Ergebnis der Nutzenfunktions-Anfrage auf die Services des Logistikbeispiels (Screenshot)

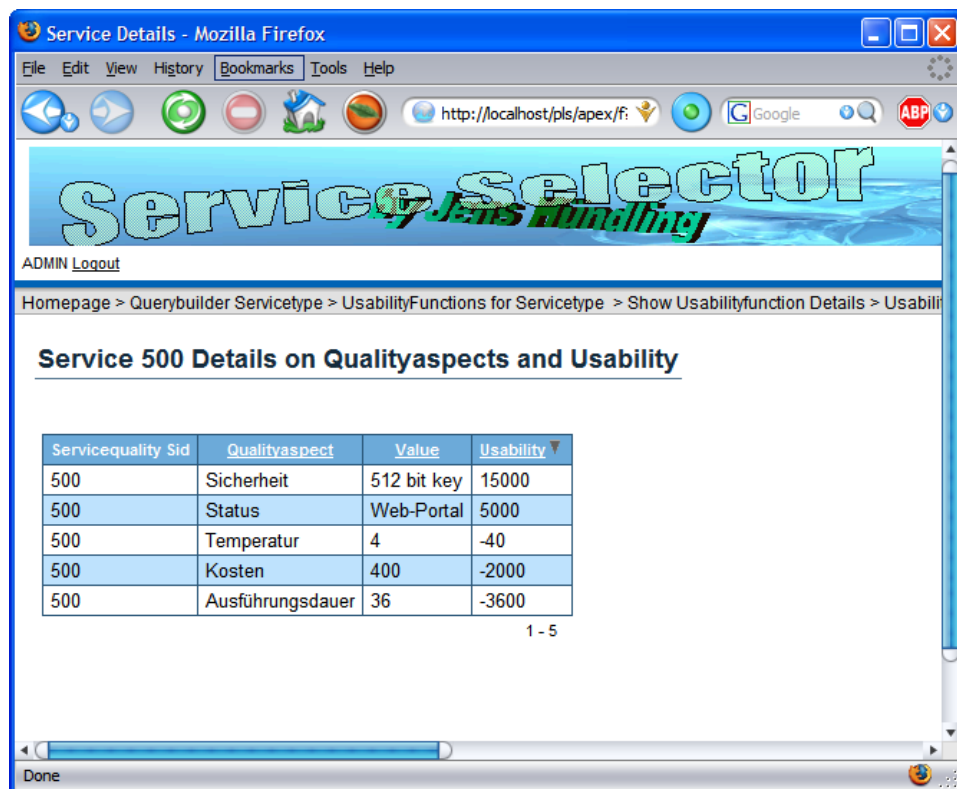


Abbildung 4.8: Zusammensetzung des Gesamtnutzen für den Service mit der ID 500 (Screenshot)

auf die Angabe der Werte in einer definierten Maßeinheit beziehungsweise auf die Umrechnung zwischen den Maßeinheiten erzielt werden. Es ist dann je Maßeinheit eine Abbildung zu definieren, die Werte der einen Maßeinheit in eine andere Maßeinheit abbildet.

**Beispiele** Angenommen, für das Qualitätsmerkmal  $q_{temp}$  kann die Angabe der Ausprägung in °F und °C erfolgen. Für diese Maßeinheiten kann die Umrechnung nach dem folgenden Verhältnis erfolgen:

$$\begin{aligned}\text{°F} &= \text{°C} * 1,8 + 32 \\ \Leftrightarrow \text{°C} &= \frac{(\text{°F} - 32)}{1,8}\end{aligned}$$

Als weiteres Beispiel sollen Maßvielfache betrachtet werden: Wertangaben in Kilometer, Meter und Zentimeter sind jeweils Vielfache voneinander. Hier kann als Basiseinheit ein Meter definiert werden und jeweils die Umrechnung definiert werden. Weiterhin können verschiedene Maßsysteme wie metrisches System und angloamerikanisches Maßsystem betrachtet werden. Auch hierbei sind Umrechnungen in Formeln möglich. Allerdings wird dieses erschwert durch verschiedene Maßvielfache wie Kilometer (1000 Meter), Foot (12 inch), Yard (3 foot), etc. Eine Umsetzung basiert auf einer festgelegten Maßeinheit und der Angabe von Formeln zur jeweiligen Umrechnung in diese Basis-Maßeinheit.

Für das Qualitätsmerkmal  $q_{cost}$  können ebenfalls weitere Maßeinheiten sinnvoll sein, beispielsweise US-\$. Hier ist dann keine feste Abbildung anzugeben, sondern eine Spezifikation, wie der aktuell gültige Wechselkurs zu ermitteln ist.

**Änderungen am Modell** Die Berücksichtigung von verschiedenen Maßeinheiten ist als Bestandteil der Metrik anzusehen. Wie oben beschrieben, muss die Umrechnung zwischen den Maßeinheiten in der Metrik definiert werden. Die Umrechnung muss eindeutig und präzise erfolgen, sowie für alle Maßeinheiten möglich sein. Dann bleibt die Vergleichbarkeit der Ausprägungen gegeben. Die Vergleichbarkeit der Services bleibt erhalten, wenn die Ausprägungen über eine Basis-Maßeinheit erfolgt.

Diese Erweiterung erhöht zwar den Komfort für den Provider bei der Veröffentlichung seiner Services und für den Requestor bei der Formulierung seiner Restriktionen und Wünsche in der Anfrage, aber es wird keine erweiterte Ausdrucksfähigkeit erworben. Grundsätzlich sind außerhalb der Metrik keine Änderungen am Modell erforderlich. Bei einer Umsetzung ist allerdings ein erhöhter Aufwand erforderlich, da die Umrechnung realisiert werden muss.

### 4.5.2 Richtung der Qualitätssteigerung

Es ist zusätzlich in der Metrik spezifizierbar, welche Werte besser sind, also in welcher Richtung die Qualität steigt. Dies wurde bereits in Abschnitt 4.1 im Kontext

der Beispiele diskutiert. Die Richtung der Qualitätssteigerung bedeutet, dass angegeben werden kann ob ein höherer Zahlenwert eine bessere Qualität bedeutet und somit wünschenswerter ist als ein niedrigerer.

**Beispiele** Das Qualitätsmerkmal Kosten ist aus Sicht des Requestors so zu definieren, dass die Qualität mit sinkenden Kosten steigt. Das bedeutet, dass ein günstigerer Service aus seiner Sicht eine höhere Qualität hat. Hierbei ist zu berücksichtigen, dass Requestor und Provider gegensätzliche Wünsche haben. Für den Requestor ist das Qualitätsmerkmal Kosten offensichtlich ein Merkmal, dass so gering wie möglich ausfallen sollte, wohingegen der Provider es für wünschenswert hält, höhere Kosten verlangen zu können. Ähnliches gilt für ein Qualitätsmerkmal der Ausführungsdauer.

Problematisch ist hierbei unter anderem, dass es für einen Requestor ebenfalls Obergrenzen und Untergrenzen geben kann. Beispielsweise ist für einen Lebensmitteltransport generell eine niedrigere Temperatur wünschenswerter, aber nicht jede Ware darf gefroren werden, also negativen Temperaturen während des Transports ausgeliefert sein. Für andere Transportgüter kann auch eine möglichst hohe Temperatur vorteilhaft sein. Somit ist für das Merkmal Temperatur im Allgemeinen keine Richtung definierbar. Weiterhin kann eine Richtung nicht für Qualitätsmerkmale eines nominalen Skalenniveaus angegeben werden, da keine Ordnung für die Werte vorliegt.

**Änderungen am Modell** Die Richtung kann als eigenständiger Aspekt der Qualitätsmerkmale angesehen werden. Somit kann ein weiteres Element neben Metrik, Domäne und Skalenniveau modelliert werden.

Abschließend lässt sich feststellen, dass eine Erweiterung des Modells um eine vordefinierte Richtung der Qualitätssteigerung nicht sinnvoll ist, insbesondere wenn man vom Requestor spezifizierte Nutzenfunktionen berücksichtigt, wie in Abschnitt 4.3.7 beschrieben. Diese berücksichtigen keine vordefinierte Richtung sondern ermitteln einen Nutzen, der von verschiedenen Qualitätsmerkmalen abhängt. Allerdings ist ein Vorteil einer vordefinierten Richtung der Qualitätssteigerung, dass vom Broker auch dann eine Rangfolge der Services angegeben werden kann, wenn kein Optimierungskriterium vom Requestor angegeben wurde.

### 4.5.3 Komplexe Qualitätsmerkmale

Entsprechend komplexen Datenstrukturen, wie sie aus Programmiersprachen bekannt sind (*Records*), können auch Qualitätsmerkmale als komplexe Strukturen modelliert werden. Somit besteht die Ausprägung nicht aus einem Wert, sondern aus mehreren Werten, jeweils für die Einzelteile des Merkmals. Weiterhin sind Mehrfachwerte in Feldern (*Arrays*), Listen oder Mengen definierbar. Bei komplexen Qualitätsmerkmalen sind eigene Vergleichsoperatoren zu definieren und zur Verfügung zu stellen. Gleiches gilt für die Aggregation komplexer Qualitätsmerkmale.

**Beispiel** Als Beispiele sei hier angenommen, dass sich die Kosten aus verschiedenen Einzelwerten zusammensetzen, die einzeln identifizierbar sind: Grundgebühr, Steueraufschlag und Courtage. Die Summe dieser Einzelwerte ergibt dann den Gesamtwert.

Im Transport-Beispiel ist das Qualitätsmerkmal Statusüberwachung ( $q_{state}$ ) definiert. Angenommen, die definierten Ausprägungen schließen sich nicht gegenseitig aus, so dass ein Provider sowohl *Web-Portal* als auch *Telefon* anbieten kann. Somit sind mehrwertige Ausprägungen möglich. Dies kann modelliert werden, indem der Wertebereich entsprechend erweitert wird, so das *Web-Portal und Telefon* einen zulässigen Wert darstellt. Dadurch kann eine Ausprägung angegeben werden, die in die unterschiedlichen Aussagen über die angebotene Möglichkeiten zur Statusüberwachung zerlegt werden können. Somit ist ebenfalls der Vergleich anzupassen: Wenn ein Requestor die Ausprägung Web-Portal benötigt, so ist „Web-Portal und Telefon“ eine gültige Ausprägung.

Das Qualitätsmerkmal *Resolution* des Video-Services sei definiert als ordinales Merkmal mit der Maßeinheit  $\text{Pixel} \times \text{Pixel}$ . Dieses Qualitätsmerkmal ist alternativ aufteilbar in *ResolutionX* und *ResolutionY*, jeweils mit der Maßeinheit  $\text{Pixel}$ . Dann ist jeweils eine Ratioskala anwendbar, allerdings sind Vergleiche nur über eines der beiden Merkmale wenig sinnvoll.

**Änderungen am Modell** Am Modell sind keine Erweiterungen vorzunehmen, wenn komplexe Qualitätsmerkmale modellierbar sein sollen. Vielmehr ist jeweils eine geeignete Domäne anzugeben, welche die Wertebereiche für die Einzelteile des zusammengesetzten Merkmals definiert. Es muss also die Beschreibung der Domäne entsprechend erweitert werden. Hierzu können Techniken aus dem Bereich der Datenmodellierung verwendet werden, wie beispielsweise das Entity-Relationship-Modell [33]. Weiterhin sind die Vergleichsoperationen explizit zu definieren. Eine mögliche Realisierung für mehrwertige Ausprägungen ist bereits beispielhaft angegeben worden.

Problematisch ist die Einordnung in die Skalenniveaus: Wenn keine Ordnung über alle Einzelteile des Merkmals definiert werden kann, so ist die Nominalskala anzuwenden. Je nach weiterer Verwendung ist auch eine Zerlegung in einzelne Qualitätsmerkmale modellierbar. Die Erstellung einer Rangfolge ist ebenfalls aufwendiger, wenn komplexe Qualitätsmerkmale zulässig sind. Dieses ist bei der Erstellung des Qualitätsmodells für einen Service-Typ zu berücksichtigen.

### 4.5.4 Allgemeine und spezielle Qualitätsmerkmale

Ein Aspekt zur Erweiterung des Modells der Qualitätsmerkmale basiert auf der Idee der Unterscheidung in allgemeine und spezielle Qualitätsmerkmale. Unter allgemeinen Qualitätsmerkmalen werden diejenigen verstanden, die einem allgemein akzeptierten und standardisierten Grundwortschatz entstammen. Diese können dann für viele verschiedene Service-Typen angewandt werden. Dadurch kann der Modellierungsaufwand für das Qualitätsmodell eines Service-Typs verringert



werden. Zusätzlich können auch Services unterschiedlicher Service-Typen miteinander verglichen werden, wenn dasselbe Qualitätsmerkmal für beide Service-Typen verwendet wird. Spezielle Eigenschaften sind demgegenüber nur für einen Service-Typ definierte Qualitätsmerkmale.

**Beispiele** Für allgemeine Qualitätsmerkmale kann beispielsweise eine Ontologie benutzt werden [47, 143]. Dieser Grundwortschatz wird standardisiert vorgegeben. Im Kontext von Web Services sind beispielsweise mit WS-Policy [176] und WS-Security [103] bereits Begriffe für den Sicherheitsaspekt definiert worden, die als Ausprägungen für ein entsprechendes Qualitätsmerkmal verwendet werden können. Ebenso wird mit WS-Coordination [45] und WS-Transaction [88, 50] eine Grundwortschatz für Transaktionalität definiert, der auch als Qualitätsmerkmal angesehen werden kann. In einzelnen Branchen können allgemein gültige Begriffe definiert werden, indem sich alle Teilnehmer auf diese Begriffe einigen.

**Änderungen im Modell** In dem Modell dieser Arbeit ist durch das Konzept, dass die Qualitätsmerkmale je Service-Typ definiert sind, eine derartige Charakterisierung der Eigenschaften nicht notwendig. Ein Service-Typ aggregiert Qualitätsmerkmale. Die definierten Qualitätsmerkmale können durchaus auch für andere Service-Typen angegeben werden. In Qualitätsmodellen, in denen es kein Konzept der Service-Typen gibt, hat dieser Aspekt eine größere Bedeutung.

### 4.5.5 Logische und physische Qualitätsmerkmale

Ein weiterer Aspekt der Qualitätsmerkmale stellt die Abgrenzung zwischen logischen und physischen Qualitätsmerkmalen dar. Dieser Aspekt zielt auf die Messbarkeit der Merkmale ab. Es gibt für jedes Merkmal eine Möglichkeit, die Ausprägung zu messen oder zu überprüfen. Wie, wo und wann gemessen wird (Messverfahren) und was als Grundlage (Maßeinheit) dient, wird in der Metrik spezifiziert (siehe Definition 19). Für physische Qualitätsmerkmale ist ein eindeutiges, physisch realisierbares Messverfahren vorgegeben, um so einen objektiven Wert zu erhalten. Bei logischen Qualitätsmerkmalen sind die Werte nur schwer ermittelbar und werden beispielsweise mittels einer Heuristik bestimmt. Weiterhin sind die Werte einiger Qualitätsmerkmale nur schwer objektiv zu ermitteln oder werden durch das Messverfahren verfälscht. In der Diskussion zum Begriff Qualität in Abschnitt 4.3.1 wurde diese Problematik bereits geschildert. Somit ist eine logische Eigenschaft nicht im klassischen Sinne durch ein physisches Messverfahren ermittelbar.

**Beispiele** Qualitätsmerkmale wie Temperatur, Kosten, Größe und Gewicht sind mit physischen Messverfahren und technischen Hilfsmitteln leicht messbar. Als logische Merkmale seien die Verschlüsselung aus dem Transport-Beispiel betrachtet: Dem reinen Datenstrom ist die Länge des verwendeten Schlüssels nicht direkt anzusehen.

Als weiteres Beispiel für ein logisches Merkmal sei die Verfügbarkeit für einen Service als Qualitätsmerkmal definiert: Wenn eine Verfügbarkeit von 99,9% garantiert wird, so muss theoretisch immer gemessen werden, ob der Service verfügbar ist. Hier ist als Hilfsmittel in der Metrik eine Heuristik definierbar, so dass von 1000 Anfragen in einem gewissen Zeitraum, nur maximal eine Anfrage fehlschlagen darf. Für einen einzelnen Service-Aufruf ist der Service aber nur verfügbar oder nicht. Ein weiteres Beispiel für ein logisches, schwer zu messendes Qualitätsmerkmal wäre die Mean Time Between Failures (MTBF).

**Änderungen im Modell** Das Messverfahren ist als Bestandteil der Metrik zu definieren, wie bereits beschrieben wurde. Hierbei ist das physische Messverfahren zu spezifizieren beziehungsweise die Heuristik zur Ermittlung und Überprüfung anzugeben. Das Modell selbst ist somit nicht zu ändern. Dennoch kann dieser Aspekt bei der Erstellung eines Qualitätsmodells berücksichtigt werden, indem als Bestandteil der Metrik spezifiziert wird, ob es sich um ein physisches oder logisches Merkmal handelt.

### 4.5.6 Statische und dynamische Qualitätsmerkmale

Ein sich änderndes oder von äußeren Einflüssen abhängiges Qualitätsmerkmal wird als dynamisch bezeichnet. Davon lassen sich die statischen Eigenschaften abgrenzen, die konstant über den Zeitverlauf bleiben. Dynamische Qualitätsmerkmale können von einer Ressource abhängen: Zur Leistungserbringung werden vom Provider Ressourcen eingesetzt, somit sind diese Ressourcen die Grundlage der angebotenen Leistung (siehe auch Definition 6, Seite 21). Einige Qualitätsmerkmale sind von diesen Ressourcen direkt abhängig. Wenn es bei einer hohen Auslastung zu Ressourcen-Engpässen kommen kann, kann die Qualität nicht eingehalten werden. Dieses ist für den Provider relevant, da er die Ressourcen seiner Services so verwalten und einsetzen muss, dass er angebotene Qualitätsmerkmale und Service Level Agreements einhalten kann. Wird eine in einem SLA vereinbarte Qualität nicht eingehalten, so muss üblicherweise eine Konventionalstrafe gezahlt werden.

**Beispiele** Intuitives Beispiel für ein dynamisches Qualitätsmerkmal ist die Verfügbarkeit des Dienstes über einen Endpunkt. Ist der Datenverkehr zum Service aufgrund einer extrem hohen Nachfrage sehr hoch, so kann sich das negativ auf das Antwortzeitverhalten des Services auswirken. Im Video-Beispiel ist die *Frame-Rate* dynamisch (abhängig von der verfügbaren Bandbreite), während *Resolution* statisch ist. Im Transport-Beispiel ist die Verschlüsselung der Kommunikation ein statisches Qualitätsmerkmal, da sie immer in der gleichen, vereinbarten Qualität erfolgt.

Weiterhin sind zur Leistungserbringung des Transport-Beispiels LKW's und Fahrer nötig. Bei einer hohen Auslastung oder durch den Ausfall von einzelnen Ressourcen können nun unter Umständen Lieferzeiten nicht mehr eingehalten werden.

**Änderungen im Modell** Die Charakterisierung eines Qualitätsmerkmals als dynamisch oder statisch kann in der Metrik erfolgen. Für einen Provider ist dieser Aspekt beim Service Management relevant, da er die dynamischen Qualitätsmerkmale beobachten muss und beeinflussen kann. Aus Sicht eines Requestors ist dieser Aspekt aber prinzipiell nicht relevant, da er alle angebotenen Qualitätsmerkmale gleich behandelt und vom Provider verlangt, dass die angebotene Qualität eingehalten wird. Dies gilt insbesondere dann, wenn ein Service Level Agreement abgeschlossen wurde. Weiterhin sind dynamische Qualitätsmerkmalen in der Regel als Obergrenzen beziehungsweise Untergrenzen definiert, während statische Qualitätsmerkmale als garantierter, fester Wert definiert sind.

### 4.6 Zusammenfassung

In diesem Kapitel ist ein Qualitätsmodell entworfen worden, mit dem es möglich ist, Qualitätsmerkmale für Services abzubilden. Dieses kann die Beschreibung der Services beim Broker erweitern. Somit ist es möglich, unterschiedliche Services des gleichen Service-Typs bezüglich ihrer Qualitätsmerkmale zu vergleichen und der Broker kann automatisiert überprüfen, ob ein Service den Anforderungen eines Requestors genügt. Werden Optimierungskriterien angegeben, lässt sich ermitteln, welcher Service die Anforderungen des Requestors am besten erfüllt. Existieren mehrere Services, welche die Anforderungen erfüllen, so besteht zusätzlich die Möglichkeit eine Rangfolge anzugeben. Sollte ein gewählter Service wider Erwarten nicht verfügbar sein, so spezifiziert die Rangfolge alternative Services mit passenden Qualitätsmerkmalen. Dieses ist nur möglich, da die Ausprägungen der Qualitätsmerkmale von Services des gleichen Service-Typs vergleichbar sind. Somit wurden die Anforderungen an das Qualitätsmodell erfüllt.

Die zentrale Bedeutung der Werte, welche die Ausprägungen der Qualitätsmerkmale darstellen, ist untersucht worden. Sie werden durch die Angabe einer Domäne, eines Skalenniveaus und einer Metrik, die Bedeutung, Maßeinheit und Messverfahren beschreibt, als Qualitätsmerkmal modelliert. Mit diesem Meta-Modell kann nun für jeden Service-Typ ein Modell erstellt werden, dass die für den jeweiligen Service-Typ relevanten Qualitätsmerkmale spezifiziert. Wie anhand der Beispiele und potentiellen Missverständnissen aufgezeigt wurde, ist bei einer Umsetzung des Meta-Modells vor allem darauf zu achten, dass die Metrik präzise, vollständig und eindeutig spezifiziert wird. Dazu sind Angaben über die Qualitätsmerkmale so zu treffen, dass Missverständnisse vermieden werden und die Qualität der erbrachten Leistung den Erwartungen und Anforderungen des Requestors entsprechen. Durch die Spezifikation der Metrik werden Missverständnisse und fehlerhafte Annahmen über die Qualitätsmerkmale eingeschränkt. Letztendlich werden Angebot und Nachfrage besser zusammengeführt, da die Bedeutung der Qualitätsmerkmale und der Werte spezifiziert sind.

Anhand von Beispielen ist gezeigt worden, wie Qualitätsmerkmale spezifiziert werden können, wie Anfragen gestellt werden können und welche Vergleichsope-

ratoren und Optimierungskriterien für welches Skalenniveau möglich sind. Zusätzlich sind komplexe Nutzenfunktionen definierbar, die über normale Restriktionen und die Angabe eines einzelnen zu optimierenden Merkmals weit hinausgehen. Eine Nutzenfunktion bildet die individuellen Präferenzen eines Requestors für eine Anfrage ab. Es sind weiterhin verschiedene Erweiterungen und Möglichkeiten zur Charakterisierung für Qualitätsmerkmale betrachtet worden, die auf der Basis des Modells vorgenommen werden können. Es ist jeweils darauf hingewiesen worden, wie diese Aspekte durch das Qualitätsmodell unterstützt werden. Nachdem das Qualitätsmodell nun formuliert wurde, wird es im nächsten Kapitel weiter untersucht, indem mögliche Anwendungen und eine praktische Umsetzung vorgestellt werden.



## Kapitel 5

# Geschäftsprozesse

In diesem Kapitel wird die Modellierung von Qualitätsmerkmalen für Services im Kontext von Geschäftsprozessen untersucht. Dazu wird das im vorigen Kapitel formulierte Qualitätsmodell verwendet und es werden die Vorteile aufgezeigt, die sich für die Geschäftsprozessmodellierung und -optimierung ergeben. Insbesondere wird die Service-Auswahl untersucht und Aggregationsmuster für Qualitätsmerkmale formuliert.

Im Rahmen des Service-Modells wurde bereits in Abschnitt 3.4.3 erläutert, dass die Prinzipien der Service-Orientierung eine vergleichsweise einfache Integration verschiedener Services zu einem zusammengesetzten Service erlauben. Darüber hinaus werden die Vorteile der serviceorientierten Architektur besonders dann deutlich, wenn Geschäftsprozesse betrachtet werden, die während ihrer Ausführung Services nutzen [58, 61, 129, 161, 172]. Bisher ist untersucht worden, wie die konzeptionelle Grundlage des Qualitätsmodells dazu beiträgt, aus verschiedenen Services den für eine Anfrage am besten geeigneten Service auszuwählen. Hier wird nun untersucht, welche Vorteile sich bei der Modellierung von Geschäftsprozessen realisieren lassen.

Zu Beginn des Kapitels werden zunächst die Grundlagen von Geschäftsprozessen in Abschnitt 5.1 vorgestellt. Dabei werden die zentralen Begriffe definiert und Aspekte der Prozessbeschreibungen vorgestellt. Die Auswahl des am besten geeigneten Services für einen Prozessschritt kann durch das Qualitätsmodells konzeptionell unterstützt werden. Daher wird in Abschnitt 5.2 beschrieben, wie die Auswahl eines Services unter Benutzung des Qualitätsmodells durchgeführt werden kann und die resultierenden Konsequenzen werden diskutiert. Die Leistung, die bei der Ausführung eines Geschäftsprozesses erbracht wird, kann wiederum als (zusammengesetzter) Service in einer serviceorientierten Architektur angeboten werden. Die Ausprägungen der Qualitätsmerkmale des zusammengesetzten Services hängen dabei von den Qualitätsmerkmalen der ausgewählten Services und der Prozessdefinition ab. Um die Ausprägungen zu bestimmen, werden Aggregationsmuster formuliert. Dazu werden in Abschnitt 5.3 zunächst relevante Kontrollflussblöcke erarbeitet, die auf den Workflow-Patterns von van der Aalst basieren. In Ab-

schnitt 5.4 werden dann die Aggregationsmuster für die Berechnung der Qualität eines zusammengesetzten Services erarbeitet. Es werden dabei die Ausprägungen der Qualitätsmerkmale der einzelnen Services geeignet aggregiert. Abschnitt 5.5 beschreibt dann die Anwendung der Aggregationsmuster entlang der Hierarchie der Prozessdefinition.

Das Qualitätsmodell wird validiert durch eine praktischen Fallstudie, die im Rahmen einer Master-Arbeit von Harald Schubert am Hasso-Plattner-Institut für Softwaresystemtechnik, Potsdam, in Kooperation mit der SAP Systems Integration AG, Berlin, durchgeführt wurde. Dort wurden mehrere Qualitätsmerkmale aus der Anwendungsdomäne auf der Basis des Qualitätsmodells modelliert und zur Aggregation in einem Geschäftsprozess so angewandt, dass Aussagen über die Qualität des Geschäftsprozesses getroffen werden konnten (Abschnitt 5.6). Dieses Kapitel schließt mit einer Zusammenfassung (Abschnitt 5.7).

## 5.1 Grundlagen der Geschäftsprozesse

Die Prozessorientierung beschreibt eine ganzheitliche Sicht auf die Organisation einer Unternehmung [60], wobei Identifikation und kontrollierte Ausführung von *Geschäftsprozessen* eine zentrale Rolle spielen. Michael Hammer und James Champy definieren einen Geschäftsprozess wie folgt:

„We define a process as a collection of activities that takes one or more kinds of input and creates an output that is of value for the customer.“ [60]

Diese Definition soll auch für diese Arbeit gelten. Ein Geschäftsprozess besteht also aus einer Menge von *Aktivitäten*. Bei der Ausführung wird eine gewisse Eingabe entgegengenommen und eine Ausgabe erzeugt. Genauer betrachtet beinhaltet ein Geschäftsprozess auch *Ausführungsregeln* für die Menge der Aktivitäten. Somit stellen die Aktivitäten bei einer Ausführung *Prozessschritte* dar, deren Ausführungsreihenfolge durch einen *Kontrollfluss* definiert wird. Einen Geschäftsprozess ausführen bedeutet, die Aktivitäten gemäß den Ausführungsregeln auszuführen, um ein Ziel zu erreichen [61]. Dabei wird jeweils ein Mehrwert für einen Kunden erzielt<sup>1</sup>.

**Prozessdefinition** Die Menge der Aktivitäten und die Ausführungsregeln werden in einer *Prozessdefinition* formal beschrieben. Die grundlegenden Konstrukte für eine Prozessdefinition stellt eine *Prozessbeschreibungssprache* zur Verfügung. Die Erstellung einer Prozessdefinition wird auch *Prozessmodellierung* genannt. Man unterscheidet zwischen der Modellierungszeit und der Laufzeit: Während der *Modellierungszeit* (englisch Build-Time) werden alle Vorbereitungen für

---

<sup>1</sup>Hier sei darauf hingewiesen, dass ein für einen Kunden ausgeführter Geschäftsprozess als Leistung angesehen werden kann (siehe auch Definition 5). Dieses wird in Abschnitt 5.4 aufgegriffen, wo ein Geschäftsprozess als Service angesehen wird.

die Steuerung und Überwachung der Geschäftsprozesse getroffen und insbesondere die Prozessdefinitionen erstellt. Diese werden dann zur *Laufzeit* (englisch Run-Time) ausgeführt.

**Prozessinstanz** Die Ausführung einer Prozessdefinition erfolgt dadurch, dass auf Basis der Prozessdefinition eine *Prozessinstanz* erzeugt wird, die von einer *Prozesssteuerungskomponente* ausgeführt wird. Dabei werden die Aktivitäten entsprechend der Ausführungsregeln ausgeführt. Es können parallel mehrere Instanzen einer Prozessdefinition ausgeführt werden, die jeweils einen Geschäftsfall bearbeiten. In der Literatur ist für die Automatisierung eines Geschäftsprozesses auch der Begriff *Workflow* [55, 63, 69, 83, 164] gebräuchlich. Das IT-System zur Prozessinstanziierung, kontrollierten Ausführung und Überwachung der Instanzen wird dann als *Workflow-Management-System* [61, 83, 159, 166, 173, 188] bezeichnet.

**Modellierungsaspekte** Bei der Modellierung eines Geschäftsprozesses werden unterschiedliche Aspekte abgebildet. Der Verhaltensaspekt wird durch den bereits erwähnten *Kontrollfluss* beschrieben. Organisatorische Aspekte werden modelliert, indem die *Prozessbeteiligten* angegeben werden; eine Fragestellung dazu lautet: *Wer führt einen Prozessschritt aus?* Der Informationsaspekt wird durch die Angabe von *Datenfluss*-Konstrukten spezifiziert; hier lautet die Fragestellung: *Welche Daten werden für einen Prozessschritt benötigt und welche Daten werden durch einen Prozessschritt erzeugt?* Es existiert eine Vielzahl Prozessbeschreibungssprachen mit jeweils unterschiedlichen Schwerpunkten. Dabei unterscheiden sich die Sprachen vor allem in den grundlegenden Konzepten, wie der Kontrollfluss spezifiziert wird. Im Folgenden werden daher die wichtigsten Ansätze zur Spezifikation der Ausführungsreihenfolge klassifiziert.

### 5.1.1 Graphbasierte Prozessbeschreibungssprachen

Ein Graph eine Menge von Knoten, die durch Kanten verbunden sind [83]. Angewendet auf die Beschreibung von Geschäftsprozessen, repräsentieren die Knoten die Prozessschritte und die Kanten bestimmten die Ausführungsreihenfolge [159]. Die mögliche Menge aller Graphen wird von den jeweiligen Beschreibungssprachen, die auf der Graphentheorie basieren, eingeschränkt. Beispielsweise kann verlangt werden, dass gerichtete, azyklische Graphen verwendet werden. Weiterhin wird ein Knoten ohne eingehende Kanten als Startpunkt der Ausführung definiert und ein Knoten ohne ausgehende Kanten als Endpunkt. Von diesen Start- und Endpunkten darf es für einen Geschäftsprozess genau jeweils einen geben. Derartige Anforderungen werden beispielsweise für Workflow-Netze [159] getroffen. Zu den graphbasierten Prozessbeschreibungssprachen gehören Petri-Netze, (erweiterte) Ereignisgesteuerte Prozessketten und UML Aktivitätsdiagramme [83, 123, 159].

Eine Prozessdefinition als gerichteter, azyklischer Graph ist auf einer Hierarchie-Ebene angegeben. Derartige Prozessdefinitionen können besonders für kom-



plexe Geschäftsprozesse sehr groß und damit unübersichtlich werden. Um diese Komplexität zu beherrschen besteht die Möglichkeit, einzelne Abschnitte eines Prozesses zusammenzufassen und als Sub-Prozesse zu modellieren. Dadurch entsteht eine Hierarchie, so dass einzelne Knoten nicht nur atomare Aktivitäten, sondern können auch einen Sub-Prozess repräsentieren können.

Je nach Prozessbeschreibungssprache sind Regeln definiert, welche Prozessschritte wie zusammengefasst werden können. Dadurch ergeben sich zwei Vorteile: Erstens wird die Übersichtlichkeit erhöht und der Prozess ist leichter verständlich, da mehrere Prozessschritte zusammengefasst werden. Zweitens können einzelne Sub-Prozesse wiederverwendet werden, so dass komplexe Teilaufgaben einmal modelliert und in mehreren Prozessdefinitionen verwendet werden.

### 5.1.2 Blockstrukturierte Prozessbeschreibungssprachen

Die Hierarchisierung stellt ein zentrales Konzept der blockstrukturierten Prozessbeschreibungssprachen dar, in denen einzelne Aktivitäten zu einem Block zusammengefasst werden. Für diesen Block wird dann ein Kontrollfluss definiert. Diese Blöcke können dann mit anderen Blöcken oder Aktivitäten zu größeren Blöcken zusammengefasst werden. Dadurch wird eine hierarchische Struktur gebildet, welche die bereits erwähnten Vorzüge aufweist. Auf der obersten Ebene gibt es genau einen Block, mit dem die hierarchische Struktur beginnt. Jeder Prozessschritt kann eine einzelne Aktivität sein oder ein Block und somit ein weiterer (Teil-) Prozess.

Für jeden Block gibt es einen definierten Start und ein festgelegtes Ende. Die Ausführung bleibt innerhalb des Blocks, das heißt, die Blockgrenzen werden im zwischen Start und Ende nicht überschritten. Auch wenn mehrere Blöcke parallel ausgeführt werden, beeinflussen sich die Blöcke nicht gegenseitig. Für einen Block ist dann dessen Ausführungsregel, also der definierte Kontrollfluss, zu beachten. Auf der obersten Ebene eines Prozesses befindet sich genau ein Block. Wird ein Prozess instanziiert, so wird dieser Block entsprechend seiner Ausführungsregel ausgeführt. Die Prozessschritte werden entsprechend der Ausführungsregel ausgeführt und wenn es sich dabei um weitere Blöcke handelt, werden diese ebenfalls instanziiert.

Die strikte Trennung in Blöcke führt zu dem Nachteil, dass Prozessschritte in einem Block keine direkten Auswirkungen auf Prozessschritte in anderen Blöcken, also über Blockgrenzen hinweg, haben können. Diese Einschränkung wird in hybriden Ansätzen, wie sie im nächsten Abschnitt erwähnt werden, aufgehoben.

### 5.1.3 Weitere Ansätze

Im Kontext von Web Services hat sich mit der Business Process Execution Language for Web Services (WS-BPEL) [5] eine Spezifikation entwickelt, die wesentliche Konzepte sowohl aus der Graphentheorie als auch aus blockstrukturierten Ansätzen übernimmt [84]. Mit WS-BPEL können Geschäftsprozesse modelliert werden, die Web Services aufrufen und deren Ergebnisse entgegennehmen und auswer-

ten [32, 58, 100, 120]. Grundsätzlich ist in WS-BPEL ein blockstrukturierter Ansatz, bei dem je Block der Kontrollfluss zu spezifizieren ist und dessen Prozessschritte Interaktionen mit Services darstellen. Darüber hinaus gibt es in WS-BPEL auch die Möglichkeit, Kontrollfluss-Kanten zu definieren, die über die Grenzen eines Blocks hinausgehen können. Daher ist WS-BPEL ein hybrider Ansatz, der Blockstrukturen und Aspekte von Graphen beinhaltet. Dies resultiert aus der historischen Verschmelzung der beiden Vorgänger: der graphbasierten Web Services Flow Language (WSFL) [82] und dem blockstrukturierten XLANG [147].

Zur Visualisierung von Geschäftsprozessen wurde mit der Business Process Modeling Notation (BPMN) der OMG [107] eine grafische Darstellungsform spezifiziert. Es ist ebenfalls spezifiziert, wie eine WS-BPEL Prozessbeschreibung in der BPMN darzustellen ist [170]. Einen Geschäftsprozesses als XML-Dokument darzustellen, ist unübersichtlich und nicht intuitiv verständlich, daher wird mit BPMN insbesondere die Anschaulichkeit verbessert. Wie auch WS-BPEL ist BPMN im Grunde eine hybride Prozessbeschreibungssprache. Der in Abbildung 4.1 (Seite 72) dargestellte Beispiel-Prozess ist in BPMN formuliert. Diese Notation wird im weiteren Verlauf der Arbeit für die Visualisierung der Beispiele eingesetzt.

Weiterhin gibt es Beschreibungssprachen, die auf einer Algebra zu formalen Beschreibung von Prozessen basieren, wie beispielsweise dem  $\pi$ -Kalkül [101]. Vorteil derartiger Prozessbeschreibungssprachen ist, dass die Ausführung eines Prozesses eindeutig definiert ist, da die Semantik formal beschrieben ist [124].

## 5.2 Service-Auswahl

Bei der Modellierung eines Geschäftsprozesses in einer serviceorientierten Umgebung sind aus den verfügbaren, beim Broker veröffentlichten Services, geeignete Services auszuwählen [121]. In der Prozessdefinition kann dann jeweils beschrieben werden, wie die ausgewählten Services aufzurufen sind. Die Informationen zum Aufruf stammen aus der veröffentlichten Service-Beschreibung. Bei einer Prozessausführung werden die ausgewählten Services dann entsprechend aufgerufen, wenn die Prozesssteuerungskomponente an der betreffenden Stelle in der Prozessinstanz angekommen ist. In diesem Abschnitt wird beschrieben, welche Varianten es für die Auswahl der Services gibt, wobei insbesondere das Service-Modell und das Qualitätsmodell berücksichtigt wird.

**Service-Typ** Bei der Entwicklung des Service-Modells in Kapitel 3 wurde das Konzept der Service-Typen eingeführt. Ein Service-Typ fasst Services zusammen, die dieselbe Funktionalität und dieselbe Schnittstelle zur Verfügung stellen. Bei der Untersuchung des Modells wurde herausgearbeitet, dass insbesondere das Konzept der Service-Typen eine lose Kopplung erlaubt. Services desselben Typs sind relativ einfach auszutauschen, denn die Kommunikationsschnittstelle ist bei allen Services eines Typs identisch. Somit sind die Operationen und die Nachrichtentypen bekannt, wenn bereits einmal ein Service dieses Typs aufgerufen worden ist. Da

auch die Funktionalität von Services desselben Typs identisch ist, kann somit jeder Service eines Typs eingesetzt werden um die entsprechende Leistung zu erbringen.

Eine Prozessdefinition kann nun so erstellt werden, dass für Prozessschritte, die durch Service-Aufrufe realisiert werden sollen, der Service-Typ festgelegt wird. Dann kann bei einer Prozessausführung jeweils für den definierten Service-Typ dynamisch ein Service gewählt werden. Hier können dann die aktuell verfügbaren Services mit den besten Ausprägungen der Qualitätsmerkmale aufgerufen werden. Zusammengefasst wirkt sich das Konzept der Service-Typen des Service-Modells wie folgt auf die Modellierungsaspekte eines Prozesses aus:

- **Informationsaspekt:** Da alle Services eines Typs dieselbe Kommunikationsschnittstelle realisieren, kann der Datenfluss vollständig modelliert werden. Eingabe- und Ausgabeparameter sind durch die Nachrichtentypen spezifiziert.
- **Verhaltensaspekt:** Der Kontrollfluss kann ebenso modelliert werden und beschreibt, an welcher Stelle der Prozessausführung ein Service des festgelegten Typs auszurufen ist.
- **Organisationsaspekt:** Der organisatorische Aspekt wird nicht vollständig modelliert. Es bleibt noch offen, welcher Service die Ausführung des Prozessschrittes übernimmt, da lediglich der Service-Typ spezifiziert wird.

**Qualitätsmerkmale** Das Qualitätsmodell liefert eine konzeptionelle Grundlage um Services desselben Typs vergleichen zu können. Dadurch kann eine Rangfolge passender Services für einen Prozessschritt erstellt werden. Bei der Untersuchung des Qualitätsmodells in Kapitel 4 wurde erläutert, wie Anfragen an den Broker formuliert werden können, so dass der Broker den besten Service beziehungsweise eine Rangfolge ermitteln kann. Beispiele für derartige Anfragen wurden bereits vorgestellt.

Wenn ein Prozessschritt durch Aufruf eines Services ausgeführt werden soll, so ist bereits bei der Modellierung des Geschäftsprozesses für den jeweiligen Prozessschritt ein Service ermittelt worden. Es ist somit für die Prozessdefinition neben dem Service-Typ auch die Anfrage an einen entsprechenden Service Broker zu modellieren. Zu diesem Zweck seien die folgenden Varianten betrachtet, wie und wann Anfragen an einen Broker geschickt werden können:

1. **Service-Auswahl zur Modellierungszeit:** Wird die Anfrage zur Modellierungszeit ausgeführt, so kann direkt ein Service aus der Rangfolge bestimmt und in der Prozessdefinition festgelegt werden.
2. **Service-Auswahl während der Prozessinstanziierung:** Die Anfrage wird erst dann ausgeführt, wenn der Prozess gestartet werden soll, also während der Prozessinstanziierung. Dann werden alle aktuell verfügbaren Services berücksichtigt, eine Rangfolge erstellt und aus ein Service ausgewählt.

3. **Service-Auswahl zur Laufzeit:** Die Anfrage wird direkt vor der Ausführung des Prozessschrittes ausgeführt und der beste Service somit unmittelbar vor seinem Aufruf bestimmt.

In den folgenden Abschnitten werden diese Varianten genauer untersucht und der Bezug zum Qualitätsmodell aufgezeigt. Es wird gezeigt, wie die Service-Auswahl durch die konzeptionelle Grundlage der Qualitätsmerkmale unterstützt werden kann und die Auswahl verbessert wird.

### 5.2.1 Prozessbeispiel

Das Prozessbeispiel aus Abschnitt 4.1.2 soll im Folgenden verwendet werden. Er besteht auch hier aus den vier Prozessschritten *Auftragsannahme*, *Produktion* und parallel die Schritte *Abrechnung* und *Transport*. Abbildung 5.1 zeigt die Prozessdefinition in BPMN. Als Beschreibungen zu den jeweiligen Prozessschritten sind Ausprägungen für die in Kapitel 4 spezifizierten Qualitätsmerkmale Kosten ( $q_{cost}$ ) und Ausführungsdauer ( $q_{time}$ ) definiert. Es seien also die Metrik, Skala und Wertebereich wie beschrieben festgelegt.

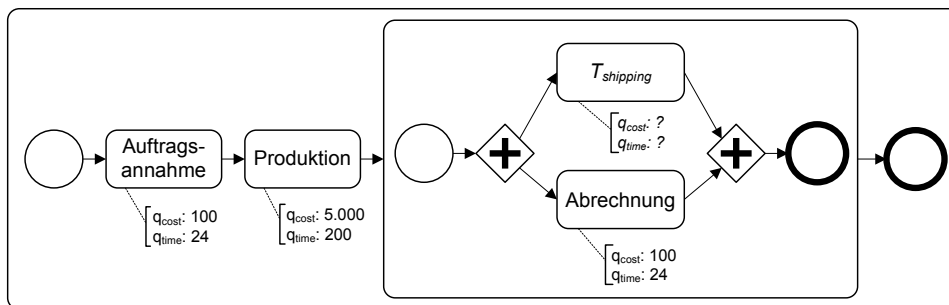


Abbildung 5.1: Prozessdefinition des Produktionsprozess, erweitert um Qualitätsmerkmale Kosten und Zeit sowie Service-Typ  $T_{shipping}$ , dargestellt in BPMN [107]

Anstelle des Prozessschrittes *Transport* aus dem ursprünglichen Beispiel (siehe Abbildung 4.1) ist in Abbildung 5.1 der Service-Typ  $T_{shipping}$  angegeben. Dadurch wird ausgedrückt, dass für diesen Prozessschritt ein Service gesucht wird, der den Service-Typ  $T_{shipping}$  implementiert. Hier soll einer der Services aus dem Transport-Beispiel aus Abschnitt 4.1.2 (Seite 69) gewählt werden; Kosten und Zeit der Services  $S_1$  bis  $S_6$  sind in Tabelle 4.2 dargestellt. Die tatsächlichen Werte für Kosten und Zeit sind erst dann bekannt, wenn ein Service ausgewählt ist.

Darüber hinaus fällt bei der Analyse des Gesamtprozesses auf, dass aggregierte Werte für Kosten und Zeit für den gesamten Prozess errechnet werden können. Für die Kosten des Gesamtprozesses sind dazu die Kosten der einzelnen Prozessschritte zu addieren. Demgegenüber ist die Zeit anders zu aggregieren, da die beiden letzten Prozessschritte parallelen ausgeführt werden. Die Ausführungsdauer des Blocks ist das Maximum der beiden Einzelwerte für  $q_{time}$ . Die Betrachtung des

Beispiels zeigt, dass es unterschiedliche Arten der Aggregation gibt, die abhängig vom Qualitätsmerkmal und vom Kontrollfluss sein können. Dies wird später aufgegriffen und untersucht (Abschnitt 5.4).

### 5.2.2 Service-Auswahl zur Modellierungszeit

Die Auswahl der Services kann bei dieser Variante als statisch angesehen werden, da zur Modellierungszeit bereits die Auswahl durchgeführt wird und in der Prozessdefinition festgelegt wird. Zur Laufzeit einer jeden Instanz dieser Prozessdefinition werden dann jeweils dieselben Services aufgerufen. Beispielsweise sei für den Produktionsprozess die Anfrage so formuliert, dass die Lieferzeit minimiert werden soll. Daher wird zur Modellierungszeit der Transport-Service  $S_4$  (Kosten: 320 €, Lieferzeit: 18 Stunden) gewählt, weil dieser die geringste Lieferzeit garantiert. Dies ist in Abbildung 5.2 dargestellt.

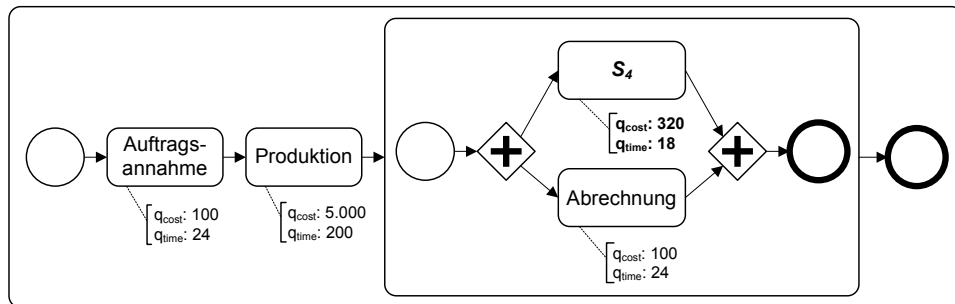


Abbildung 5.2: Produktionsprozess mit ausgewähltem Service  $S_4$

Das Qualitätsmodell liefert die konzeptionelle Grundlage für eine rationale Service-Auswahl und eine informierte Entscheidung, da die Services eines Typs vergleichbar sind. Durch die Spezifikation der Qualitätsmerkmale können Optimierungskriterien und Restriktionen in den Anfragen formuliert werden, so dass eine Rangfolge der Services ermittelt werden kann. Dadurch wird die Prozessmodellierung unterstützt.

Bei der Ausführung einer Instanz kann allerdings das Problem entstehen, dass einer der zur Modellierungszeit ausgewählten Service nicht verfügbar ist. Der Service-Aufruf schlägt somit fehl und die Ausführung des Geschäftsprozesses kann an dem entsprechenden Prozessschritt nicht weiter ausgeführt werden. Zur Laufzeit einer Prozessinstanz sei  $S_4$  nun nicht verfügbar. Der Service-Aufruf schlägt also fehl und der Transport kann nicht durchgeführt werden. Die Prozessinstanz kann nicht abgeschlossen werden und es ist ein menschlicher Eingriff nötig, um den Prozess weiterzuführen.

**Modellierung alternativer Services** Um das Problem eines ausgefallenen Services zu lösen, kann die Prozessdefinition erweitert werden: Es wird nicht nur ein

Service ( $S_4$ ) ausgewählt, sondern zusätzlich alternative Services angegeben (beispielsweise  $S_3$  mit Kosten: 150 €, Lieferzeit: 24 Stunden). Diese sind durch den Modellierer aus der Rangfolge zu entnehmen, die der Service Broker als Ergebnis der Anfrage ermittelt hat. Grundsätzlich zeigt sich hier ein Vorteil der Service-Orientierung: Das Prinzip der losen Kopplung erlaubt einen relativ einfachen Austausch von Services desselben Typs. Allerdings ist offensichtlich eine Prozesssteuerungskomponente notwendig, die einen fehlgeschlagenen Service-Aufruf registriert und entsprechend die modellierte Alternative ausführt. Ein ähnlicher Ansatz wird in [121] mit dem *Best-m-Approach* vorgeschlagen, der die besten  $m$  Services auswählt.

Damit gehen die Prozessmodellierung und die Ausführung von Prozessinstanzen über klassische Ansätze des Workflow-Managements hinaus. Voraussetzung ist neben der zusätzlichen Funktionalität der Prozesssteuerungskomponente, dass die Qualitätsmerkmale entsprechend des Qualitätsmodells aus Kapitel 4 modelliert werden. Erst dadurch ist es möglich, eine Rangfolge von Services zu ermitteln, so dass alternative Services in die Prozessdefinition aufgenommen werden können. Ohne eine Umsetzung des Qualitätsmodells müsste eine erneute Anfrage an einen Service Broker manuell ausgewertet werden und ein alternativer Service bestimmt werden. Dabei ist ebenfalls zu beachten, dass die bereits fehlgeschlagenen Services aus der Ergebnismenge eliminiert werden müssen. Während dieser Zeit ist der Geschäftsprozess an dieser Stelle unterbrochen. Dies kann die Prozessausführung erheblich verzögern.

**Dokumentation der Anfrage** Die Anfrage zur Service-Auswahl kann in der Prozessdefinition dokumentiert werden, so dass bei einer späteren Prozessoptimierung [60] dieselbe Anfrage erneut ausgeführt werden kann. Dann kann ein ursprünglich gewählter Service durch einen inzwischen verfügbaren, besseren Service ausgetauscht werden und somit eine verbesserte Prozessdefinition entstehen. Wird beispielsweise ein neuer Transport-Service  $S'_4$  angeboten, der 299 € kostet und ansonsten dieselben Ausprägungen der Qualitätsmerkmale aufweist. Bei einer Analyse des Produktionsprozesses durch einen Modellierer kann die Anfrage erneut ausgeführt werden und der neue Service  $S'_4$  in eine verbesserte Prozessdefinition anstelle von  $S_4$  (Kosten: 320 €) aufgenommen werden. Anschließend können neue Prozessinstanzen zu geringeren Kosten ausgeführt werden.

Die Möglichkeiten einer Geschäftsprozessoptimierung werden durch das Qualitätsmodell erheblich erweitert. Die ursprüngliche Service-Auswahl ist durch die dokumentierte Anfrage auch im Nachhinein rational nachvollziehbar. Darüber hinaus kann eine erneute Anwendung zu einem späteren Zeitpunkt verbesserte Ergebnisse erzielen. Zusätzlich wird die Optimierung vereinfacht, wenn die vorherigen Entscheidungen formal dokumentiert sind, da die Optimierung auf die existierenden Anfragen zurückgreifen kann.

**Individuelle Service Level Agreements** Die Variante der statischen Service-Auswahl birgt weiteres Optimierungspotenzial: Wird ein Service zur Modellierungszeit festgelegt und werden zukünftig viele Prozessinstanzen ausgeführt, so kann es sich lohnen, mit dem Provider des ausgewählten Services ein spezielles, individuelles Service Level Agreement zu vereinbaren. Dadurch ist es möglich, die spezifischen Anforderungen des Geschäftsprozesses zu berücksichtigen. Dieses Service Level Agreement kann von den veröffentlichten Ausprägungen der Qualitätsmerkmale abweichen.

Zum Beispiel kann in einem Ausführungspfad des Prozesses, der nicht zeitkritisch ist, mit dem Provider eine erlaubte Ausführungsdauer vereinbart werden, die über der öffentlich angebotenen Ausführungsdauer liegt. Dafür kann der Provider diesen Service im Gegenzug günstiger anbieten. Somit werden durch die Berücksichtigung der Qualitätsmerkmale Angebot und Nachfrage besser zusammengeführt. Im Beispiel sei der Service  $S_4$  gewählt, wie in Abbildung 5.2 dargestellt. Ferner sei festgelegt, dass als untere Grenze lediglich eine Lieferzeit von 20 Stunden für den Transport-Service relevant ist. Somit ist die Ausführungsdauer von  $S_4$  mit 18 Stunden geringer als eigentlich benötigt. In einem individuellen Service Level Agreement wird nun mit dem Provider von  $S_4$  eine Lieferzeit von 20 Stunden und Kosten von 300 € vereinbart. Somit können die Kosten um 20 € verringert werden. Diese Vereinbarung wird dann in der Prozessdefinition abgebildet und bedeutet, dass zukünftig jede Prozessinstanz zu reduzierten Kosten ausgeführt wird und je Prozessinstanz 20 € eingespart werden. Abbildung 5.3 zeigt die veränderte Prozessdefinition mit dem Prozessschritt  $S_4^{SLA}$ .

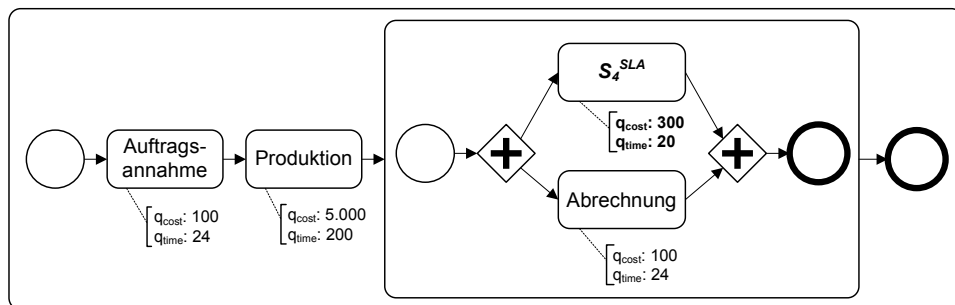


Abbildung 5.3: Produktionsprozess mit individuellem Service Level Agreement ( $S_4^{SLA}$ )

Derartige individuelle Vereinbarungen sind in heutigen Geschäftsbeziehungen durchaus üblich und somit auch für Geschäftsprozesse relevant. Durch das Qualitätsmodell besteht nun die Möglichkeit, dieses explizit in die Geschäftsprozessmodellierung einzubeziehen. Generell betrachtet sind ähnliche Vereinbarungen für alle Ausprägungen der modellierten Qualitätsmerkmale möglich. Es muss allerdings beachtet werden, ob sich der zusätzliche Aufwand für die Verhandlung der Service Level Agreements lohnt.

### 5.2.3 Service-Auswahl während Prozessinstanziierung

Die zweite Variante zur Service-Auswahl sieht vor, die Services zu Beginn der Prozessausführung auszuwählen. Dies geschieht während die Prozessinstanz erzeugt wird. Dazu werden die Anfragen an den Broker während der Prozessmodellierung formuliert, aber noch nicht durchgeführt. Erst wenn eine Prozessinstanz erzeugt wird, werden die spezifizierten Anfragen an einen Broker gesendet und die Ergebnisse ausgewertet. Somit besteht ein Teil der Prozessinstanziierung darin, die Services für die entsprechenden Prozessschritte auszuwählen.

**Vorteile** Die Variante erlaubt die Service-Auswahl aus einem aktuelleren Datenbestand des Brokers, da der Zeitpunkt der Service-Auswahl näher an der tatsächlichen Nutzung des Services liegt, als in der ersten Variante. Dadurch wird das in der vorigen Variante diskutierte Risiko, dass der gewählte Service nicht mehr verfügbar ist, verringert. Weiterhin können in der Zeit zwischen der Prozessmodellierung und der Prozessinstanziierung neue Services beim Broker veröffentlicht worden sein (wie  $S'_4$ ) oder Services abgemeldet werden. Die Rangfolge der Services kann sich also gegenüber der Modellierungszeit ändern. Diese Variante berücksichtigt alle beim Start der Prozessinstanz angebotenen Services.

Zusätzlich ist bei der Service-Auswahl während der Prozessinstanziierung vorteilhaft, dass bereits Informationen über die Anforderungen an diese Prozessinstanz verfügbar sein können. Beispielsweise können Termine (also Restriktionen der Zeit) oder Budget-Restriktionen feststehen, die eine genauere Kalkulation und Planung ermöglichen. Dieses kann dann in den Anfragen berücksichtigt werden. Hierzu ist allerdings eine Anfragesprache notwendig, die entsprechende Variablen unterstützt.

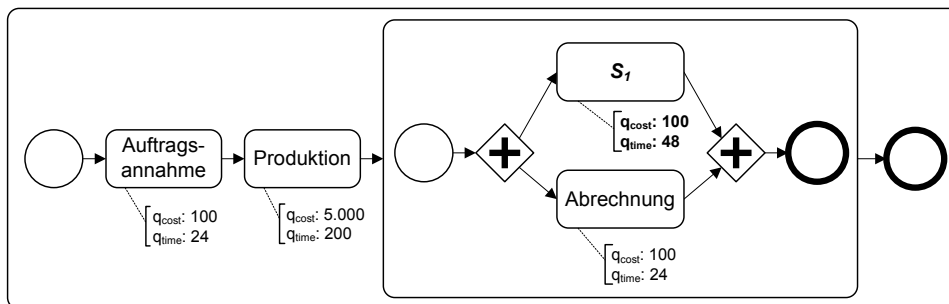


Abbildung 5.4: Prozessinstanz mit ausgewähltem Service  $S_1$

Für eine Instanz des Beispiel-Prozesses sei vor dem Start festgelegt, dass diese Instanz nicht zeitkritisch ist. Daher wird das Optimierungsziel der Anfrage für einen  $T_{shipping}$  Service von „minimale Lieferzeit“ auf „minimale Kosten“ geändert. Daher wird nun der Service  $S_1$  ausgewählt, der zwar 48 Stunden Lieferzeit benötigt, aber dafür nur 100 € kostet. Die Prozessdefinition für diese Instanz ist in Abbildung 5.4 dargestellt. Somit werden für diese Instanz 220 € eingespart.



**Nachteile** Allerdings bedeuten das Senden der Anfrage und die Auswertung der Ergebnisse zusätzlichen Aufwand. So ist die zusätzliche Zeit zu berücksichtigen, um die Anfrage an den Broker zu senden, die Antwort zu erhalten und das Resultat auszuwerten. Wie in Kapitel 4 aufgezeigt, kann durch entsprechende Anfragen die Auswahl automatisiert durchgeführt werden. Weitgehende Automatisierung ist ein primäres Ziel einer serviceorientierten Architektur. Daher kann davon ausgegangen werden, dass die Service-Auswahl unter Benutzung des Qualitätsmodells effizient ausgeführt wird.

Weiterhin ist zu berücksichtigen, dass im Vergleich zu der ersten Variante wesentlich mehr Anfragen an den Service Broker gesendet werden. Die erste Variante sieht nur jeweils eine Anfrage pro Service-Auswahl während der Modellierung sowie Anfragen zur Fehlerbehandlung oder Optimierung vor. Für fehlerfreie Prozessinstanzen sind keine weiteren Anfragen an den Broker zur Laufzeit nötig. In dieser Variante sind für jede Prozessinstanz Anfragen durchzuführen. Ist eine Anfrage an einen Service Broker beispielsweise mit zusätzlichen Kosten verbunden, so fallen diese Kosten für jede Prozessinstanz an.

Ein weiterer Nachteil dieser Variante ist, dass die Ausprägungen der Qualitätsmerkmale für die Services erst bei der Prozessinstanziierung feststehen. Sie sind daher nicht zur Modellierungszeit bekannt. Für den Produktionsprozess ist somit nicht bekannt, wie lange der Transport dauert. Dadurch ist unklar, wie lange es dauert, bis das Produkt beim Kunden ist. Zur Modellierungszeit können allerdings Services ermittelt werden, die als Grundlage für eine Schätzung dienen können, wie lange es dauert, bis das Produkt beim Kunden ist. Somit können auch für einen neu modellierten Prozess Ausprägungen für die Qualitätsmerkmale angegeben werden.

**Zusammenfassung** Durch die Auswahl der Services erst zur Prozessinstanziierung ist eine individuelle Anpassung für jede einzelne Prozessinstanz möglich. Werden die Anfragen entsprechend spezifiziert und werden bei der Instanziierung die notwendigen Informationen erfasst und zur Verfügung gestellt, so ergibt sich sogar die Möglichkeit zur individuellen Prozessoptimierung jeder Instanz. Diese Form der Prozessmodellierung geht über die bisherigen Ansätze der Geschäftsprozessmodellierung weit hinaus.

Die Vorteile der vorherigen Variante aus Abschnitt 5.2.2 gelten teilweise auch hier: Die Möglichkeiten zur Fehlerbehandlung durch vordefinierte, alternative Services bestehen ebenfalls. Individuelle Service Level Agreements können genutzt werden, wenn sie automatisiert ausgehandelt werden können. Allerdings entsteht hier erneut zusätzlicher Aufwand, den es zu berücksichtigen gilt.

### 5.2.4 Service-Auswahl zur Laufzeit

Bei der Service-Auswahl zur Laufzeit wird die Anfrage direkt vor dem Start des Prozessschrittes an den Broker gesendet. Die Anfrage ist dabei während der Modellierung spezifiziert worden, aber noch nicht an den Broker gesendet. Somit wird die Anfrage nur dann ausgeführt, wenn der Prozessschritt tatsächlich ausgeführt

wird. Liegt ein Schritt beispielsweise auf einem Pfad, der in der betrachteten Instanz nicht betreten wird, wird für diesen Prozessschritt keine Service-Auswahl durchgeführt.

**Vorteile** Die dritte Variante stellt die flexibelste Variante der Service-Auswahl dar, da der Service unmittelbar vor seinem Aufruf ausgewählt wird und somit auf den aktuellsten Datenbestand über verfügbare Services zugegriffen wird. Somit wird der Vorteil der zweiten Variante noch ausgebaut, indem die Fehlerhäufigkeit durch die aktuelleren Ergebnisse potentiell vermindert wird.

Ebenso wie in der vorherigen Variante, kann hier eine individuelle Anpassung der Anfragen und somit der Service-Auswahl vorgenommen werden. Der Vorteil kann bei dieser Variante ausgebaut werden, da die bisherige Ausführung der Prozessinstanz bei der Anfrage berücksichtigt werden kann.

Angenommen die normale Prozessausführung verlaufe wie in Abbildung 5.4 angegeben, also mit möglichst geringen Kosten für den Transport (100 €, 48 Stunden). Während der Ausführung einer Prozessinstanz sei nun die Aktivität Produktion verzögert und dauert 220 Stunden anstelle von 200 Stunden. Dadurch droht eine Verletzung eines vereinbarten Termins mit dem Kunden um 20 Stunden. Es kann nun bei der Anfrage bezüglich des  $T_{shipping}$  Services nach einem Service gesucht werden, der die verlorenen 20 Stunden wieder aufholt. Eine entsprechende Anfrage mit der Zeitrestriktion kann dann wie folgt aussehen:

$$R_5 = \{S \mid \lambda_S(q_{time}) \leq 28\}$$

Aus der Ergebnismenge  $R_5$  wird nun der Service mit den geringsten Kosten gewählt. Dies ist der Service  $S_3$  (Kosten: 150 €; Lieferzeit 24 Stunden). Die Instanz ruft nun  $S_3$  auf und der vereinbarte Termin kann eingehalten werden. Die entsprechende Prozessdefinition ist in Abbildung 5.5 dargestellt.

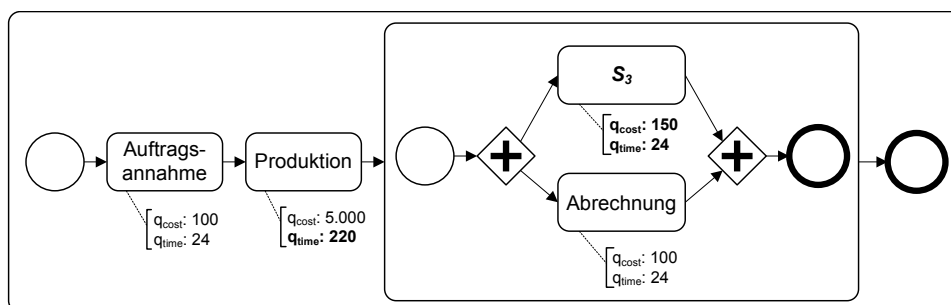


Abbildung 5.5: Prozessinstanz mit verzögertem Prozessschritt Produktion und ausgewähltem Service  $S_3$

**Nachteile** Die bereits im vorherigen Abschnitt diskutierten Nachteile des zusätzlichen Zeitaufwandes und der häufigeren Broker-Anfragen gelten bei dieser Variante ebenso. Hinzu kommt, dass die Anfragen nicht alle beim Start der Instanz

ausgeführt werden, sondern während der Ausführung, jeweils direkt vor einem Prozessschritt.

Ein spezieller Nachteil dieser Variante ist der zusätzliche Aufwand der Broker-Anfrage, der nicht die Instanziierung, sondern die Prozessausführung verlangsamten kann. Da für jeden relevanten Prozessschritt die entsprechende Service-Auswahl unmittelbar vor seinem Aufruf durchgeführt wird, wird die Prozessausführung an den Stellen der Service-Auswahl unterbrochen. Es ist allerdings zu beachten, dass durch den möglichst hohen Automatisierungsgrad in einer serviceorientierten Architektur die Auswahl relativ schnell getroffen werden kann. Dies gilt insbesondere dann, wenn das Qualitätsmodell umgesetzt wird und die Anfrage mit einer eindeutigen Rangfolge beantwortet werden kann.

Ebenso sind auch in dieser Variante die Qualitätseigenschaften einer Prozessinstanz erst während beziehungsweise nach deren Ausführung bekannt. Die Qualitätseigenschaften des Prozesses können also während der Modellierung lediglich geschätzt werden.

**Zusammenfassung** Diese Variante erlaubt eine sehr späte Auswahl der Services und führt zudem nur für tatsächlich auszuführende Prozessschritte eine Service-Auswahl durch. Dadurch wird ein Vorteil gegenüber der Service-Auswahl während der Prozessinstanziierung erzielt.

Durch das Beispiel eines verzögerten Prozessschrittes wurde insbesondere die Möglichkeit illustriert, den bisherigen Verlauf der Prozessinstanz zu berücksichtigen. Anfragen können für die jeweilige Instanz individuell angepasst werden. Die konzeptionelle Grundlage für Qualitätsmerkmale erlaubt es, unvorhergesehene Abweichungen von Qualitätsmerkmalen in folgenden Prozessschritten zu berücksichtigen, in dem die Anfragen entsprechend angepasst werden. Mächtige Techniken zur Prozessüberwachung und Prozessoptimierung werden möglich, so dass die Flexibilität und Anpassungsfähigkeit der Geschäftsprozesse erhöht wird.

Durch den zusätzlichen Aufwand der Auswahl vor jedem Prozessschritt und die damit verbundene Verzögerung ist diese Variante für besonders zeitkritische und schnell auszuführende Geschäftsprozesse unter Umständen nicht effizient anwendbar. Besonders im Kontext von lang laufenden Geschäftsprozessen, die sich über mehrere Tage oder Wochen erstrecken können, kann es dennoch sinnvoll sein, erst zum spätest möglichen Zeitpunkt die Anfrage an den Service Broker zu senden, um so die aktuellsten Services zu berücksichtigen.

### 5.2.5 Zusammenfassung der Varianten

Wie durch die Variante zur Service-Auswahl gezeigt wurde, ergeben sich bei der Modellierung eines Geschäftsprozesses insbesondere dann Vorteile, wenn nicht ein Service für einen Prozessschritt festgelegt wird, sondern lediglich der Service-Typ und eine entsprechende Anfrage.

An dieser Stelle zeigt sich ein besonderer Vorteil einer serviceorientierten Architektur gegenüber klassischen Geschäftsprozessen: Zur Laufzeit einer Prozess-

instanz können flexibel die aktuell besten Services aus dem Angebot ausgewählt werden und die Besonderheiten der Instanz berücksichtigt werden. Andererseits können während der Modellierung des Geschäftsprozesses beispielsweise besonders gute Konditionen mit Providern ausgehandelt werden, so dass ein optimierter Geschäftsprozess mit explizit ausgewählten Services entsteht. Folgende Vorteile in einer serviceorientierten Architektur wurden herausgearbeitet:

- Die Berücksichtigung möglichst aktueller Informationen über die Qualitätsmerkmale der angebotenen Services wird möglich und erlaubt die Auswahl des besten verfügbaren Services.
- Die Angabe von alternativen Services (entsprechend der Rangfolge) für unvorhergesehene Fehlerfälle ist möglich und erlaubt jeweils eine schnelle Reaktion.
- Die Verwendung des Qualitätsmodells offenbart erweiterte Möglichkeiten zur Optimierung der Geschäftsprozesse:
  - Individuelle SLA's erlauben eine Optimierung speziell für eine Prozessdefinition. Diese Optimierung kann dann für alle Instanzen der Prozessdefinition genutzt werden.
  - Die Berücksichtigung individueller Daten und Restriktionen, die beim Start einer Prozessinstanz bekannt sind, ermöglicht eine individuelle Prozessoptimierung.
  - Die aktuelle Prozessinstanz und ihr bisheriger Verlauf können bei der Service-Auswahl berücksichtigt werden, so dass auf unvorhergesehene Ausnahmen und Abweichungen vom geplanten Verhalten dynamisch und flexibel reagiert werden kann.

Weiterhin lassen sich die vorgestellten Varianten auch kombinieren, so dass einige Services statisch bei der Modellierung vorgegeben werden, weitere bei der Instanziierung ausgewählt werden und einzelne Services direkt vor ihrer Benutzung ausgewählt werden. Letzteres bietet sich beispielsweise für Ausführungspfade an, die statistisch gesehen nur selten betreten werden. Dann ist eine Service-Auswahl hier nur selten durchzuführen. Zusammenfassend lässt sich also festhalten, dass das Qualitätsmodell einen erhöhten Flexibilitätsgrad bei der Ausführung von Geschäftsprozessen bei gleichzeitig erhöhtem Grad der Automatisierung ermöglicht.

**Auswirkung auf zusammengesetzte Services** Im weiteren Verlauf der Arbeit soll berücksichtigt werden, dass ein modellierter Geschäftsprozess als (zusammengesetzter) Service angeboten werden kann. Wie auch für jeden anderen Service, soll für den zusammengesetzten Service ebenfalls die Ausprägung der relevanten Qualitätsmerkmale angegeben werden. Es ist bereits angesprochen worden, dass die Qualität der im Prozess aufgerufenen Services die Qualität des Prozesses beeinflusst. Es wurde im Beispiel gezeigt, dass die Lieferzeit des Transport-Service die Gesamtdauer des Produktionsprozesses beeinflusst.

Grundsätzlich ist dies unabhängig von der verwendeten Variante der Service-Auswahl. Es ist bereits erwähnt worden, dass die Ausprägungen der Qualitätsmerkmale einzelner Prozessschritte je nach Variante noch nicht feststehen, wenn der Prozess modelliert worden ist. Hierzu wurde vorgeschlagen, die Anfragen bereits zur Modellierungszeit durchzuführen, um dann eine Abschätzung der Eigenschaften durchführen zu können. Daher gelte im Folgenden die Annahme, dass der Prozess vollständig modelliert sei und Services ausgewählt sind beziehungsweise deren Ausprägungen der Qualitätsmerkmale durch eine Schätzung festgelegt sind.

### 5.3 Kontrollflussblöcke

In diesem Abschnitt werden Kontrollflussblöcken spezifiziert, welche die Basis für die Aggregationsmuster bilden. Dazu werden zunächst Workflow-Patterns [156] eingeführt, die allgemeine Kontrollflusskonstrukte beschreiben. Anschließend werden relevante Workflow-Patterns ausgewählt und zu Kontrollflussblöcken zusammengeführt [72]. Auf Basis der Blöcke werden Geschäftsprozesse hinsichtlich ihrer Qualitätsmerkmale analysiert und es können Aussagen über aggregierte Qualitätsmerkmale eines Geschäftsprozesses zu treffen, der dann als zusammengesetzter Service angeboten werden kann.

#### 5.3.1 Workflow-Patterns

Um die unterschiedlichen Prozessbeschreibungssprachen existierender Workflow-Management-Systeme vergleichen zu können, wurden unter anderem von Wil van der Aalst die Workflow-Patterns in [156] definiert. Da Workflow-Management-Systeme und ihre Sprachen oft auf sehr unterschiedlichen Konzepten basieren (siehe auch Abschnitt 5.1), war ein direkter Vergleich der Systeme nicht möglich und die Unterschiede in ihrer Ausdrucksmächtigkeit waren nur schwer erkennbar. Beispielsweise existieren für Kontrollflusskonstrukte des einen Systems in einem anderen System unterschiedliche Varianten, um dieselbe Ausführung zu erreichen.

Mit allgemein gültigen Workflow-Patterns ist es möglich, sowohl Workflow-Management-Systeme [158] als auch Prozessmodellierungssprachen zu analysieren und zu vergleichen. Darüber hinaus existieren Umsetzungen der Workflow-Patterns beispielsweise in die Prozessbeschreibungssprache YAWL (Yet Another Workflow Language) [157], eine Formalisierung basierend auf dem  $\pi$ -Kalkül [125] und eine Visualisierungen durch BPMN [169].

Für den Kontrollfluss sind insgesamt 20 grundlegende Muster als *Workflow-Patterns* definiert, die in sechs Kategorien eingeteilt sind und hier kurz vorgestellt werden sollen. Es werden hierbei die in der englischsprachigen Literatur gebräuchlichen Namen der Muster und der Kategorien verwendet.

### Basic Control Patterns

Das Kontrollflussmuster *Sequenz* führt die Aktivitäten nacheinander aus. Ein *Parallel Split* spaltet die Ausführung in mehrere parallele Pfade auf und ein *Exclusive Choice* wählt aus mehreren alternativen Pfaden einen Pfad aus. Für diese beiden Muster gibt es jeweils die passenden Muster, um die Ausführungen wieder zusammenzuführen: Die *Synchronization* führt zwei nebenläufige Pfade zusammen und *Simple Merge* vereinigt zwei alternative Ausführungspfade.

### Advanced Branching and Synchronization Patterns

Neben den einfachen Basis-Mustern werden weitere grundlegende Muster für den Kontrollfluss spezifiziert. *Multiple Choice* stellt eine Erweiterung des Exclusive Choice Musters dar, da aus mehreren Pfaden mehrere ausgewählt werden. Zum Zusammenführen der Verzweigungen können die folgenden Muster gewählt werden:

- Ein *Synchronizing Merge* führt mehrere Ausführungspfade zusammen und synchronisiert diese, wenn mehr als ein Pfad beschritten wurde. Die nachfolgende Aktivität wird erst ausgeführt, wenn alle vorherigen Pfade, die ausgeführt wurden, vollständig beendet sind. Die Ausführung wird also synchronisiert.
- Ein *Multiple Merge* führt mehrere Pfade zusammen, ohne diese zu synchronisieren. Für jeden Pfad, der beschritten wurde, wird die nachfolgende Aktivität ausgeführt.
- Ein *Discriminator* führt ebenfalls mehrere Pfade zusammen, ohne diese zu synchronisieren. Allerdings wird hierbei die nachfolgende Aktivität nur einmal ausgeführt. Dieses geschieht, sobald der erste Pfad an diesem Kontrollfluss-Pattern angekommen ist.
- Ein *N-out-of-M Join* führt mehrere ( $M$ ) Ausführungspfade zusammen und synchronisiert die weitere Ausführung. Allerdings wird nach der Beendigung von  $N$  Pfaden bereits die nachfolgende Aktivität genau einmal gestartet.

### Structural Patterns

Die strukturellen Muster umfassen *Arbitrary Cycles*, also willkürliche Schleifen ohne Einschränkung der Struktur und *Implicit Termination*, also die implizite Beendigung eines (Sub-) Prozesses, wenn keine weitere Aktivität auszuführen ist.

### Patterns Involving Multiple Instances

Diese Muster beschreiben Kontrollflüsse mit mehreren Instanzen (englisch Multiple Instances, MI). Dies bedeutet, dass eine Aktivität mehrfach ausgeführt wird. Es

werden folgende Ausprägungen unterschieden:

- *MI without synchronization*: Von der betrachteten Aktivität werden mehrere Instanzen erzeugt, die anschließend nicht synchronisiert werden.
- *MI with a priori known design time knowledge*: Bereits zur Modellierungszeit ist bekannt, wie viele Instanzen von der betrachteten Aktivität erzeugt werden. Diese werden nach der Ausführung synchronisiert.
- *MI with a priori known runtime knowledge*: Zur Laufzeit des Prozesses ist zu einem bestimmten Zeitpunkt bekannt, wie viele Instanzen der Aktivität erzeugt werden. Dieses ähnelt einer aus Programmiersprachen bekannten `for`-Schleife, allerdings werden die Aktivitäten parallel ausgeführt.
- *MI with no a priori runtime knowledge*: Es werden mehrere Instanzen der betrachteten Aktivität erzeugt, ohne dass die genaue Anzahl bekannt ist. Dieses ähnelt einer aus Programmiersprachen bekannten `while`-Schleife, allerdings werden die Aktivitäten parallel ausgeführt.

### State-based patterns

Diese Muster betrachten einen bestimmten Zustand der Prozessinstanz während der Ausführung. *Deferred Choice* beschreibt einen Zustand, in dem zwei Aktivitäten möglich wären – ähnlich dem oben beschriebenen *Exclusive Choice*. Allerdings ist die Entscheidung, welcher der Aktivitäten ausgeführt wird, nicht durch prozessrelevante Daten der vergangenen Aktivitäten bestimmt oder durch eine Bedingung definiert, sondern wird von außen getroffen, beispielsweise durch einen menschlichen Benutzer oder andere Ereignisse des Umfeldes. Für die Prozesssteuerung bedeutet dieses auch, dass die nicht gewählte Alternative nicht ausgeführt werden darf.

Das Muster *Interleaved Parallel Routing* beschreibt die Ausführung von mehreren Aktivitäten in einer zufälligen Reihenfolge, aber nicht parallel. Die Reihenfolge wird zur Laufzeit bestimmt. Dies impliziert, dass alle Sequenzen der Aktivitäten für die Prozessausführung zulässig sind. Es ähnelt somit einer Sequenz, bei dem alle Permutationen der Aktivitätsreihenfolge möglich sind. Ein *Milestone*-Muster aktiviert eine Aktivität nur in einem bestimmten Zustand des Prozesses. Ob diese Aktivität ausgeführt wird, hängt somit auch von einem anderen Teil des Prozesses ab.

### Cancellation Patterns

Die beiden Cancellation Muster brechen die Ausführung einer Prozessinstanz bei einem bestimmten Ereignis oder einer Interaktion mit einem menschlichen Benutzer ab. Dies gilt entweder für eine Aktivität (*Cancel Activity*) oder für die gesamte Prozessinstanz (*Cancel Case*). Diese Muster beschreiben einen ungewollten Prozessabbruch, der nicht den Regelfall darstellt. Dennoch wird explizit modelliert, wo

und unter welchen Bedingungen eine Prozessinstanz beziehungsweise eine Aktivität abgebrochen werden kann.

### 5.3.2 Bildung von Kontrollflussblöcken

Die Aggregation von Qualitätsmerkmalen ist abhängig vom Kontrollfluss, wie im Beispiel des Produktionsprozesses in Abschnitt 5.2.1 bereits erläutert wurde. Daher werden in diesem Abschnitt Kontrollflussblöcke gebildet. Zur Bildung der Blöcke wird ein auf Workflow-Patterns basierender, blockstrukturierter Ansatz gewählt.

Die Kontrollflussblöcke dienen der Untersuchung des Qualitätsmodells. Unter Benutzung der Kontrollflussblöcke können Aggregationsmuster für Qualitätsmerkmale formuliert und anhand von Beispielen illustriert werden. Durch sukzessives Aggregieren der Werte können dann Aussagen über Qualitätsmerkmale eines Geschäftsprozesses getroffen werden. Weiterhin sei auf zwei ähnliche Ansätze verwiesen: Das in [31] vorgestellte Verfahren zur Aggregation benutzt eine graphbasierten Prozessbeschreibungssprache, die auf gerichteten, azyklischen Graphen arbeitet. Aggregationen von vier speziellen Qualitätsmerkmalen werden in [72] für ähnliche Blöcke definiert. Eine detaillierte Betrachtung dieser verwandten Arbeiten findet in Abschnitt 6.5 statt.

Um eine gültige Prozessdefinition zu erstellen, sind einige der Workflow-Patterns (insbesondere *Splits* und *Merges*) geeignet zu kombinieren. Es sind nicht alle Kombinationen zulässig: So kann beispielsweise die Ausführung nach einem *Parallel Split* nicht durch einen *Simple Merge* zusammengeführt werden. Die Gültigkeit einer Prozessdefinition wird unter anderem in [159] beschrieben. Außerdem werden in einem ersten Schritt zur Bildung von Kontrollflussblöcken die folgenden Workflow-Pattern zusammengefasst:

- Die Workflow-Pattern *Synchronization* und *Synchronizing Merge* werden als äquivalent angesehen: Wenn vorher ein *Parallel Split* ausgeführt wurde, synchronisieren Sie die Ausführung. Eine *Synchronization* ist ein Spezialfall des *Synchronizing Merge* mit nur zwei eingehenden Ausführungspfaden.
- Die Workflow-Pattern *Simple Merge* und *Synchronizing Merge* werden als äquivalent angesehen: Wenn vorher ein *Exclusive Choice* ausgeführt wurde, dienen sie als Zusammenführung der möglichen Ausführungspfade. Eine *Simple Merge* ist ein Spezialfall des *Synchronizing Merge* mit nur zwei eingehenden Ausführungspfaden.
- Die Workflow-Pattern *Discriminator* und *N-out-of-M Join* werden als äquivalent angesehen: Der *Discriminator* ist ein Spezialfall eines *N-out-of-M Join*, mit  $N = 1$ .
- Die Workflow-Pattern *Deferred Choice* und *Exclusive Choice* werden als äquivalent angesehen: Da jeweils genau einer der folgenden Ausführungspfade ausgeführt wird, sind die Muster *Deferred Choice* und *Exclusive Choice*



*ce* aus Sicht der Ermittlung aggregierter Qualitätsmerkmale als äquivalent anzusehen.

Ein Kontrollflussblock definiert die Ausführung der Aktivitäten in dem Block. Die Prozessschritte in einem Kontrollflussblock können eine Aktivität oder ein weiterer Kontrollflussblock sein, wie bei blockstrukturierten Ansätzen üblich (siehe auch Abschnitt 3.4.3). Eine Aktivität kann durch einen Service-Aufruf realisiert werden. Außerdem sei für jeden Block das Muster *Implicit Termination* definiert, also die implizite Beendigung des Blocks, falls kein weiterer Prozessschritt mehr auszuführen ist. Im Folgenden seien diese Blöcke definiert und in BPMN illustriert.

Zum besseren Verständnis sind die Prozessschritte in den folgenden Beispielen als Aktivität  $A_n$  angegeben. Diese Aktivität kann auch ein Service-Aufruf oder ein weiterer Kontrollflussblock sein. Der Beispiel-Kontrollflussblock ist immer mit  $B$  bezeichnet. Zur Identifizierung der unterschiedlichen Kontrollflussblöcke werden hier Namen eingeführt, die in der Überschrift vor den verwendeten Workflow-Pattern angegeben sind.

**Sequenz: Sequence oder Interleaved Parallel Routing**

Eine *Sequenz* ist die Ausführung der Prozessschritte nacheinander. Das Beispiel in Abbildung 5.6(a) zeigt eine Sequenz, in der zuerst  $A_1$ , dann  $A_2$  und dann  $A_3$  ausgeführt werden. Ein Prozessschritt kann erst dann starten, wenn jeweils der vorherige Prozessschritt beendet ist.

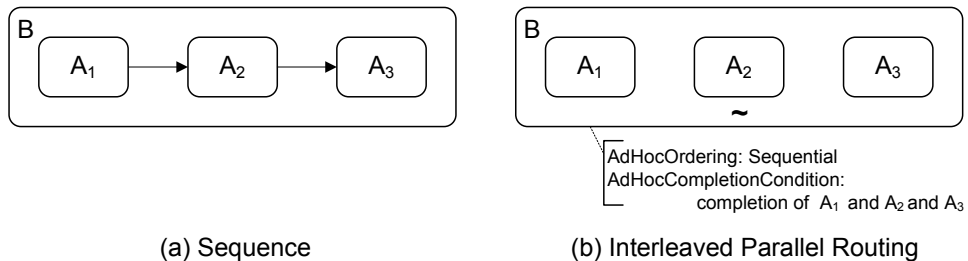
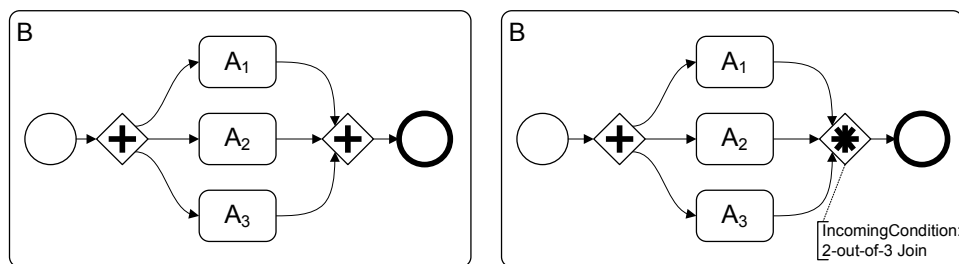


Abbildung 5.6: Kontrollflussblöcke Sequenz

Abbildung 5.6(b) definiert für die Schritte  $A_1$ ,  $A_2$  und  $A_3$  das Workflow-Pattern *Interleaved Parallel Routing*, dargestellt als BPMN Ad-Hoc-Teilprozess, der so definiert sei, dass die Ausführung sequenziell erfolgt und alle Aktivitäten ausgeführt werden müssen. Dies ist in Abbildung 5.6(b) durch die BPMN-Attribute *AdHocOrdering* und *AdHocCompletionCondition* ausgedrückt. Die Reihenfolge der Aktivitäten wird erst zur Laufzeit festgelegt. Für die Analyse der Aggregation ist das Muster *Interleaved Parallel Routing* als äquivalent anzusehen. Da die Qualitätsmerkmale unabhängig von der Reihenfolge sind, ist für die aggregierten Qualitätsmerkmale des Blocks die Reihenfolge der Prozessschritte ebenfalls nicht relevant.

### AND: Parallel-Split und Synchronizing Merge

Der Kontrollflussblock *AND* beginnt mit einem Workflow-Pattern *Parallel Split*. Es werden also alle nachfolgenden Prozessschritte gleichzeitig aktiviert. Das Beispiel in Abbildung 5.7(a) zeigt dafür einen Parallel-Split in BPMN als Gateway mit einem Kreuz. Die Prozessschritte  $A_1$ ,  $A_2$  und  $A_3$  werden parallel aktiviert. Die parallelen Ausführungspfade werden am Ende durch ein *Synchronizing Merge* zusammengeführt, ebenfalls dargestellt in der BPMN. Wie zu Beginn des Abschnitts erwähnt, ist das zusammenführende Muster ein *Synchronization*, wenn es sich um zwei parallele Ausführungspfade handelt, sonst ein *Synchronizing Merge*.



(a) Parallel-Split und Synchronizing Merge (b) Parallel-Split und N-out-of-M-Join

Abbildung 5.7: Kontrollflussblöcke mit Parallel-Split (AND und AndNM)

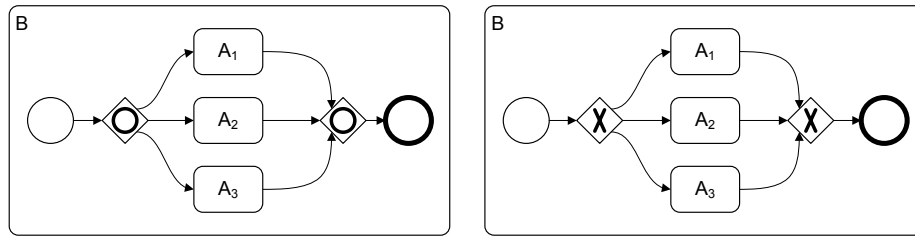
### AndNM: Parallel-Split und N-out-of-M-Join

Der Kontrollflussblock *AndNM* startet ebenso wie *AND*-Block mit einem Workflow-Pattern *Parallel Split*. Die Ausführung wird dann zusammengeführt durch einen *N-out-of-M-Join*. Wie oben erwähnt, ist ein *Discriminator* eine Spezialisierung des *N-out-of-M-Joins* mit  $N = 1$ . Das Beispiel in Abbildung 5.7(b) zeigt einen derartigen Block. Die Prozessschritte  $A_1$ ,  $A_2$  und  $A_3$  werden parallel aktiviert und die Ausführungspfade werden zusammengeführt mit einem *Complex Gateway*, bei dem das `IncomingCondition`-Attribut auf „2-out-of-3-Join“ gesetzt ist. Sobald zwei Ausführungspfade eintreffen kann die Ausführung fortgesetzt werden; der dritte eintreffende Ausführungspfad wird ignoriert.

### OR: Multiple Choice und Synchronizing Merge

Wie im Workflow-Pattern *Multiple Choice* definiert, werden mehrere der folgenden Prozessschritte gleichzeitig aktiviert. Das Beispiel in Abbildung 5.8(a) zeigt dafür einen *Or-Split* in der BPMN als Gateway mit einem Kreis. Die mindestens einer der Prozessschritte  $A_1$ ,  $A_2$  und  $A_3$  wird aktiviert. Die parallelen Ausführungspfade werden am Ende durch ein *Synchronizing Merge* zusammengeführt, dargestellt als *Or-Join* in BPMN.

In der Literatur werden ebenfalls Kontrollflussstrukturen definiert, die neben dem *And-Split* vor einem *N-out-of-M-Join* einen *OR-Split* vorsehen [72]. Hier-



(a) Multiple Choice und Synchronizing Merge (b) Exclusive Choice und Synchronizing Merge

Abbildung 5.8: Kontrollflussblöcke mit Multiple und Exclusive Choice (OR und XOR)

bei ergibt sich allerdings eine zusätzliche Problematik. Wenn nach einem OR-Split weniger als  $N$  Ausführungspfade betreten werden, ist nicht eindeutig definiert, wie die  $N$  eingehenden Kontrollflüsse zustande kommen sollen. Werden die nicht betretenen Pfade für die  $N$  benötigten berücksichtigt, im Sinne der Dead-Path-Elimination [83], so kann bei  $N$  nicht betretenen Pfaden der N-out-of-M-Join direkt nach dem OR-Split den Kontrollfluss weiterführen müsste.

### XOR: Exclusive Choice and Synchronizing Merge

Eine XOR-Block wählt aus alternativen Pfaden genau eine aus. Diese Auswahl kann entweder durch einen Kontrollfluss wie im Workflow-Pattern *Exclusive Choice* definiert erfolgen oder durch einen Kontrollfluss wie im Workflow-Pattern *Deferred Choice*. Wie zu Beginn des Abschnitts erwähnt, sind die beiden Workflow-Pattern für die Aggregation der Qualitätsmerkmale als äquivalent anzusehen. Das Beispiel in Abbildung 5.8(b) zeigt einen XOR-Block mit einem XOR-Split in BPMN, in dem genau einer der Prozessschritte  $A_1$ ,  $A_2$  oder  $A_3$  ausgeführt wird. Die Ausführungspfade werden durch einen XOR-Join zusammengeführt.

### Loop: Multiple Merge und Exclusive Choice

Das Workflow-Pattern *Arbitrary Cycles* erlaubt willkürliche Schleifen. Hierbei besteht eine Schleife vor allem aus einem Rücksprung zu einer bereits ausgeführten Aktivität. Für die Kontrollflussblöcke werden derartige Rücksprünge wie folgt eingeschränkt: Eine Schleife wird innerhalb eines Kontrollflussblocks definiert. Der Prozessschritt innerhalb der Schleife wird dann solange immer wieder ausgeführt, bis die Abbruchbedingung erreicht ist. Wie bei allen Blöcken, kann der Prozessschritt wiederum ein Block sein, so dass beispielsweise auch eine Sequenz mehrfach ausgeführt werden kann.

In Abbildung 5.9 ist ein Schleifen-Block dargestellt, der mit einem *Multiple-Merge* beginnt. Nach dem Prozessschritt (hier:  $A_1$ ) wird ein *Exclusive Choice* ausgeführt, mit dem die Ausführung wieder zum Multiple-Merge zurückgeht, sofern

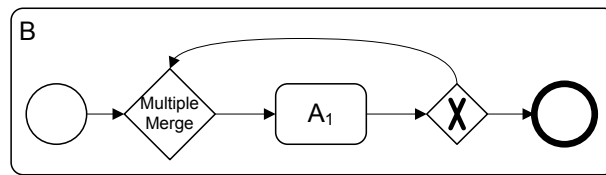


Abbildung 5.9: Kontrollflussblock zur Realisierung von Schleifen (Loop)

die Abbruchbedingung noch nicht erfüllt ist. Dadurch kann bei der Ausführung eine Schleife realisiert werden.

## 5.4 Aggregationsmuster für Qualitätsmerkmale

In diesem Abschnitt werden Aggregationsmuster vorgestellt, mit denen die Ausprägungen der Qualitätsmerkmale einzelner Prozessschritte für einen Block aggregiert werden können. Dabei werden die Aspekte des Qualitätsmodells untersucht, die für die Aggregationsmuster relevant sind. Insbesondere werden die Qualitätsmerkmale der Prozessschritte und das aggregierte Qualitätsmerkmal des Blocks betrachtet. Außerdem wird jeweils untersucht, für welche Kontrollflussblöcke die Muster anwendbar sind.

Der Begriff Muster wurde in der Software-Entwicklung im Zusammenhang mit Entwurfsmustern (englisch design pattern) unter anderem in [54] vorgestellt. Entwurfsmuster beschreiben eine Vorlage zur Problemlösung. Ein anderes Beispiel für Muster sind die in Abschnitt 5.3.1 vorgestellten Workflow-Pattern. Allgemein definieren Muster grundlegende Konzepte, die wiederverwendbar sind. Muster klassifizieren und beschreiben die Anwendung der Konzepte. Das Ziel der Verwendung von Aggregationsmustern für Qualitätsmerkmale ist es, sie so auf einen Geschäftsprozess anzuwenden, dass dieser als zusammengesetzter Service mit entsprechend aggregierten Qualitätsmerkmalen angeboten werden kann (siehe Abschnitt 3.4.3).

### 5.4.1 Vorbemerkungen und Annahmen

Ein Aggregationsmuster beschreibt, wie die Ausprägungen der Qualitätsmerkmale von Prozessschritten in einem Kontrollflussblock aggregiert werden. Dadurch ergibt sich eine Ausprägung für ein Qualitätsmerkmal, das dem Kontrollflussblock zugeordnet wird.

**Rekursive Aggregation** Ausgehend von der obersten Ebene der Hierarchie, also dem obersten Kontrollflussblock, wird über die Prozessschritte in diesem Block aggregiert. Wenn ein Prozessschritt wiederum ein Sub-Prozess ist, der durch einen weiteren Kontrollflussblock definiert ist, so ist zunächst dieser Kontrollflussblock zu aggregieren. Somit lässt sich durch ein rekursives Vorgehen bis zur untersten

Ebene der Prozessdefinition die Aggregation durchführen. Auf der untersten Ebene der Hierarchie sind die Ausprägungen der Qualitätsmerkmale für alle Prozessschritte bekannt, so dass die Aggregation hier durchgeführt werden kann. Die Abbruchbedingung der Rekursion ist erreicht und auf den höheren Ebenen der Prozessdefinition können die Qualitätsmerkmale aggregiert werden. Somit können die Muster rekursiv und blockweise angewandt und letztendlich Aussagen über die Qualität des Gesamtprozesses getroffen werden.

Für den Produktionsprozess aus Abbildung 5.2 sind somit zunächst die Qualitätsmerkmale für den AND-Block zu aggregieren. Die Ergebnisse für Kosten und Zeit sind dann für die Sequenz zu ermitteln, wobei neben den beiden ersten Prozessschritten auch der aggregierte Block berücksichtigt wird.

**Annahmen** Damit Aggregationsmuster sinnvoll angewandt werden können, sind die folgenden Annahmen zu treffen:

- Es existiert eine Prozessdefinition nach dem blockstrukturierten Ansatz, mit genau einem Block auf der höchsten Ebene.
- Ein Block besteht aus mindestens einem Prozessschritt und ein Prozessschritt kann ein atomarer Prozessschritt oder ein weiterer Block sein.
- Während der Ausführung einer Prozessinstanz werden Services aufgerufen, das heißt, der Geschäftsprozess wird in einer serviceorientierten Umgebung ausgeführt.
- Für alle atomaren Prozessschritte sind Qualitätsmerkmale definiert und die Ausprägungen angegeben. Dann kann das Aggregationsmuster auf die Ausprägungen der Qualitätsmerkmale aller Prozessschritte angewandt werden.
- Es wird eine rekursive, blockweise Anwendung von Aggregationen entlang der Hierarchie vorgenommen.
- Die Aggregation der Qualitätsmerkmale für den höchsten Block liefert die Qualitätsmerkmale für den Geschäftsprozess.

Ein Prozessschritt kann also ein Block sein oder ein atomarer Prozessschritt, der wiederum einen Service aufrufen kann. Es wird davon ausgegangen, dass für alle atomaren Prozessschritte das zu aggregierende Qualitätsmerkmal definiert sein muss. Daher wird im Folgenden ein atomarer Prozessschritt als der Aufruf eines Services angesehen. Die Aggregationsmuster werden somit über Services definiert, also für atomare Prozessschritte. Daher werden im Folgenden die Begriffe Prozessschritt und Service als Synonyme verwendet. Wie in der rekursiven Aggregation beschrieben, lässt sich auch über einen Block aggregieren, wobei zunächst die Qualitätsmerkmale des Block selbst zu ermitteln sind.

**Anforderungen an Qualitätsmerkmale** Die Aggregation von Qualitätsmerkmalen ist nur dann sinnvoll möglich, wenn einer der folgenden drei Fälle zutrifft:

1. Die zu aggregierenden Qualitätsmerkmale sind für Services spezifiziert, die denselben Service-Typ implementieren. Sie basieren also auf derselben Definition. Hierbei handelt es sich dann um Ausprägungen desselben Qualitätsmerkmals entsprechend Definition 21.
2. Die zu aggregierenden Qualitätsmerkmale sind für Services unterschiedlicher Service-Typen spezifiziert, basieren aber auf derselben Definition eines Qualitätsmerkmals. Dies gilt beispielsweise für allgemeine Qualitätsmerkmale, die als standardisiert angesehen werden können, wie in Abschnitt 4.5.4 beschrieben.
3. Bei der Prozessmodellierung wird explizit festgelegt, welche Qualitätsmerkmale kompatibel sind, so dass ein Aggregationsmuster angewandt werden kann. Beispielsweise können unterschiedliche Service-Typen verschiedene Definitionen eines Qualitätsmerkmals *Kosten* besitzen, aber die Ausprägungen sind dennoch vergleichbar und aggregierbar. Eine Festlegung, welche Qualitätsmerkmale kompatibel sind, ist vom Modellierer explizit zu treffen.

**Anwendung der Muster** Wird für ein Qualitätsmerkmal  $q$  der Prozessschritte  $S_i$  ein Aggregationsmuster angewendet, dann ist der aggregierte Wert die Ausprägung für ein Qualitätsmerkmal  $q'$  des Blocks. Dieses aggregierte Qualitätsmerkmal des Blocks ( $q'$ ) ist zu spezifizieren, so dass es für weitere Aggregationen (auf der nächsthöheren Ebene) verfügbar ist. Sei für den AND-Block des Produktionsprozess aus Abbildung 5.2 das Qualitätsmerkmal *Kosten* aggregiert. Dann ist für den AND-Block ebenfalls ein Qualitätsmerkmal *Kosten* zu definieren und das Resultat der Aggregation ist dessen Ausprägung. In diesem Fall ist das Qualitätsmerkmal des Blocks ebenfalls  $q_{cost}$ . Wie gezeigt wird, kann das Qualitätsmerkmal aber nicht immer übernommen werden, sondern muss unter Umständen angepasst werden.

Innerhalb eines Kontrollflussblocks werden mehrere Prozessschritte spezifiziert. Ein Prozessschritt kann einen Block repräsentieren, einen Service-Aufruf oder eine andere Aktivität. Im Folgenden wird davon ausgegangen, dass jeder Prozessschritt durch einen Service-Aufruf realisiert wird. Wenn andere Aktivitäten modelliert werden sollen, so sind die Qualitätsmerkmale dieser Aktivitäten so zu definieren, dass die Aggregationsmuster angewandt werden können. Dies bedeutet insbesondere, dass die Ausprägungen der Qualitätsmerkmale, die aggregiert werden sollen, auf identischen oder kompatiblen Definitionen basieren. Daher kann aus Sicht der Aggregationsmuster eine solche Aktivität auch als Service angesehen werden. Zum besseren Verständnis wird daher im Folgenden ein Prozessschritt als Service angesehen. Ebenso wird davon ausgegangen, dass für alle Prozessschritte dasselbe Qualitätsmerkmal  $q$  definiert ist, das aggregiert werden soll.

**Aufbau der Muster** Jedes Muster wird nach dem folgenden Schema vorgestellt:

- In einer **Einleitung** wird das Muster benannt und kurz beschrieben. Der Zweck des Musters wird anhand typischer Qualitätsmerkmale illustriert und somit der Einsatz des Musters motiviert.
- **Definition:** Angabe des Aggregationsverfahrens.
- Die **Anwendbarkeit** wird aus der Sicht zweier Aspekte beschrieben:
  - **Qualitätsmerkmal:** Es wird beschrieben, welche Anforderungen bezüglich Skalenniveau und Metrik erfüllt werden müssen, damit die Aggregation sinnvoll angewendet werden kann. Weiterhin wird beschrieben, inwieweit die Definition des Qualitätsmerkmals für das aggregierte Qualitätsmerkmal angepasst oder übernommen werden kann. Hierbei wird insbesondere die Metrik betrachtet.
  - **Kontrollfluss:** Es wird festgelegt, bei welchen Kontrollflussblöcken dieses Aggregationsmuster anwendbar sind und welche weiteren Voraussetzungen erfüllt sein müssen.
- **Beispiel:** Beispiele für Qualitätsmerkmale und Kontrollflussblöcke werden vorgestellt, die den Einsatz des Aggregationsmusters zeigen und die Ergebnisse der Aggregation erklären.
- **Diskussion:** Die Diskussion umfasst Konsequenzen (Vor- und Nachteile) und gegebenenfalls Querverweise auf andere Muster. Insbesondere werden hier auch Varianten diskutiert.

## 5.4.2 Summe

Das Aggregationsmuster Summe addiert die Ausprägungen der Qualitätsmerkmale aller Prozessschritte eines Kontrollflussblocks. Die Summe ist beispielsweise anzuwenden auf das Qualitätsmerkmal Kosten, wo die Summe über alle Prozessschritte zu bilden ist, unabhängig davon, ob die Ausführung nacheinander oder nebenläufig geschieht.

**Definition 27** Sei ein Kontrollflussblock  $B$  mit  $n$  Prozessschritten  $(S_1, \dots, S_n)$  spezifiziert und alle Prozessschritte haben Ausprägungen für das Qualitätsmerkmal  $q$ . Dann liefert die Anwendung des Aggregationsmusters Summe die Ausprägung des Qualitätsmerkmals  $q'$  des Kontrollflussblocks  $B$  wie folgt:

$$\lambda_B(q') = \sum_{i=1}^n \lambda_{S_i}(q)$$

**Qualitätsmerkmal** Diese Aggregation ist ab einem Qualitätsmerkmal der Intervallskala möglich. Erst ab diesem Skalenniveau sind die Abstände zwischen einzelnen Werten definiert, so dass eine Addition zweier Werte einen sinnvollen Wert ergeben kann. Die Metrik muss unter Umständen angepasst werden: Wenn es sich um einen OR-Kontrollflussblock handelt, so stellt die Summe den maximalen Wert dar, der dann erreicht wird, wenn alle der alternativen Pfade betreten werden. Daher ist das Qualitätsmerkmal als  $q'$  bezeichnet.

**Kontrollfluss** Das Aggregationsmuster Summe kann für alle Kontrollflussblöcke angewandt werden, bei denen alle modellierten Prozessschritte ausgeführt werden, also Sequenz und paralleler Ausführung (And und AndNM). Bei einem Loop-Kontrollflussblock muss bekannt sein, wie oft die Schleife durchlaufen wird. Dann wird der Prozessschritt  $S_i$  innerhalb der Schleife  $loop_i$ -mal durchlaufen, so dass die Summe auch in der folgenden Form berechnet werden kann:

$$\lambda_B(q') = loop_i \lambda_{S_i}(q)$$

Für die Kontrollflussblöcke, bei denen nicht immer alle Prozessschritte ausgeführt werden (XOR und OR) sowie für den AndNM-Block sind unter Umständen andere Aggregationsmuster anzuwenden (siehe Abschnitte 5.4.5 und 5.4.7). Sollen die Werte aller Prozessschritte, unabhängig davon, ob sie tatsächlich aufgerufen werden oder nicht, aggregiert werden, so ist die Summe ebenfalls bei XOR und OR anzuwenden. Bei einem OR-Block stellt die Summe den Wert da, der auftritt, wenn alle Pfade ausgeführt werden. Somit ist der aggregierte Wert ein Maximalwert und die Metrik des Qualitätsmerkmals ist entsprechend anzupassen.

**Beispiel** Ein Beispiel für ein Qualitätsmerkmal sind die Kosten für die Ausführung eines Services, die in einer Währung angegeben werden. Für alle Services, die aufgerufen werden fallen diese Kosten an, daher sind die Kosten für den gesamten Kontrollflussblock als Summe anzugeben. Ein weiteres Beispiel ist der Verbrauch eines Betriebsmittels, wie Kraftstoff. Hat jeder aufgerufene Service seinen Verbrauch angegeben, so ergibt die Summe den Gesamtverbrauch des Kraftstoffes.

Seien die Kosten für die Services definiert als Fixkosten die anfallen, sobald der Service aufgerufen werden könnte. Somit ist die Bereitstellung des Services bereits als Ursache der Kosten anzusehen [134]. Dann kann die Summe auch für die Kosten aller XOR und OR Kontrollflussblöcke als sinnvoller Wert berechnet werden.

**Diskussion** In Bezug auf die Kontrollflussblöcke wurden bereits weiterführende Aspekte diskutiert und auf spätere Abschnitte verwiesen. Das Qualitätsmerkmal des Blocks kann identisch mit den aggregierten Qualitätsmerkmalen sein (zum Beispiel  $q_{cost}$ ). Unter Umständen sind aber vom Modellierer Anpassungen vorzunehmen, so dass ein neues Qualitätsmerkmal ( $q'$ ) definiert werden muss.



### 5.4.3 Produkt

Das Aggregationsmuster Produkt multipliziert die Ausprägungen aller Prozessschritte. Das Produkt ist üblicherweise bei Verhältnissen sinnvoll: Das Qualitätsmerkmal Verfügbarkeit, das eine Erreichbarkeit des Services in Prozent angibt, ist beispielsweise zu multiplizieren. Für alle Prozessschritte ist dabei das Produkt zu bilden, unabhängig davon, ob die Ausführung nacheinander oder nebenläufig geschieht.

**Definition 28** Sei ein Kontrollflussblock  $B$  mit  $n$  Prozessschritten  $(S_1, \dots, S_n)$  spezifiziert und alle Prozessschritte haben Ausprägungen für das Qualitätsmerkmal  $q$ . Dann liefert die Anwendung des Aggregationsmusters Produkt die Ausprägung des Qualitätsmerkmals  $q'$  des Kontrollflussblocks  $B$  wie folgt:

$$\lambda_B(q') = \prod_{i=1}^n \lambda_{S_i}(q)$$

Für einem Schleifen-Kontrollflussblock wird der Prozessschritt  $S_i$  innerhalb der Schleife  $loop_i$ -mal durchlaufen, so dass das Produkt auch in der folgenden Form berechnet werden kann:

$$\lambda_B(q') = (\lambda_{S_i}(q))^{loop_i}$$

**Qualitätsmerkmal** Diese Aggregation ist nur für ein Qualitätsmerkmal der Ratioskala möglich. Die Metrik muss unter Umständen angepasst werden: Wenn es sich um einen OR-Kontrollflussblock handelt, so stellt das Produkt den Wert dar, der dann erreicht wird, wenn alle der alternativen Pfade betreten werden (siehe auch Aggregationsmuster Summe).

**Kontrollfluss** Das Produkt kann für alle Kontrollflussblöcke angewandt werden, bei denen alle modellierten Prozessschritte ausgeführt werden, also Sequenz und parallele Ausführung (AND und AndNM). Bei einer Schleife muss zusätzlich bekannt sein, wie oft sie durchlaufen wird, so dass die oben angegebene Variable  $loop_i$  die Anzahl der Schleifendurchläufe darstellt.

Für die Kontrollflussblöcke, bei denen nicht immer alle Prozessschritte ausgeführt werden (OR und XOR), sowie für den AndNM-Block sind unter Umständen andere Aggregationsmuster anzuwenden. Sollen die Werte aller Prozessschritte, unabhängig davon, ob sie tatsächlich aufgerufen werden oder nicht, aggregiert werden, so ist das Produkt ebenfalls bei XOR und OR anzuwenden.

**Beispiele** Verfügbarkeit: Seien drei Services in einer Sequenz aufgerufen, für die jeweils dasselbe Qualitätsmerkmal für die Verfügbarkeit definiert und die Ausprägungen jedes Services mit 99,9% angegeben. Dann beträgt die Verfügbarkeit eines Prozesses der diese drei Services aufruft 97,0299%.

**Reinheit einer Substanz:** Sei ein Geschäftsprozess definiert, der eine Substanz in mehreren Schritten sequentiell bearbeitet, wobei in jeden Bearbeitungsschritt die Reinheit der Substanz abnimmt. Wenn die Reinheit nach dem Schritt jeweils auf 95% absinkt, so liegt die Reinheit nach zehn Bearbeitungsschritten unter 60%.

**Diskussion** Das Aggregationsmuster Produkt ist dem Aggregationsmuster Summe semantisch relativ ähnlich, wird aber für andere Qualitätsmerkmale eingesetzt, wie in den Beispielen geschildert. Für derartige Qualitätsmerkmale wird in der Literatur vorgeschlagen, dass die Werte logarithmiert und dann addiert werden können [185]. Dann kann zwar das Aggregationsmuster Summe angewendet werden, allerdings geht dann die Aussagekraft des Wertes verloren.

#### 5.4.4 Extremwerte: Maximum und Minimum

Diese Aggregation ist ab einem Qualitätsmerkmal der Intervallskala möglich. Erst ab diesem Skalenniveau sind die Abstände zwischen einzelnen Werten definiert, so dass eine Addition zweier Werte einen sinnvollen Wert ergeben kann.

Zur Ermittlung der Extremwerte seien zwei Aggregationsmuster definiert: Maximum und Minimum. Bei einem Extremwert wird eine einzelne Ausprägung eines Prozessschrittes des Blocks betrachtet und für den Block übernommen. Der Wert stellt die Ausprägung eines Qualitätsmerkmals dar, das als Extremwert zu spezifizieren ist. Mit diesem Muster wird beispielsweise die maximale und minimale Temperatur für einen Block definiert.

**Definition 29** Sei ein Kontrollflussblock  $B$  mit  $n$  Prozessschritten  $(S_1, \dots, S_n)$  definiert und alle Prozessschritte haben Ausprägungen für das Qualitätsmerkmal  $q$ . Sei  $A_q$  die Menge aller Ausprägungen von  $q$ , also  $A_q = \{\lambda_{S_i}(q) \mid 1 \leq i \leq n\}$ . Weiterhin liefere die Funktion  $max()$  das größte Element einer Menge und  $min()$  das kleinste Element einer Menge [24].

- Dann liefert die Anwendung des Aggregationsmusters Maximum die Ausprägung des Qualitätsmerkmals  $q'$  des Kontrollflussblocks  $B$  wie folgt:

$$\lambda_B(q') = \max(A_q)$$

- Dann liefert die Anwendung des Aggregationsmusters Minimum die Ausprägung des Qualitätsmerkmals  $q'$  des Kontrollflussblocks  $B$  wie folgt:

$$\lambda_B(q') = \min(A_q)$$

**Qualitätsmerkmal** Extremwerte können berechnet werden, sobald auf den Werten eine Ordnung definiert ist, somit ab der Ordinalskala. Zu beachten ist, dass die Metrik entsprechend angepasst werden muss, falls vorher kein Extremwert angegeben wurde.

**Kontrollfluss** Extremwerte sind für alle Kontrollflussblöcke anwendbar. Es wird eine Ausprägung eines Prozessschrittes ermittelt. Bei einer Schleife (Loop) ist es der Wert des Prozessschrittes innerhalb der Schleife.

**Beispiele** Die maximale und minimale Temperatur während eines Prozesses kann mit diesem Aggregationsmuster ermittelt werden. Weiterhin ist bei Sicherheitsaspekten der minimale Extremwert relevant: Sei beispielsweise die Verschlüsselung der Kommunikation angegeben wie im Transport-Beispiel definiert. Dann gibt der Minimalwert die minimale Sicherheit für den gesamten Prozess an. Generell ist bei allen Qualitätsmerkmalen, für die üblicherweise die Extremwerte für einen Requestor relevant sind, kann dieses Aggregationsmuster angewendet werden. Als weiteres Beispiel sei der Durchsatz in Transaktionen pro Sekunde genannt.

**Diskussion** Wie bereits oben geschildert, ist in einigen Fällen eine Summe als Extremwert anzusehen, beispielsweise, wenn die Werte aller OR-Pfade addiert werden. Dann ist die Metrik entsprechend anzupassen. Bemerkenswert hierbei ist, dass diese Summe den maximalen Wert für die Kosten angibt, diese aber nicht übereinstimmt mit dem Extremwert der betrachteten Prozessschritte. Bei der Modellierung solcher Qualitätsmerkmale ist also sorgfältig zu entscheiden, ob von den verfügbaren Werten einer als Extremwert gewählt werden soll, oder die Summe als maximal erreichbarer Wert spezifiziert werden soll.

### 5.4.5 Aggregationen unter Wahrscheinlichkeiten

Bei den Kontrollflussblöcken XOR und OR ist zur Modellierungszeit nicht bekannt, welcher Pfad bei einer Instanz betreten wird. Ebenso ist bei einer Schleife unter Umständen nicht bekannt, wie häufig sie durchlaufen wird. Dies wird erst zur Laufzeit entschieden und kann von vorhergehenden Prozessschritten oder Daten der Prozessinstanz abhängen. Daher kann keine eindeutige Aussage zur Modellierungszeit über Qualitätsmerkmale getroffen werden.

In der Modellierung von Geschäftsprozessen ist dennoch ein Lösungsansatz möglich: Dazu werden den alternativen Pfaden jeweils Wahrscheinlichkeiten für ihre Ausführung zugeordnet [31, 72, 134, 185]. Dadurch sind weitere Aggregationsmuster anwendbar, die im Folgenden beschrieben werden.

**Definition 30** Für jeden Prozessschritt  $S_i$  ( $1 \leq i \leq n$ ) sei eine Wahrscheinlichkeit  $p_i \in [0, 1]$  definiert, mit der dieser Prozessschritt ausgeführt wird. Zusätzlich gelte je nach Kontrollflussblock:

- In einem XOR-Block ist die Summe aller Wahrscheinlichkeiten 100%, also
$$\sum_{i=1}^n p_i = 1.$$

- In einem OR-Block ist die Summe aller Wahrscheinlichkeiten größer<sup>2</sup> als 100%, also  $\sum_{i=1}^n p_i \geq 1$ .

Dann können folgende Werte durch die Anwendung eines Aggregationsmusters berechnet werden:

- **Erwartungswert:** Es kann der Erwartungswert unter Berücksichtigung der Wahrscheinlichkeiten, dass der jeweilige Prozessschritt ausgeführt wird, errechnet werden [24]:

$$E(q) = \sum_{i=1}^n p_i \lambda_{S_i}(q)$$

- **Varianz und Standardabweichung:** Die Varianz  $V(q)$  und die Standardabweichung  $\sigma_q$  sind demnach wie folgt definiert [24]:

$$\begin{aligned} V(q) &= E((\lambda_{S_i}(q) - E(q))^2) \\ \sigma_q &= \sqrt{V(q)} \end{aligned}$$

**Qualitätsmerkmal** Diese Aggregation ist für Qualitätsmerkmale ab der Intervallskala möglich. Erst ab der Intervallskala sind die Abstände zwischen einzelnen Werten definiert, so dass die Summe der Werte einen sinnvollen Wert ergeben kann.

Es ist zu beachten, dass sich die Semantik des Wertes gegenüber dem aggregierten Qualitätsmerkmal ändert, da ein Erwartungswert abgebildet wird. Zusätzlich lassen sich auch die Varianz oder die Standardabweichung angeben. Die Metrik des Qualitätsmerkmals des Blocks ist also anzupassen. Da die Standardabweichung die gleiche Maßeinheit wie der Erwartungswert besitzt, ist sie im Allgemeinen vorzuziehen. Aggregiert können nur Qualitätsmerkmale, deren Werte feste Werte oder Erwartungswerte darstellen. Bei Extremwerten ist eine Aggregation unter Wahrscheinlichkeit nicht sinnvoll anwendbar.

**Kontrollfluss** Wie bereits erwähnt, kann dieses Muster angewandt werden bei XOR- und OR-Blöcken. Weiterhin ist eine Übertragung auf eine Schleife möglich, wenn die Anzahl der Schleifendurchläufe zur Modellierungszeit nicht feststeht. Dabei kommt das *Loop Unrolling*-Verfahren [57] zum Einsatz. Dabei wird für jeden Schleifendurchlauf beschrieben, mit welcher Wahrscheinlichkeit ein weiterer Schleifendurchlauf durchgeführt wird beziehungsweise die Schleife abgebrochen wird. Dadurch kann eine Wahrscheinlichkeit für einen Schleifendurchlauf, für zwei Schleifendurchläufe et cetera errechnet werden, so dass die Werte zu einem Erwartungswert aggregiert werden können.

<sup>2</sup>Diese Summe stellt hier keine Wahrscheinlichkeit dar, sondern dient lediglich der Betrachtung des Zusammenhangs zwischen den Summanden. Es ergibt sich ein für die Service-Qualität relevanter Unterschied zwischen einem OR und einem XOR-Split. Dies hat dann Auswirkungen auf die Bedeutung (also die Metrik) des aggregierten Wertes. In einem OR-Block wird für jeden Pfad die Wahrscheinlichkeit angegeben, mit der er betreten wird, wobei zu bedenken ist, dass *mindestens* ein Pfad betreten werden muss. Siehe dazu auch Abbildung 5.10(b).

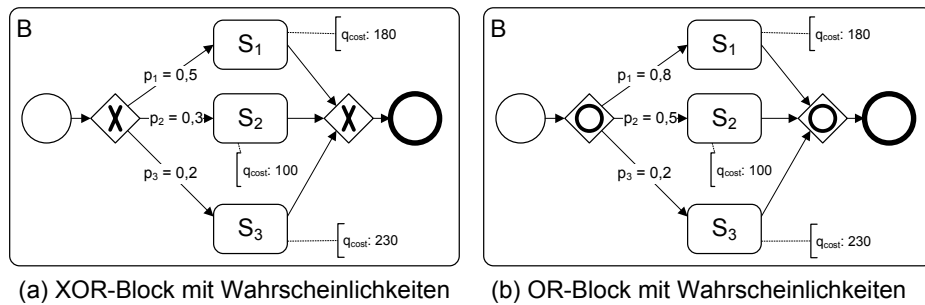


Abbildung 5.10: Kontrollfluss-Blöcke mit Angabe der Wahrscheinlichkeiten

**Beispiele** Abbildung 5.10 zeigt zwei Beispiel-Blöcke mit den Prozessschritten  $S_1$  bis  $S_3$  und angegebenen Wahrscheinlichkeiten für die Ausführung der einzelnen Pfade. Für die Prozessschritte ist das Qualitätsmerkmal für Kosten  $q_{cost}$  angegeben. Seien die Kosten für  $S_1$  180 €,  $S_2$  100 € und für  $S_3$  230 €, so ergibt sich für den XOR-Block (Abbildung 5.10(a)) ein Erwartungswert von 166 € bei einer Standardabweichung von 47,16 € und für den OR-Block (Abbildung 5.10(b)) ein Erwartungswert von 240 € bei einer Standardabweichung von 112,69 €. Für Beispiele zum *Loop Unrolling*-Verfahren sei auf [57] verwiesen.

**Diskussion** Die Berechnung von Erwartungswerten und Standardabweichungen ist nicht immer sinnvoll einzusetzen. Die aggregierten Ausprägungen müssen von Qualitätsmerkmalen stammen, die feste Werte darstellen, wie beispielsweise Kosten. Wenn die zu aggregierenden Werte Extremwerte darstellen, so ist eine Aggregation als Erwartungswert in der Regel nicht sinnvoll. Beispielsweise ist die Aggregation von Werten, die eine maximale Temperatur angeben, als Erwartungswert nicht sinnvoll. Hier ist ein Aggregationsmuster Extremwert sinnvoll.

### 5.4.6 Mittelwerte

Für ein Qualitätsmerkmal in einem Kontrollflussblock können Mittelwerte berechnet werden. Entgegen dem vorherigen Aggregationsmuster sind dabei keine Wahrscheinlichkeiten notwendig, wenn mittlere Ausprägungen des aggregierten Qualitätsmerkmals abgebildet werden sollen. Welche Mittelwerte berechenbar sind, hängt hier vor allem vom Skalenniveau des Qualitätsmerkmals ab, wie bereits in Abschnitt 4.3.5 beschrieben. Im Folgenden werden die Mittelwerte und ihre Berechnung vorgestellt.

**Definition 31** Sei ein Kontrollflussblock  $B$  mit  $n$  Prozessschritten  $(S_1, \dots, S_n)$  spezifiziert und alle Prozessschritte haben Ausprägungen für das Qualitätsmerkmal  $q$ . Dann liefert die Anwendung eines Mittelwert-Musters die Ausprägung des Qualitätsmerkmals  $q'$  des Kontrollflussblocks  $B$  wie folgt:

- **Modus** Bei nominal skalierten Qualitätsmerkmalen kann nur der Modus angegeben werden, der den häufigsten Wert angibt. Somit ist bei Anwendung des Modus die Ausprägung des Blocks  $B$  der häufigste Wert der Prozessschritte. Die Definition des Qualitätsmerkmals  $q'$  ist ähnlich der Definition  $q$ , wobei allerdings ein Modus angegeben wird.
- **Median** Ab ordinal skalierten Qualitätsmerkmalen kann der Median berechnet werden, der den Wert angibt, der die Ausprägungen in zwei gleich große Hälften teilt. Um den Median zu ermitteln, sind die Ausprägungen zunächst zu ordnen: Seien  $(a_1, a_2, \dots, a_n)$  die geordneten Ausprägungen  $\lambda_{S_i}(q)$ . Dann ergibt die Anwendung des Medians die Ausprägung für den Block  $B$  wie folgt:

$$\lambda_B(q') = \begin{cases} a_{(n+1)/2} & n \text{ ungerade} \\ \frac{1}{2} (a_{(n/2)} + a_{(n/2+1)}) & n \text{ gerade} \\ a_{(n/2)} \vee a_{(n/2+1)} & n \text{ gerade} \wedge scale_q = ordinal \end{cases} \wedge scale_q \in \{interval, ratio\}$$

Im letzten Fall muss bei der Anwendung des Aggregationsmusters angegeben werden, ob der Untermedian  $a_{(n/2)}$  oder der Obermedian  $a_{(n/2+1)}$  als Median genutzt werden soll.

- **Arithmetisches Mittel** Ab Qualitätsmerkmalen auf einer Intervallskala kann das arithmetische Mittel berechnet werden, das sich errechnet als Summe über die Ausprägungen geteilt durch  $n$ :

$$\lambda_B(q') = \frac{1}{n} \sum_{i=1}^n \lambda_{S_i}(q)$$

- **Geometrische Mittel** Ab rational skalierten Qualitätsmerkmalen ist das geometrische Mittel anwendbar, dass ein geeigneter Mittelwert für Qualitätsmerkmale ist, von denen das Produkt anstelle der Summe sinnvoll ist, zum Beispiel von Verhältnissen wie der Verfügbarkeit (siehe Aggregationsmuster Produkt, Abschnitt 5.4.3). Das geometrische Mittel ist die  $n$ -te Wurzel aus dem Produkt der Ausprägungen:

$$\lambda_B(q') = \sqrt[n]{\prod_{i=1}^n \lambda_{S_i}(q)}$$

**Qualitätsmerkmal** Welche Mittelwerte anwendbar sind, hängt wie oben geschildert vom Skalenniveau ab. Offensichtlich ist das geometrische Mittel nur für Werte  $\geq 0$  definiert, hier muss also zusätzlich die Domäne  $dom_q$  beachtet werden. Das aggregierte Qualitätsmerkmal  $q'$  des Blockes ist jeweils so zu definieren, dass der entsprechende Mittelwert angegeben wird.

**Kontrollfluss** Die Mittelwert-Muster sind prinzipiell für alle Kontrollfluss-Blöcke anwendbar. Da bei Schleifen allerdings jeweils die gleiche Ausprägung für den Prozessschritt innerhalb der Schleife vorliegt, ist der aggregierte Mittelwert mit der Ausprägung des Prozessschrittes identisch.

**Beispiel** Es können somit auch ohne Angabe von Wahrscheinlichkeiten die mittleren Kosten für den in Abbildung 5.10(a) dargestellten Block errechnet werden: Das arithmetische Mittel für  $q_{cost}$  beträgt 170 € und der Median 180 €

**Diskussion** Neben der Aggregation unter Wahrscheinlichkeit sind also andere Mittelwerte berechenbar. In der Statistik existieren noch weitere Mittelwerte, so dass zusätzliche Aggregationsmuster definiert werden können, die hier nicht betrachtet wurden [24, 44]. Wird bei einer Aggregation unter Wahrscheinlichkeit für alle Pfade die gleiche Wahrscheinlichkeit angegeben, so sind Erwartungswert und Mittelwert identisch.

#### 5.4.7 Aggregation der ersten N Prozessschritte

Für den Kontrollflussblock AndNM, dessen parallele Ausführung mit einem *N-out-of-M-Join* zusammengeführt wird, können für die Aggregation unter Umständen nur die ersten  $N$  Prozessschritte relevant sein. Es sind also die ersten  $N$  Prozessschritte zu ermitteln, die ihre Ausführung beenden, also diejenigen mit der geringsten Ausführungsdauer. Sie werden in der Menge  $\mathfrak{N}$  zusammengefasst. Listing 5.1 skizziert einen Algorithmus zur Ermittlung der ersten  $N$  Prozessschritte (Services).

**Definition 32** Sei  $\mathfrak{N}$  die Menge der  $N$  Prozessschritte mit der geringsten Ausführungsdauer des Kontrollflussblocks  $B$ . Dann ist die Aggregation auf alle Elemente der Menge  $\mathfrak{N}$  anzuwenden. Somit ergeben sich jeweils die folgenden Aggregationsregeln:

$$\text{Summe : } \lambda_B(q') = \sum_{S \in \mathfrak{N}} \lambda_S(q)$$

$$\text{Produkt : } \lambda_B(q') = \prod_{S \in \mathfrak{N}} \lambda_S(q)$$

Für die Extremwerte ist die Menge der Ausprägungen anzupassen, so dass nur die  $\mathfrak{N}$  berücksichtigt werden. Diese Menge sei  $A_q^{\mathfrak{N}} = \{\lambda_S(q) \mid S \in \mathfrak{N}\}$

$$\text{Minimalwert : } \lambda_B(q') = \min \left( A_q^{\mathfrak{N}} \right)$$

$$\text{Maximalwert : } \lambda_B(q') = \max \left( A_q^{\mathfrak{N}} \right)$$

Listing 5.1: Algorithmus zur Ermittlung der Menge  $\mathfrak{N}$  der  $N$  schnellsten Services in einem AndNM-Kontrollflussblock

---

```

Prozedur ersteN(qtime, services, n){

    Eingabe: qualitätsmerkmal qtime, Set services, int n
    Ausgabe: Set nservices

    /* qtime : Das Qualitätsmerkmal Ausführungsdauer */
    /* services : Menge aller Services */
    /* nservices : Menge der ersten N Services */

    sortiere services über qtime nach listservices

    for int i=1 to n {
        füge i. Element aus listservices nservices hinzu
    }
    Gebe nservices zurück
}

```

---

**Qualitätsmerkmale** Je nach angewendeter Aggregationsregel ist das notwendige Skalenniveau zu beachten. Somit ist für die Summe mindestens eine Intervallskala notwendig, für das Produkt die Ratioskala und für Extremwerte mindestens eine ordinale Skala.

**Kontrollfluss** Dieses Aggregationsmuster ist nur anwendbar für einen Kontrollflussblock, dessen parallele Ausführung durch einen N-out-of-M-Join zusammengeführt wird. Somit ist es nur für einen AndNM-Block anwendbar.

**Beispiel** Als Beispiel für einen N-out-of-M-Join wird in der Literatur ein Gutachter-Verfahren genannt: drei Gutachter werden beauftragt und sobald zwei Gutachten eingetroffen sind, kann die Ausführung fortgesetzt werden. Die Ausführungsdauer für den Kontrollflussblock ist dann der Maximalwert der ersten  $N$ , also:  $\lambda_B(q_{time}) = \max(A_{q_{time}}^{\mathfrak{N}})$ .

**Diskussion** Die Definition der Workflow-Patterns sieht für einen N-out-of-M-Join vor, dass nach Eintreffen des Kontrollflusses über  $N$  Kanten die Ausführung fortgesetzt wird [156]. Somit ist die Ausführungsdauer aller Prozessschritte in einem AndNM-Block zur Ermittlung der ersten  $N$  notwendig und muss als Qualitätsmerkmal entsprechend definiert sein. Unter anderem muss sie als fester Wert definiert sein und nicht beispielsweise als maximale Ausführungsdauer.



### 5.4.8 Qualitätsmerkmal übernehmen

Dieses Aggregationsmuster beschreibt die Übernahme eines Qualitätsmerkmals von einem Prozessschritt für den Block. Das Qualitätsmerkmal kann dabei nur für einen Prozessschritt definiert sein oder nur bei einem Prozessschritt als relevant angesehen werden. Beispielsweise sei für den Produktionsprozess die Angabe über die Sicherheit der Kommunikation relevant. Allerdings findet nur im Transport-Prozessschritt eine Kommunikation mit Dritten statt, so dass das Qualitätsmerkmal  $q_{encrypt}$  (siehe Abschnitt 4.4) für den Produktionsprozess übernommen wird. Dabei werden Definition des Qualitätsmerkmals und Ausprägung jeweils vom dazugehörigen Block übernommen.

**Definition 33** Sei ein Kontrollflussblock  $B$  mit  $n$  Prozessschritten  $(S_1, \dots, S_n)$  definiert und mindestens ein Prozessschritt hat eine Ausprägungen für das Qualitätsmerkmal  $q$ . Weiterhin sei das Qualitätsmerkmal des Prozessschrittes  $S_i (1 \leq i \leq n)$  für den Block zu übernehmen. Dann liefert die Anwendung des Aggregationsmusters Übernahme die Ausprägung des Qualitätsmerkmals  $q$  des Kontrollflussblocks  $B$  wie folgt:

$$\lambda_B(q) = \lambda_{S_i}(q)$$

**Qualitätsmerkmal** Prinzipiell kann jedes Qualitätsmerkmal für den übergeordneten Block übernommen werden. Es gibt somit keine Einschränkungen bezüglich des Qualitätsmerkmals. Das Qualitätsmerkmal des Blocks ist in der Regel identisch mit dem übernommenen, das bedeutet, die Metrik, Domäne und Skalenniveau ist dasselbe.

**Kontrollfluss** Es gibt prinzipiell keine Einschränkung für die Übernahme. Allerdings ist zu beachten, ob der Kontrollflussblock alternative Pfade spezifiziert (OR bzw. XOR). Stammt das übernommene Qualitätsmerkmal aus einem Pfad, der nicht in allen Ausführungen des Kontrollflussblocks ausgeführt wird, so ist die Anwendung des Aggregationsmusters in der Regel nicht sinnvoll. Soll das Qualitätsmerkmal dennoch übernommen werden, ist die Metrik unter Umständen anzupassen.

**Beispiel** Nur im Transport-Prozessschritt des Produktionsprozesses wird eine sicherheitsrelevante Kommunikation durchgeführt. Für diesen Prozessschritt ist das Qualitätsmerkmal  $q_{encrypt}$  definiert. Die Ausprägung wird für AND-Block übernommen und anschließend für den gesamten Prozess. Das Qualitätsmerkmal bleibt dasselbe und wird somit nach außen unverändert weitergegeben.

**Diskussion** Bei diesem Aggregationsmuster werden zwar keine Werte aggregiert, aber es werden die Prozessschritte zusammengefasst. Es findet also eine Informationsverdichtung statt, indem das Qualitätsmerkmal eines Prozessschrittes

für die höhere Ebene übernommen wird. Daher betrachtet das Muster ein Qualitätsmerkmal, welches aggregiert über alle Prozessschritte des Blockes gilt. Somit ist die Bezeichnung Aggregationsmuster auch hier zutreffend.

Das Qualitätsmerkmal des Blockes kann auch geändert werden, wenn auf der höheren Ebene ein anderes Qualitätsmerkmal definiert werden soll. Dann ist allerdings vom Modellierer gegebenenfalls zu spezifizieren, wie der Wert zu transformieren ist.

### 5.4.9 Aggregation mehrerer Qualitätsmerkmale

Bei der Aggregation über mehrere Qualitätsmerkmale werden die Ausprägungen von mindestens zwei Qualitätsmerkmalen zu einem neuen Qualitätsmerkmal  $q'$  für einen Block  $B$  aggregiert. Beispielsweise kann eine Temperatur mit der Zeit in Verbindung gesetzt werden, so dass eine mittlere Temperatur während der Prozessausführung errechnet werden kann.

Die zu aggregierenden Qualitätsmerkmale müssen dazu explizit vom Prozessmodellierer ausgewählt werden, so dass die 3. Anforderung aus Abschnitt 5.4.1 nicht nur Qualitätsmerkmale verschiedener Prozessschritte, sondern auch verschiedene Qualitätsmerkmale eines Prozessschrittes betrifft.

**Definition 34** Sei  $B$  ein Kontrollflussblock mit den Prozessschritten  $S_i, 1 \leq i \leq n$  auf denen die Qualitätsmerkmale  $q_j, 1 \leq j \leq m$  spezifiziert sind. Jeder Prozessschritt habe Ausprägungen für die Qualitätsmerkmale. Die Ausprägungen werden wie folgt als  $(n \cdot m)$ -Tupel definiert:

$$\alpha = (\lambda_{S_1}(q_1), \lambda_{S_1}(q_2), \dots, \lambda_{S_1}(q_m), \lambda_{S_2}(q_1), \dots, \lambda_{S_2}(q_m), \dots, \lambda_{S_n}(q_m))$$

Das Tupel  $\alpha$  enthält also die Ausprägungen aller relevanten Qualitätsmerkmale aller Prozessschritte. Sei  $A$  die Menge aller möglichen  $\alpha$ , also aller Tupel der Ausprägungen für die betrachteten Prozessschritte und Qualitätsmerkmale. Dazu sei  $dom_{q_{ij}}$  die Domäne des Qualitätsmerkmals  $q_j$  des Prozessschrittes  $S_i$ . Dann ist  $A$  das Kreuzprodukt der einzelnen  $dom_{q_{ij}}$ :

$$A = dom_{q_{11}} \times dom_{q_{12}} \times \dots \times dom_{q_{1m}} \times dom_{q_{21}} \times dom_{q_{22}} \times \dots \times dom_{q_{nm}}$$

Weiterhin ist zu spezifizieren, wie die Werte in einem Kontrollflussblock  $B$  zu einer Ausprägung für ein Qualitätsmerkmale  $q'$  mit einer Domäne  $dom_{q'}$  aggregiert werden sollen: Sei  $f$  die Funktion, die eine aggregierte Ausprägung für ein Qualitätsmerkmal  $q'$  berechnet, wobei gilt:

$$f : A \rightarrow dom_{q'}$$

Dann liefert die Anwendung der Funktion auf  $\alpha$  die Ausprägung des Qualitätsmerkmals  $q'$  für den Block  $B$ , das heißt:

$$\lambda_B(q') = f(\alpha)$$

**Qualitätsmerkmal** Eine derartige Aggregation ist für verschiedene Qualitätsmerkmale zulässig. Dabei ist allerdings das Skalenniveau der Qualitätsmerkmale und die zulässigen mathematischen Operationen zu beachten, wenn die Funktion  $f$  erstellt und auf  $\alpha$  angewandt wird. Es entsteht in jedem Fall ein neues Qualitätsmerkmal, dessen Metrik die Funktion  $f$  berücksichtigen muss.

**Kontrollfluss** Dieses Aggregationsmuster ist für alle Kontrollflussblöcke zulässig. Allerdings muss jeweils beachtet werden, dass das Kontrollflussmuster die Funktion bestimmen kann. Beispielsweise hat die Berechnung einer mittleren Temperatur über den Zeitverlauf bei einer Sequenz (Mittelwert über die Zeit) eine andere Bedeutung als bei einem XOR-Block (Erwartungswert).

**Beispiel** Sei  $q_{fixtemp}$  ein fester Temperatur-Wert und  $q_{fixtime}$  eine feste Ausführungsdauer, dann ist die mittlere Temperatur über den Zeitverlauf ( $q_{meantemp}$ ) für einen Sequenz-Block  $B$  mit  $S_i$  Prozessschritten wie folgt zu errechnen:

$$\lambda_B(q_{meantemp}) = \frac{\sum_{i=1}^n \lambda_{S_i}(q_{fixtime}) \cdot \lambda_{S_i}(q_{fixtemp})}{\sum_{i=1}^n \lambda_{S_i}(q_{fixtime})}$$

**Diskussion** Diese Form der Informationsverdichtung erfordert eine präzise Analyse der relevanten Qualitätsmerkmale. Ebenso ist jeweils die Metrik des neuen Qualitätsmerkmals und insbesondere die Art des Wertes (fester Wert, Extremwert oder Erwartungswert) zu beachten.

Darüber hinaus ist zu untersuchen, wie das aggregierte Qualitätsmerkmal in der nächsthöheren Ebene zu aggregieren ist. Soll beispielsweise die Temperatur  $q_{meantemp}$  auf höheren Hierarchie-Ebenen weiter aggregiert werden, ist ebenfalls die Gesamtdauer als Ausprägung eines Qualitätsmerkmals zu spezifizieren. Hierzu ist das Aggregationsmusters Summe über  $q_{fixtime}$  anzuwenden.

Dieses Aggregationsmuster stellt einen allgemeinen Fall dar. Die vorherigen Aggregationsmuster sind als Sonderfälle mit  $m = 1$  dieses Aggregationsmuster anzusehen, wobei jeweils  $f$  entsprechend zu definieren ist.

## 5.5 Metrik der aggregierten Werte

Für die Aggregationsmuster wurden Prozesse betrachtet, die auf einer Blockstruktur basieren. Dadurch ergibt sich eine hierarchische Struktur. Die Anwendung der Aggregationsmuster erfolgt rekursiv entlang der Hierarchie-Ebenen, wie bereits in Abschnitt 5.4.1 skizziert. Somit werden die Qualitätsmerkmale der unteren Ebene aggregiert auf die nächsthöhere übernommen. Dabei entsteht ein Qualitätsmerkmal, das unter Umständen dieselbe Definition hat. Wie bei einigen Aggregationsmustern beschrieben, ist gegebenenfalls die Metrik anzupassen, beispielsweise

se wenn ein Erwartungswert berechnet wird. Die Bedeutung des neuen Qualitätsmerkmals muss also so definiert werden, dass das Aggregationsmuster berücksichtigt wird.

**Art der Werte** Zusätzlich ist zu beachten, dass die Art der Werte die mögliche Semantik eines aggregierten Qualitätsmerkmals einschränkt. Wurden über ein Qualitätsmerkmal aggregiert, das einen Extremwert darstellt, so kann das aggregierte Qualitätsmerkmal des Blocks ebenfalls nur ein Extremwert sein. Wird beispielsweise das Qualitätsmerkmal  $q_{time}$  von  $T_{shipping}$  summiert, so stellt der Wert ebenfalls einen Extremwert dar. Die Ausprägungen der aggregierten Prozessschritte stellten maximale Ausführungszeiten dar, so dass die tatsächlichen Werte bei einer Ausführung geringer sein können. Daher kann ein aggregierter Wert ebenfalls nur einen Maximalwert darstellen, eine andere Art des Wertes ist nicht sinnvoll. Folgende Einschränkungen der Wertart lassen sich definieren:

- Aus einem Maximalwert kann nur ein Maximalwert werden.
- Aus einem Minimalwert kann nur ein Minimalwert werden.
- Aus einem Erwartungswert kann nur ein Erwartungswert werden.

Aus einem festen Wert kann allerdings durch Anwendung eines Aggregationsmusters eine andere Wertart werden: Ein Maximalwert entsteht beispielsweise durch Anwendung des Maximum-Musters oder durch Anwendung der Summe bei einem OR-Block (alle Pfade werden ausgeführt). Ein Minimalwert entsteht durch Anwendung des Minimum-Musters. Ein Mittelwert entsteht bei Anwendung des Mittelwert-Musters oder einer Aggregation unter Wahrscheinlichkeit bei einem XOR- oder OR-Block. Weiterhin kann die Wertart des aggregierten Qualitätsmerkmals auch ein fester Wert bleiben, beispielsweise bei Anwendung der Summe bei einer Sequenz oder bei Maximum-Muster auf einen festen Zeitwert in einem AND-Block. Aus einem festen Wert kann also:

- ein Extremwert (Maximalwert oder Minimalwert) werden,
- ein Erwartungswert werden oder
- ein fester Wert werden.

## 5.6 Order-To-Cash-Prozess

In eine Kooperation mit der SAP SI in Berlin wurde das Qualitätsmodell auf einen repräsentativen Prozess, der mit der SAP SI definiert wurde, angewendet. Dieser Geschäftsprozess wird als *Order-To-Cash-Prozess* bezeichnet und besteht aus den vier Teilprozessen Anfragebearbeitung, Auftragsbearbeitung, Auftragserfüllung und Abrechnung. Im Rahmen einer Master-Arbeit wurden die relevanten Qualitätsmerkmale für diesen Prozess analysiert und formuliert. Anschließend wurden

Kategorie	Qualitätsmerkmal	Domäne	Metrik	Skala
Performanz	Mittlere Antwortzeit	double	Zeit in Millisek. (ms)	Ratio
	Mittlerer Durchsatz	double	Transaktionen / Sek. (tps)	Ratio
Stabilität	Verfügbarkeit	double	Prozent (%)	Ratio
	Zuverlässigkeit	double	Zeit in Stunden (h)	Ratio
Kosten	Nutzungskosten	double	Währung in €	Ratio
	Fixkosten	double	Währung in €	Ratio
	Gesamtkosten	double	Währung in €	Ratio

Tabelle 5.1: Die Qualitätsmerkmale für den Order-To-Cash-Prozess

die Aggregationsmuster angewandt. Als serviceorientierte Umgebung wurde eine firmeninterne Web Services Infrastruktur angenommen, in der Services für die Prozessschritte verfügbar sind. Ziel des Projektes war es, aus unterschiedlichen Varianten der Service-Auswahl die beste Variante zu ermitteln. [134]

### 5.6.1 Qualitätsmerkmale

In einer ersten Phase wurde dazu ein Katalog der relevanten Qualitätsmerkmale für diesen Prozess erarbeitet. Es wurden hier die Qualitätsmerkmale so definiert, dass Ausprägungen für alle Prozessschritte spezifiziert werden konnten. Somit haben alle Services der firmeninternen SOA dieselben Qualitätsmerkmale. Es wurden die in Tabelle 5.1 dargestellten Qualitätsmerkmale definiert. In der Tabelle sind jeweils Name, Domäne, Skala und eine Kurzform der Metrik dargestellt. Zur besseren Übersicht wurden die Qualitätsmerkmale zudem in die Kategorien *Performanz*, *Stabilität* und *Kosten* zugeordnet. Im Folgenden werden die Metriken kurz erläutert.

**Performanz** Für die Performanz sollten die mittlere Antwortzeit sowie der mittlere Durchsatz betrachtet werden. Die *Antwortzeit* ist definiert worden als die Zeit vom Aufruf eines Services bis zur Antwort an den Requestor; als Maßeinheit ist Millisekunden (ms) definiert. Der *Durchsatz* ist definiert worden als die Anzahl der Service-Aufrufe pro Zeiteinheit. Es wird angegeben, wie hoch die Kapazität des Services ist, also wie viele Anfragen in einer Sekunde entgegen genommen und bearbeitet werden können. Als Maßeinheit wurde Transaktionen pro Sekunde (tps) gewählt.

**Stabilität** Für die Kategorie Systemstabilität wurden die Qualitätsmerkmale Verfügbarkeit und Zuverlässigkeit wie folgt vereinbart: *Verfügbarkeit* ist die Aussage, mit welcher Wahrscheinlichkeit (Angabe in Prozent) der Service bei einem Aufruf in einem funktionsfähigem Zustand ist, also die Leistung erbringen kann. Die *Zuverlässigkeit* gibt eine kontinuierliche Stabilität an, indem die Fähigkeit eines Services angegeben wird, über einen definierten Zeitraum funktionsfähig zu sein (Angabe in Stunden: h). Diese Qualitätsmerkmale basieren auf ermittelten Werten

bezüglich der mittleren Zeit eines Fehlereintritts (englisch Mean Time to Failure) und der mittleren Reparaturzeit (englisch Mean Time to Repair). Die genauen Metriken sind in [134] definiert und basieren auf [6, 13].

**Kosten** Bei der Kostenkalkulation wurde folgender Weg gewählt: Die Qualitätsmerkmale unterscheiden zwischen den Kosten, die für die tatsächliche Leistungserbringung anfallen (Nutzungskosten) und den Kosten, die durch die Bereitstellung zur Nutzung des Services entstehen (Fixkosten). Als Maßeinheit wurde Euro (€) vereinbart und eine Metrik wurde spezifiziert. Die Gesamtkosten ergeben sich dann aus der aggregierten Summe der Fixkosten und dem Produkt aus Anzahl der Service-Aufrufe und den Nutzungskosten. Dazu wurden also durch Anwendung der Aggregationsmuster die Fixkosten und Nutzungskosten aufsummiert. Anschließend wurde die Anzahl der Service-Aufrufe für einen Zeitraum ermittelt, mit den Nutzungskosten multipliziert und so die Gesamtkosten ermittelt.

**Domänen und Skalenniveau** Die Domänen sind in Anlehnung an einen Programmiersprachen-Datentyp als *double* definiert, also eine rational skalierte Zahl. Es existieren weitere Einschränkungen, wie beispielsweise Verfügbarkeit zwischen 0 und 100 und Kosten als positiver Wert. Da das Projekt auf die Berechnung der aggregierten Informationen über den Prozess fokussierte, wurden keine weiteren Qualitätsmerkmale betrachtet, so dass nur diese relativ einfachen Domänen und Skalenniveaus zum Einsatz kamen.

### 5.6.2 Analyse des Geschäftsprozesses

Um den Order-To-Cash-Prozess bezüglich seiner Qualitätsmerkmale analysieren zu können, wurden die vier Teilprozesse modelliert, die anzuwendenden Aggregationsmuster spezifiziert und die möglichen Kombinationen der Service-Auswahl ermittelt.

**Modellierung der Teilprozesse** Abbildung 5.11 zeigt exemplarisch den Teilprozess Auftragsbearbeitung. Der Teilprozess besteht aus einer Sequenz von `createOrder` bis `sendOrderConfirmation`. Je nach Zahlungsmethode wird entweder die Kreditkarte oder das Konto geprüft (XOR-Block mit `checkCreditCard` und `checkAccount`). Nach der Überprüfung der Adresse (`verifyAddress`) wird parallel der Kreditrahmen und die Verschuldung des Kunden überprüft (AND-Block mit `checkCreditLimit` und `checkIndebtness`). Darauf folgt die Überprüfung der Verfügbarkeit für jedes der benötigten Materialien (Loop-Block um `checkAvailability`) und anschließend werden diese Materialien reserviert (Loop-Block um `allocateStock`). Zum Abschluss des Teilprozesses wird die Bestätigung des Auftrages an den Kunden verschickt (`sendOrderConfirmation`). Im gesamten Order-To-Cash-Prozess wurden die Kontrollflussblöcke Sequenz, XOR, Loop und AND verwendet.

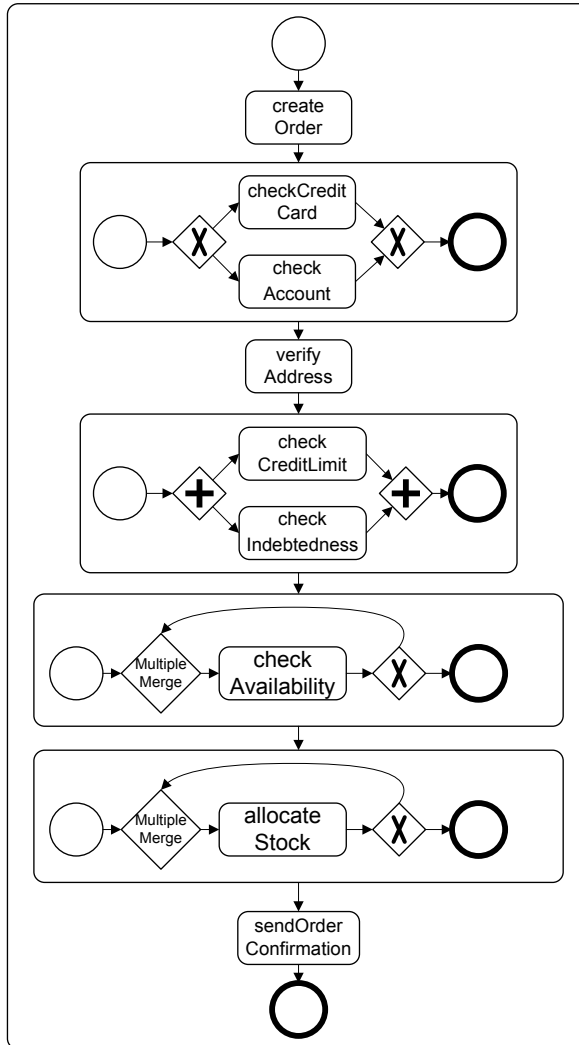


Abbildung 5.11: Teilprozess Auftragsbearbeitung des Order-To-Cash-Prozesses, dargestellt in BPMN

Qualitätsmerkmal	Sequenz	XOR	Loop	AND
Anwortzeit	Summe	Erwart.-wert	Summe	Maximum
Durchsatz	Minimum	Minimum	Übernehmen	Minimum
Verfügbarkeit	Produkt	Erwart.-wert	Produkt	Produkt
Zuverlässigkeit	Minimum	Minimum	Übernehmen	Minimum
Fixkosten	(Summe)	(Summe)	(Übernehmen)	(Summe)
Nutzungskosten	Summe	Erwart.-wert	Summe	Summe

Tabelle 5.2: Aggregationsmuster für die Qualitätsmerkmale je Kontrollflussblock

Qualitätsmerkmal	Maßeinheit	I	II	III	IV
Anwortzeit	ms	1085	1085	1085	1085
Durchsatz	tps	85,94	85,94	84	84
Verfügbarkeit	%	89,32	90,78	89,32	90,78
Zuverlässigkeit	h	1000	1000	1000	1000
Fixkosten	€	15.700	17.300	17.300	16.300
Nutzungskosten	€	0,067	0,065	0,065	0,063
Gesamtkosten	€	884.020	859.700	859.700	832.780

Tabelle 5.3: Aggregierte Ausprägungen der Qualitätsmerkmale für den Order-To-Cash-Prozess für die Service-Kombinationen I-IV (mittlere Auslastung: 5 tps, Laufzeit: 1 Monat) [134]

### 5.6.3 Modellierung der Aggregation

Weiterhin ist gezeigt worden, dass unter Verwendung der Aggregationsmuster die Qualitätsmerkmale aggregiert werden konnten. Dazu sind für jede Art Kontrollflussblock und für jedes Qualitätsmerkmal die zu verwendenden Aggregationsmuster bestimmt worden. Tabelle 5.2 zeigt die Zuordnung von Qualitätsmerkmal und Kontrollflussblock zu Aggregationsmuster. Bei den Fixkosten ist zusätzlich zu beachten, dass jeder Service genau einmal bei der Aggregation berücksichtigt werden darf und dass die Fixkosten anfallen, sobald der Service zur Verfügung gestellt werden muss. Sie sind also auch zu entrichten, falls der Service auf einem alternativen Pfad liegt, der nicht in jeder Prozessinstanz betreten wird. Somit konnte für den Order-To-Cash-Prozess die Aggregation durchgeführt werden.

### 5.6.4 Ermittlung der gültigen Service-Auswahl

Für die Prozessschritte waren teilweise mehrere Services verfügbar. Die alternativen Services unterschieden sich in den Ausprägungen der Qualitätsmerkmale. Es ergaben sich verschiedene Kombinationen, bei denen Services für die jeweiligen Prozessschritte ausgewählt wurden. Unter Anwendung der Aggregationsmuster wurde für jede Kombination die Ausprägungen der Qualitätsmerkmale über den gesamten Prozess aggregiert. Im nächsten Schritt wurden die Werte einzelner Qualitätsmerkmale eingeschränkt, beispielsweise mittlere Antwortzeit von unter 1200 ms und mittlerer Durchsatz von mindestens 80 tps. Dadurch wurde die Anzahl gültiger Kombinationen mit jeweils verschiedener Service-Auswahl auf vier eingeschränkt. Die aggregierten Qualitätsmerkmale dieser vier Service-Kombinationen sind in Tabelle 5.3 dargestellt.

## 5.7 Zusammenfassung

In diesem Kapitel ist das Qualitätsmodell im Kontext von Geschäftsprozessen untersucht worden. Dazu wurde die Service-Auswahl untersucht, die Aggregation



von Qualitätsmerkmalen über einen gesamten Geschäftsprozess anhand von Aggregationsmustern aufgezeigt und das Qualitätsmodell schließlich in einem praktischen Einsatz validiert.

Zunächst wurde untersucht, welche Vorteile sich für die Service-Auswahl realisieren lassen. Durch ein Qualitätsmodell wird die Vergleichbarkeit der Services ermöglicht, so dass eine Rangfolge erstellt werden kann, welche die individuellen Anforderungen des Requestors berücksichtigt. Dabei werden, je nach Variante der Service-Auswahl, möglichst aktuelle Informationen über die Services bei der Auswahl des besten Services berücksichtigt. Zusätzlich können aus der Rangfolge alternative Services für unvorhergesehene Fehlerfälle definiert werden.

Es ist gezeigt worden wie das Qualitätsmodells erweiterte Möglichkeiten zur Optimierung der Geschäftsprozesse ermöglicht: Individuelle SLA's erlauben eine Optimierung speziell für eine Prozessdefinition. Diese Optimierung kann dann für alle Instanzen der Prozessdefinition genutzt werden. Die Berücksichtigung individueller Daten und Restriktionen, die beim Start einer Prozessinstanz bekannt sind, ermöglicht andererseits eine individuelle Prozessoptimierung. Darüber hinaus kann der Zustand der aktuellen Prozessinstanz bei der Service-Auswahl berücksichtigt werden.

Als weiterer Aspekt dieses Kapitels wurden Aggregationsmuster herausgearbeitet, die eine Informationsverdichtung der Qualitätsmerkmale eines Prozesses erlauben. Es wurde untersucht, welche Aggregationen wann möglich sind. Dazu wurden Anforderungen an die Qualitätsmerkmale und den Kontrollfluss spezifiziert und insbesondere Aspekte der Metrik eines Qualitätsmerkmals untersucht. Weiterhin wurden Beispiele für Qualitätsmerkmale aufgezeigt und die Verwendung der Aggregationsmuster illustriert.

Die vorgestellten Aggregationsmuster wurden anhand eines allgemeinen Modells blockstrukturierter Prozessbeschreibungssprachen definiert und die rekursive Anwendung entlang der Hierarchie der Prozessdefinition skizziert. Die vorgestellten Aggregationsmuster erheben nicht den Anspruch auf Vollständigkeit sondern können erweitert werden: Eine andere Prozessbeschreibungssprachen und die dort verfügbaren Kontrollfluss-Konstrukte können zu weiteren sinnvollen Aggregationsmustern führen oder Anpassungen nötig machen. Ebenso können bei speziellen Qualitätsmerkmalen andere Aggregationen sinnvoll sein. Das ist Abschnitt 5.4.9 spezifizierte Aggregationsmuster zur Aggregation mehrerer Qualitätsmerkmale kann dabei als Ausgangsbasis dienen, da es allgemein die Anwendung einer zu definierenden Funktion  $f$  zur Aggregation beschreibt.

Abschließend wurde das Modell durch einen praktischen Einsatz validiert. Es ist dabei aufgezeigt worden, wie das Qualitätsmodell angewendet wurde um Qualitätsmerkmale zu definieren. Außerdem wurde eine praktische Anwendung der Aggregationsmuster auf die definierten Qualitätsmerkmale geschildert, die für einen Geschäftsprozess in einer serviceorientierten Umgebung durchgeführt wurde. Es konnten qualitative Aussagen über den Geschäftsprozess getroffen werden und die Auswirkungen einer variierenden Service-Auswahl aufgezeigt werden.

# Kapitel 6

## Verwandte Arbeiten

Die verwandten Arbeiten zum Thema Modellierung von Qualitätsmerkmalen werden in diesem Kapitel betrachtet. Der Begriff Qualität wird in vielen verschiedenen Bereichen verwendet und hat dabei oft eine unterschiedliche Bedeutung. Bei der Formulierung des Qualitätsmodells wurde in Abschnitt 4.3.1 der Bezug zum Qualitätsbegriff in der Wirtschaftslehre und im EN/ISO-Standard EN ISO 9000:2000 dargestellt, sowie die Bedeutung für das Qualitätsmodell dieser Arbeit herausgearbeitet. Hier fand auch eine Abgrenzung zum Begriff nicht-funktionale Eigenschaften statt.

Zu Beginn findet in Abschnitt 6.1 eine Einordnung des Qualitätsmodells dieser Arbeit statt. Wie bereits vorher angesprochen, stammen die Begriffe Quality of Services (QoS) und Service Level Agreement (SLA) aus der Kommunikationstechnik und werden auch in der Software-Entwicklung verwendet. Abschnitt 6.2 schildert die Bedeutung der Begriffe in diesen Kontexten. Im Abschnitt 6.3 werden Qualitätsmodelle für serviceorientierte Architekturen vorgestellt und untersucht. Oftmals werden explizit die relevanten Qualitätsmerkmale für den jeweiligen Zweck passend definiert. Daher findet für jeden Ansatz eine Einordnung statt, sowie eine Abgrenzung zum Qualitätsmodell dieser Arbeit. Anschließend werden in Abschnitt 6.4 die verwandten Arbeiten im Bereich der Geschäftsprozesse betrachtet, die Qualitätsmerkmale berücksichtigen. Zum Abschluss werden in Abschnitt 6.5 Ansätze für die automatisierte Optimierung von Geschäftsprozessen vorgestellt und untersucht.

### 6.1 Einordnung des Qualitätsmodells

Das Qualitätsmodell wurde bereits in Abschnitt 1.3 als Meta-Modell eingeordnet und wurde als ein Modell der Eigenschaftsbeschreibungen von Services beschrieben. In diesem Abschnitt soll das Qualitätsmodell im Kontext zweier umfassender Ansätze zur Beschreibung von serviceorientierten Architekturen eingeordnet werden. Außerdem wird eine andere Bedeutung des Begriffs Quality of Services im Kontext von Web Services aufgezeigt.

### 6.1.1 Business-Oriented-Management

Zur Einordnung des Qualitätsmodells dieser Arbeit wird im Folgenden ein Ansatz von Fabio Casati et alii verwendet, der in [32] vorgestellt wird. Dort wird argumentiert, dass es für das Management von Services drei Klassifikationsebenen gibt und auf jeder Ebene werden zentrale Aufgaben und das Ziel des Managements beschrieben.

Auf der untersten Ebene steht das *Infrastruktur-Management*, dass die Ausführungsperformanz beispielsweise auf der Web Services Plattform betrachtet und sicherstellt. Auf der nächsten Ebene steht das *Management der Anwendungen*, also der Services selbst, die aus Benutzersicht überwacht und ausgewertet werden. Das letztendliche Ziel des Service-Managements stellt die oberste Klassifikationsebene dar, das *Business-Oriented-Management*. Hier werden den Beobachtungen und Auswertungen geschäftlich relevante Aussagen zugeordnet. Es wird berücksichtigt, dass ein Service-Aufruf Teil eines Geschäftsprozesses oder Teil einer Konversation zwischen Geschäftspartnern sein kann. Als Beispiel wird genannt, dass eine durch die unteren Ebenen bestimmte Ausführungsdauer dann als akzeptabel angesehen werden kann, wenn sie unter 30 Sekunden liegt.

**Einordnung des Qualitätsmodells** Das Qualitätsmodell dieser Arbeit hat in Bezug auf die Klassifikationsebenen das ebenfalls Ziel, Aussagen auf der Ebene des Business-Oriented-Management treffen zu können. Es wurde ein Ansatz erarbeitet, der es ermöglicht, Qualitätsmerkmale und insbesondere Aspekte deren Metriken zu definieren. Die Metrik spezifiziert, wie beispielsweise mit Messverfahren beobachtbare Ereignisse auf der Ebene der Kommunikation und der Ausführungsperformanz Werten zugeordnet werden können. Dieses betrifft auch das Infrastruktur-Management. Es wurde in dieser Arbeit weiterhin die Benutzersicht berücksichtigt, beispielsweise indem Anfragen mit Restriktionen, Optimierungszielen und Nutzenfunktionen berücksichtigt wurden. Diese Aspekte lassen sich dem Management der Anwendungen zuordnen. Schließlich wurde aufgezeigt, wie Aggregationsmuster für einen Geschäftsprozess angewendet werden können, um geschäftlich relevante Aussagen treffen zu können. Es wurde aufgezeigt, wie die Service-Auswahl bei der Modellierung von Geschäftsprozessen Qualitätsmerkmale berücksichtigen kann. Somit wird durch das Qualitätsmodell eine konzeptionelle Grundlage gebildet, die die wesentlichen Aufgaben der allen Management-Ebenen unterstützt.

### 6.1.2 Extended SOA (xSOA)

Von Michael P. Papazoglou wird in [115] eine Erweiterung der serviceorientierten Architektur unter der Bezeichnung Extended SOA (xSOA) vorgeschlagen. Hierbei wird eine drei-Schichten-Architektur konzipiert und jeweils die Elemente und Operationen der Ebenen beschrieben.

Auf der untersten Ebene sind die *Basic Services* angesiedelt. Hier wird neben der Service-Beschreibung und der Schnittstelle auch die Qualität des Services

spezifiziert. Auf der nächsthöheren Ebene befinden sich die *Composite Services*, also unter anderem aus Basic Services zusammengesetzte Services, deren komplexe Ausführung gesteuert und überwacht wird. Hier sind aggregierte Qualitätsmerkmale für einen zusammengesetzten Service als *QoS Composition* spezifiziert. Es wird dazu angemerkt, dass diese Qualitätsmerkmale aus den Qualitätsmerkmalen der Basic Services anhand geeigneter Verfahren errechnet werden muss. Auf der höchsten Ebene findet das Management der Services statt, wobei unter anderem SLA's verwaltet und überwacht werden. Hier sind auch die Messverfahren für Qualitätsmerkmale sowie Verfahren zur Überwachung spezifiziert.

**Einordnung des Qualitätsmodells** Das Qualitätsmodell dieser Arbeit kann im Kontext der xSOA eingesetzt werden, um die Qualitätsmerkmale auf der untersten Ebene für Basic Services zu definieren. Dann können die Ausprägungen für die Basic Services spezifiziert werden. Die Aggregationsmuster können genutzt werden, um die QoS Composition zu errechnen. Aus der spezifizierten Metrik der Basic Services und der Composite Services können dann die Messverfahren in der Management-Schicht verwendet werden. Außerdem beschreibt das Qualitätsmodell auch Service Level Agreements für die Management-Schicht. Das Qualitätsmodell dieser Arbeit stellt somit eine konzeptionelle Grundlage für die qualitätsbezogenen Elemente der xSOA zur Verfügung und unterstützt die dazugehörigen Aufgaben.

### 6.1.3 Quality of Services als Web Service Schicht

Eine von dieser Arbeit stark abweichende Bedeutung eines Qualitätsmodells ist im Kontext von Web Services zu finden. Es gibt Ansätze, die Quality of Services als separate Schicht in einer SOA ansehen. In der Regel bedeutet dies, dass zusätzliche Funktionalität wie Transaktionalität und sicherer beziehungsweise verschlüsselter Nachrichtenaustausch durch eigenständige Software-Komponenten realisiert werden. Diese zusätzliche Funktionalität ist in einer separaten Schicht angesiedelt, die von allen Web Services genutzt werden kann.

Ein Modell für die Entwicklung von serviceorientierten Architekturen mit der Sichtweise einer QoS-Schicht wird von Ali Arsanjani in [7] vorgestellt. Aus Sicht eines Web Service Providers, wird die Service-Beschreibung als zentraler Bestandteil angesehen. Die in [7] vorgestellte Vorgehensweise definiert ein Schichtenmodell für die SOA, wobei ausgehend von den operativen Systemen so genannte *Enterprise Components* definiert werden. Diese Komponenten bieten jeweils einen oder mehrere Services an. Quality of Services wird dort als eigene Schicht angesehen, die orthogonal zu den Schichten auch das Monitoring ermöglicht und somit eine Grundvoraussetzung für das Management der Qualitätsmerkmale bereitstellt. Ziel hierbei ist es aus Provider-Sicht zu garantieren, dass die angebotene Qualität eingehalten werden kann. Als Beispiele für Qualitätsmerkmale werden hier *security*, *performance* und *availability* genannt.

## 6.2 Rechnernetze und Software-Entwicklung

Der Begriff Quality of Service und die Abkürzung QoS stammen aus dem Bereich der Rechnernetze und von Telekommunikationsanbietern [146]. Daher wird im Folgenden dieser Ursprung betrachtet und die Bedeutung für das Qualitätsmodell untersucht. Anschließend werden Qualitätsmodelle in der Software-Entwicklung betrachtet, wo nicht-funktionale Eigenschaften spezifiziert werden können, die den Qualitätsmerkmalen dieser Arbeit ähnlich sind. Aus der Vielzahl der Ansätze in diesem Bereich werden zwei ausgewählte Ansätze betrachtet.

### 6.2.1 QoS und SLA in Rechnernetzen

In Kontext der Rechnernetze werden Eigenschaften wie Kommunikationsverzögerung, Kapazität, Verfügbarkeit, Bandbreite und Fehlerrate unter dem Begriff Quality of Service zusammengefasst [146]. Hier ist der Service die Kommunikation über ein Medium und die Qualität beschreibt die Eigenschaften dieser Kommunikation. Es existieren aber auch abweichende Auffassungen, die beispielsweise QoS als Bereitstellung einer garantierten Kommunikationsqualität von einem Anbieter betrachtet oder Fähigkeit eines Datennetzwerkes eine bestimmte Qualität automatisiert garantieren zu können. Ein Überblick über Theorien und Modelle zum Thema QoS bezogen auf das Kommunikationsmedium Internet findet sich in [48].

Weiterhin wird mit QoS auch die Priorisierung von IP-Datenpaketen anhand bestimmter Merkmale bezeichnet. Mit diesen Mechanismen ist es möglich einen verzögerungsfreien und kontinuierlichen Datenstrom, wie er beispielsweise für Voice-over-IP benötigt wird, stärker zu bevorzugen als das Herunterladen von einem Dateiserver oder der Aufruf von Webseiten. Dazu wird diese Information im Kopf der Datenpakete vermerkt, so dass die Datenpakete priorisiert behandelt werden können.

In der Kommunikationstechnik wird ebenfalls der Begriff Service Level Agreement benutzt. Bei einem Aufbau einer Kommunikationsverbindung handelt der Kunde mit dem Anbieter eine garantierte Qualitätsniveau (QoS-Klasse) aus. Die Garantien werden in einem Service Level Agreement dokumentiert. Es wird eine vertragliche Dokumentation über garantierte Qualität beschrieben, somit ist hier ist die Bedeutung sehr ähnlich zur Auffassung eines Vertrages in serviceorientierten Architekturen [131, 93] und somit auch zur Bedeutung im Qualitätsmodell dieser Arbeit. Allerdings ist Funktionalität des Services in diesem Fall immer dieselbe, nämlich die Kommunikation.

### 6.2.2 Qualitätsmerkmale in der Software-Entwicklung

Bei der Entwicklung von Software werden Qualitätsmerkmale vor allem während der Analyse- und Design-Phase definiert, so dass überprüft werden kann ob die entwickelte Software den Anforderungen entspricht [26, 56, 85]. Unter dem Begriff Quality of Service werden hier beispielsweise Eigenschaften wie Änderbar-

keit, Wiederverwendbarkeit und Interoperabilität verstanden. Diese Eigenschaften werde in den Modellierungssprachen oftmals durch natürlichsprachliche Kommentare oder informale Beschreibungen angegeben [106]. Als Synonym für QoS wird in der Software-Entwicklung auch nicht-funktionale Eigenschaft verwendet (siehe auch Abschnitt 4.3.1). Der Fokus liegt dann auf einer klaren Trennung der Qualität zur Funktionalität der Software: Der „Service“ einer Software wird dann als Realisierung der Funktionalität angesehen und die Quality of Service beschreibt, wie (in welcher Qualität) diese Funktionalität realisiert wird.

Neben der Berücksichtigung von Qualitätsmerkmalen während der Software-Entwicklung und der Überprüfung ihrer Einhaltung nach Beendigung der Entwicklung, können Qualitätsaspekte auch zur Laufzeit relevant sein. Als Beispiel für verwandte Arbeiten in diesem Kontext sei auf die grundlegende Taxonomie für Qualitätsmerkmale von Software in verteilten Umgebungen von Bikash Sabata et alii in [130] verwiesen. Ziel der Taxonomie ist es, grundlegende Regeln für die Ressourcen-Verteilung in verteilten Systemen aufzustellen, so dass spezifizierte Ausprägungen von Qualitätsmerkmale eingehalten werden. Dazu wird eine festgelegte Menge von relevanten Qualitätsmerkmalen definiert und unterschiedliche Perspektiven berücksichtigt: die Systemperspektive, die Anwendungsperspektive und die Ressourcenperspektive. Die Anwendung der Regeln der Taxonomie führt dann zur optimierten Ressourcen-Verteilung im System. Dieses allgemeine Ziel eines Qualitätsmodells wurde in Abschnitt 4.2.2 im Rahmen des Service Management beschrieben. In einer serviceorientierten Architektur ist die Einhaltung der Qualität das Ziel des Providers. In dieser Arbeit wurde das interne Ressourcen-Management eines Providers nicht näher betrachtet, sondern davon ausgegangen, dass ein Provider seine Services zu der angebotenen Qualität realisieren kann.

Im Folgenden werden zwei umfassende Ansätze für die Modellierung von Qualitätsmerkmalen in der Software-Entwicklung vorgestellt. Dabei wird insbesondere der Zusammenhang mit dem Qualitätsmodell dieser Arbeit aufgezeigt. Es wird dabei gezeigt, dass zum Teil ähnliche Konzepte in den Qualitätsmodellen existieren, aber der Fokus und der Zweck der Modelle unterschiedlich sind.

### **Die Quality Modeling Language (QML)**

Eine Sprache zur Modellierung von Qualitätsmerkmalen von Software-Komponenten wird von Svend Frølund und Jari Koistinen in [52] mit der Quality Modeling Language (QML) vorgestellt. Hierbei werden Qualitätsmerkmale als nicht-funktionalen Eigenschaften einer Software-Komponente beschrieben, die während der Design-Phase der Software-Entwicklung zu definieren sind. Ziel einer Qualitätsspezifikation in QML ist es, durch explizite Spezifikation von Qualitätsmerkmalen während der Design-Phase qualitativ bessere Software-Komponenten zu realisieren. QML definiert dazu drei Abstraktionsebenen:

1. Ein *Contract Type* definiert einen Typen, der spezifische Aspekte repräsentiert, beispielsweise Performanz oder Zuverlässigkeit. Ein Contract Type de-

finiert die *Dimension*, das heißt die (möglicherweise geordneten) Werte. Dabei gibt es unterschiedliche Arten von Domänen: Mengen, Aufzählungen und numerische Domänen.

2. Ein *Contract* ist eine Instanz eines Contract Types und stellt somit eine konkrete QoS Spezifikation dar. Dabei können für die Dimensionen eines Contract Types Einschränkungen (*Constraints*) definiert werden.
3. Ein *QML Profile* verknüpft einen Contract mit Schnittstellen, Methoden und Ein- und Ausgabeparametern einer Komponente.

QML sieht vor, dass bei geordneten Werten angegeben werden kann, ob ein kleinerer Wert besser ist oder ein grösserer Wert; diese geschieht durch die Schlüsselwörter *decreasing* und *increasing*. Bei der Spezifikation eines Contracts können auch genauere Vorgaben für mögliche Werte definiert werden, wie zum Beispiel Minima oder Maxima. Neben einfachen Einschränkungen lassen sich auch so genannte *Dimension Aspects* definieren: Mittelwert und Varianz, Häufigkeit oder Quantile. Bei der Spezifikation eines Contracts mit QML können also auch Quantile (*Percentile*) angegeben werden: *percentile 80 < 6* bedeutet, dass das 80. Quantil für diese Dimension kleiner als 6 sein muss.

Die Sprache QML ist nicht bezogen auf ein spezielles Anwendungsgebiet und spezifiziert auch keine allgemeinen Qualitätsmerkmale. Ursprünglich wurde QML zur Spezifikation von QoS Eigenschaften für die Design-Phase einer Softwareentwicklung erstellt, aber es wird bereits in [52] darauf hingewiesen, dass man QML-Spezifikationen auch dynamisch, beispielsweise für Verhandlungen benutzen kann.

Im Vergleich zu dem Qualitätsmodell dieser Arbeit ist erkennbar, dass in QML Typen für Qualitätsmerkmale definiert werden und Wertebereiche definiert werden. Diese ähneln den Qualitätsmerkmalen für Service-Typen, allerdings fehlt die Spezifikation des Skalenniveaus und die Spezifikation einer Metrik ist nur in Ansätzen beschrieben. Qualitätsmerkmale können für mehrere Service-Typen benutzt werden und stellen daher ein mit den Contract-Types vergleichbares Konzept dar. Zu den Qualitätsmerkmalen wurden ebenfalls das Konzept einer Richtung der Qualitätssteigerung und die Wertart als Bestandteile der Metrik diskutiert.

Dennoch ist QML für die Entwicklung von Software-Systemen mit verteilten Objekten konzipiert worden. Eine Übertragung auf serviceorientierte Architekturen ist noch nicht geschehen. Es wird weiterhin in [51] vorgeschlagen, die Qualitätsmerkmalen mit einer Definition einer Komponenten-Schnittstelle in IDL zu verknüpfen. Ob sich dieses auf Services und beispielsweise die WSDL übertragen lässt, ist noch zu überprüfen. QML betrachtet außerdem nicht, wie Qualitätsmerkmale aggregiert werden können, wie es in Kapitel 5 beschrieben wurde.

### **Das UML-Profil für QoS und Fehlertoleranz**

Die Object Management Group [104] hat ein UML-Profil für die Spezifikation von Qualitätseigenschaften veröffentlicht [106]. Es wird ein Framework für Qualitätsmerkmale definiert, das bei Software-Entwicklung eingesetzt werden kann.

In dem UML-Profil werden einige Aspekte von Qualitätsmerkmalen definiert. Unter anderem sind Einheiten definierbar, eine Richtung der Qualitätssteigerung und statistische Aussagen wie Mittelwert, Verteilung, Minimum und Maximum. Die zentrale Entität des Profils ist *QoS Characteristics* die aus *QoS Dimension* und *QoS Parameter* besteht und ist eventuell einer *QoS Category* zugeordnet ist. Die Kategorie dient lediglich zur Gruppierung der QoS Characteristics. Weiterhin wird ein Katalog spezifiziert, der zwölf generelle Kategorien für Qualitätsmerkmale vordefiniert, die für unterschiedliche Anwendungsgebiete zutreffend sind.

Das UML Profil definiert ein Framework für Qualitätsmerkmale zur Modellierung von Software-Systemen. Nicht behandelt werden die Skalen der Qualitätsmerkmale, die Bedeutung einer Metrik und die Auswirkungen bei der Aggregation zu größeren Systemen. Es dient daher eher einer Ordnung und Gruppierung von Qualitätsmerkmalen und gibt dafür einen allgemeinen Rahmen vor. Weiterhin ist dieses Framework auf Objektorientierung bezogen und berücksichtigt die besonderen Anforderungen der Service-Orientierung nicht.

### **Zusammenfassung**

Generell unterscheidet sich die Auffassung von QoS-Modelle für die Software-Entwicklung in entscheidenden Aspekten von dem Qualitätsmodell für Services dieser Arbeit. Sie fokussieren auf die Entwicklung neuer Software-Komponenten und werden in Phasen der Analyse und des Designs spezifiziert. Ist die Software entwickelt und die Einhaltung der Qualitätsmerkmale überprüft, werden die spezifizierten Merkmale zunächst nicht weiter beachtet. Die Suche und Auswahl einer Software-Komponente sowie potentielle Verhandlungen über die Qualität werden in den QoS-Modellen der Software-Entwicklung in der Regel nicht betrachtet. Außerdem sollen die Software-Komponenten letztendlich aus Eingaben (Input) unter Verwendung von Ressourcen eine Ausgabe erzeugen. Die Modellierung von QoS-Merkmalen beschreibt dabei vor allem das interne Verhalten der Software und die Verwendung der Ressourcen in Abhängigkeit von der Eingabe [106].

Wie in dieser Arbeit beschrieben wurde, ist in der Service-Orientierung demgegenüber vor allem das vom Requestor beobachtbare Verhalten zu spezifizieren. Ebenso ist es das Ziel eines Providers, seine angebotenen Services gegen die der Konkurrenz abzugrenzen. Die veröffentlichten Ausprägungen der Qualitätsmerkmale werden dann bei der Ausführung berücksichtigt: zur Laufzeit muss der Provider durch eine geeignete Lastverteilung unter den einzusetzenden Ressourcen dafür sorgen, dass die garantierte Qualität eingehalten wird (Service-Management). Es gibt also prinzipielle Ähnlichkeiten, aber einen anderen Fokus. Das Qualitätsmodell dieser Arbeit ist insbesondere auf die Service-Auswahl und die Aggregation von Qualitätsmerkmalen für zusammengesetzte Services fokussiert.



## 6.3 Qualitätsmodelle für serviceorientierte Architekturen

Es existiert viele verschiedene Qualitätsmodelle für serviceorientierte Architekturen. Zur Einordnung und zur Abgrenzung der Modelle dieser Arbeit werden in diesem Abschnitt verwandte Arbeiten vorgestellt, die in serviceorientierten Architekturen eine Service-Auswahl anhand von Qualitätsmerkmalen unterstützen.

Vor allem im Kontext von Web Services und den etablierten Web Services Standards (siehe Abschnitt 2.3) existieren eine Reihe von Ansätzen, die Qualität eines Services abzubilden. Hierbei wird in der Regel eine Grammatik spezifiziert, mit der Qualitätsmerkmale mit den Entitäten eines WSDL-Dokumentes verknüpft werden können. Einige dieser Ansätze werden in diesem Abschnitt ebenfalls untersucht.

Einen Überblick über weitere Ansätze zur Spezifikation und zum Management von Qualitätsmerkmalen im Kontext der Web Services Technologie wird von Min Tian et alii in [153] geliefert. Dort werden neben Web Service Level Agreement (WSLA) [93] (siehe Abschnitt 6.3.5) insbesondere UDDI-Erweiterungen betrachtet [187, 138], sowie die Web Service Offering Language (WSOL) [154]. Diese Ansätze werden dann mit dem Ansatz der Autoren verglichen, der in Abschnitt 6.3.2 erläutert wird.

### 6.3.1 Qualitätsmodell im MAIS-Projekt

Ein Vorschlag für einen flexiblen Rahmen für die Abbildung von Qualitätsmerkmalen in serviceorientierten Architekturen wird von Cinzia Cappelletto et alii in [29] vorgeschlagen. Es wird ein Qualitätsmodell [28] entwickelt, das aus dem Forschungsprojekt *Multichannel Adaptive Information Systems*<sup>1</sup> stammt. Das MAIS-Projekt ist ein italienisches Forschungsprojekt mit dem Hauptziel der Entwicklung einer flexiblen serviceorientierten Umgebung, die Änderungen an Anforderungen, Ausführungsumgebungen und Benutzeransprüchen unterstützt.

Insbesondere wird vorgeschlagen, Techniken des Semantik Web zu benutzen, um eine Erweiterung von UDDI-konformen Service Brokern bezüglich *Qualitätsdimensionen* zu erreichen. Eine Qualitätsdimension hat eine Metrik, wird berechnet durch eine Funktion und definiert in einem Level. Das Modell zeigt eine umfassende Betrachtung von Qualitätsmerkmalen, die über die üblichen Qualitätsmerkmale wie Kosten, Ausführungsdauer oder Kommunikation hinausgeht. Es wird ebenso darauf hingewiesen, dass Requestor und Provider dieselbe Definition eindeutig und präzise verstehen müssen und das die Anforderungen an einen Service von einem Requestor abhängen. Beispielsweise wird argumentiert, dass unter Umständen keine Richtung der Qualitätssteigerung vorgegeben werden kann und Obere Grenzen für eine Optimierung existieren können.

**Einordnung und Abgrenzung** Das Qualitätsmodell in [28] ist umfangreicher und präziser als in den vorherigen Beispielen. Insbesondere wird betont, dass Re-

---

<sup>1</sup>Projekt Homepage: <http://www.mais-project.it/>

questor und Provider eine identische Auffassung von der Bedeutung der Qualitätsmerkmale haben müssen und dass die Qualität nicht von den Werten, sondern von den Anforderungen des Requestor abhängen. In diesen Aspekten ist das Qualitätsmodell des MAIS-Projektes ähnlich dem Qualitätsmodell dieser Arbeit. Allerdings werden Skalenniveau und Wertebereich nicht explizit spezifiziert und es wird kein Meta-Modell für Qualitätsmerkmale erarbeitet. Es fokussiert in den angegebenen Qualitätsdimensionen auf technische Aspekte der Kommunikation, die in dem Forschungsprojekt relevant waren.

Darüber hinaus hat das Qualitätsmodell im MAIS-Projekt keinen konkreten Bezug zu einem Service Modell; lediglich die Entität Service ist betrachtet. Es berücksichtigt somit keine Service-Typen. Außerdem ist die Aggregation von Qualitätsmerkmalen nicht betrachtet, da Geschäftsprozesse nicht im Fokus des Projektes liegen.

#### 6.3.2 WS-QoS Framework

Mit dem in [151] vorgestellten WS-QoS Framework lassen sich Qualitätsmerkmale in die Beschreibung von Web Services integrieren lassen. Es wird ein XML Schema definiert, das Eigenschaften in allgemeiner Form definieren lässt, sowie eine Architektur, die einen *Web Service Broker* als zusätzliche Rolle neben einer UDDI-konformen Registry in die Service-orientierte Architektur einführt. [150]

Es wird der Begriff *Quality of Services* (QoS) verwendet und einem Web Service zugeordnet. Die Qualitätsmerkmale beschreiben neben Kosten insbesondere Kommunikationsaspekte und Server-Performanz. Ziel ist es, die Anforderungen aus der Anwendungsebene auf Qualitätsmerkmale der Netzwerk-Schicht zu übertragen, die QoS unterstützt. Das Framework beschreibt also, wie Anforderungen an Web Services formuliert werden können und wie dieses auf der Netzwerk-Schicht abgebildet werden kann. Der Web Service Broker kann dann die Service-Auswahl unterstützen, indem er Informationen aus dem UDDI-konformen Service Broker um Kommunikationsaspekte und Server-Performanz ergänzt. Der Web Service Broker überwacht auch die Netzwerkschicht und sammelt so Informationen über die Web Services. Somit werden vor allem quantifizierbare Merkmale wie Durchsatz, Antwortzeitverhalten und Server-Kapazität betrachtet.

**Einordnung und Abgrenzung** Es wird grundsätzlich erkannt, dass eine informierte Service-Auswahl durch den Benutzer mit bestehender Web Services Technologie nicht getroffen werden kann. Daher wird eine XML-basierte Erweiterung von WSDL vorgestellt, die Qualitätsmerkmale abbilden kann.

Das WS-QoS Service-Modell gruppiert die Web Services über ein *tModel* aus der UDDI-Spezifikation, das jeweils einem WSDL-Interface zugeordnet ist. Dieses ist vergleichbar mit einem Service-Typ aus dem Service-Modell dieser Arbeit, wie auch die in Abschnitt 3.6 beschriebene Umsetzung des Service-Typ-Konzeptes durch Web Services Technologie darstellt. Zur näheren Beschreibung der Metrik eines Qualitätsmerkmals wird eine Ontologie vorgestellt. Dadurch wird

ermöglicht, eigene Qualitätsmerkmale zu spezifizieren, wobei Namen, Beschreibung, Maßeinheit und Richtung als Aspekte modelliert werden können [152].

### 6.3.3 Web Services Selection Framework

Das in [121] vorgestellte Web Services Selection Framework (WebQ) beschäftigt sich mit der Auswahl von Web Services für Geschäftsprozesse. Dabei werden so genannte *QoS Parameter* berücksichtigt, um den besten Web Service für eine Prozessinstanz auszuwählen. Es werden die üblichen Parameter wie die Performanz (Latenzzeit und Durchsatz), Kosten, Zuverlässigkeit und Verfügbarkeit spezifiziert. Hierbei wird jeweils die Bedeutung der QoS Parameter und wie die Werte ermittelt werden vergleichsweise ausführlich beschrieben. Zusätzlich gibt es einen *Task Specific QoS Parameter*, der die Qualität der Ausgabe des Services gewichtet bewertet. Hierdurch soll die Erweiterbarkeit des Ansatzes aufgezeigt werden.

**Einordnung und Abgrenzung** Der WebQ-Ansatz unterscheidet sich konzeptionell nicht wesentlich von den oben genannten Ansätzen. Allerdings untermauert der WebQ-Ansatz die Notwendigkeit einer präzisen und eindeutigen Spezifikation von Qualitätsmerkmalen. Dennoch wird auch hier kein Meta-Modell entwickelt, dass für andere Qualitätsmerkmale benutzt werden kann.

Weiterhin ist bemerkenswert, dass in WebQ vorgeschlagen wird, nicht nur den besten Service zu ermitteln, sondern die  $m$  besten Services. Dies wird damit begründet, dass in einer hochdynamischen serviceorientierten Umgebung die Auswahl eines einzelnen Services zu unsicher erscheint. In Abschnitt 5.2.2 wurde im Kontext der Geschäftsprozessmodellierung ebenfalls erwähnt, dass alternative Services aus der Rangfolge aufgerufen werden können, falls der beste Service fehlschlägt.

### 6.3.4 OASIS Web Services Quality Model (WSQM)

Das Spezifikationsgremium OASIS, das unter anderem auch WS-BPEL und UDDI verwaltet, hat ein technisches Komitee gebildet, um ein Web Services Quality Model (WSQM) zu spezifizieren. Ziel ist es, ein Qualitätsmodell zu spezifizieren, der Teilnehmern einer SOA den Kontext für Verträge über die Qualitätseigenschaften eines Services bereitstellt. Dadurch soll es möglich sein, dass die Teilnehmer die Qualität für eine Web Service Ausführung definieren können. Das technische Komitee schlägt dazu momentan einen Entwurf für ein Qualitätsmodell vor [78], um anschließend eine XML Spezifikation (Web Services Quality Description Language, WS-QDL) zu erzeugen. Das vorgestellte Qualitätsmodell definiert drei Hauptbestandteile, die hier untersucht und diskutiert werden sollen.

**Qualitätsfaktoren** Qualitätsmerkmale werden in WSQM Qualitätsfaktoren genannt und für sie wird eine Charakterisierung vorgeschlagen, die eine Schichtenstruktur bildet. Auf der obersten Ebene stehen die vom Benutzer wahrgenommenen

*Business Values.* Auf der mittleren Ebene sind die messbaren Qualitätsfaktoren wie beispielsweise Stabilität, Skalierbarkeit und Antwortzeit. Hier wird relativ detailliert vorgegeben, wie diese Qualitätsfaktoren aussehen sollten und welche Aspekte bei Spezifikation und Messung zu berücksichtigen sind. Der untersten Ebene, der System-Ebene, werden Qualitätsfaktoren der Interoperabilität, Verwaltbarkeit und Sicherheitseigenschaften zugeordnet. Diese Charakterisierung ist je Schicht weiter unterteilt und umfasst vor allem technische Aspekte der Kommunikation. Allerdings werden auch Aspekte wie Transaktionalität und Sicherheit angesprochen und charakterisiert.

**Qualitätsaktivitäten** Die Aktivitäten um einen Vertrag über zugesicherte Qualität zu erreichen (also ein SLA), werden als *Quality Activity* definiert. Neben *suchen*, *entwickeln* und *registrieren* werden vor allem Management-Aufgaben und Monitoring-Aspekte betrachtet. Die Sichtweise des WSQM legt hier nahe, dass für jeden Service-Aufruf ein SLA existiert.

**Qualitätspartner** Es werden im WSQM-Modell unterschiedliche Rollen bezüglich ihrer Verbindung zu den Qualitätsfaktoren in einer SOA als *Quality Associate* definiert. Neben den üblichen Rollen des Providers und Requestors (hier bezeichnet als: Consumer), gibt es weitere Rollen. Die Rolle des *Stakeholder* kennzeichnen die Eigentümer des Services<sup>2</sup>. Ein Stakeholder gibt die gewünschten Qualitätsfaktoren und deren Levels vor. Die Qualitätseigenschaften werden vom Entwickler (Developer) umgesetzt. Zur Überwachung (Monitoring) der Qualitätseigenschaften werden im WSQM drei weitere Rollen definiert. Ein *QoS Broker* überwacht die vom Provider angegebenen Qualitätseigenschaften und gibt einem potentiellen Benutzer darüber Auskunft. Ein QoS Broker dient also als neutraler Vermittler und Überwacher der Qualität. Zusätzlich ist die Rolle des *Quality Assurer* definiert, der die Überwachung der vertraglich zwischen Provider und Consumer festgelegten Qualität übernimmt. Der Quality Assurer überwacht also speziell eine Service-Benutzung. Schließlich ist ein *Quality Manager* definiert, der für den Provider das Monitoring übernimmt und die Ressourcen verwaltet, so dass alle vertraglichen Verpflichtungen eingehalten werden können.

Vor allem die Abgrenzung der Rollen zur Überwachung der Qualität ist in [78] schwach. Die Aufgaben und Aktivitäten scheinen sehr ähnlich zu sein und die definierten Aufgaben werden aus unterschiedlichen Blickwinkeln betrachtet, werden aber im Prinzip von einer sehr ähnlichen Rolle vorgenommen. Außerdem leuchtet es nicht ein, warum zwischen Stakeholder, Provider und Quality Manager getrennt werden muss. Die Stakeholder-Rolle ist primär auf die Entwicklung konzentriert, der Provider auf das Angebot eines Services und der Quality Manager auf die Überwachung einer Ausführung. Hier sind eher verschiedene Phasen eines Service-Lebenszyklus betrachtet als unterscheidbare Rollen.

---

<sup>2</sup>Hier ist anzumerken, dass dieses nicht den in der Wirtschaftslehre üblichen Begriff eines Stakeholders als Geschäftsinteressent oder Anspruchsgruppe [53] entspricht.

**Einordnung und Abgrenzung** Obwohl der Working Draft bereits die Versionsnummer 2.0 trägt, ist diese Spezifikation in einem frühen Stadium. Die Definitionen sind nicht eindeutig und teilweise unklar und schwammig abgegrenzt. Unklar bleibt auch der Zweck des Qualitätsmodells. Im Grunde wird eine Charakterisierung der Qualitätsmerkmale vorgeschlagen und die Aufgaben zur Qualitätsverwaltung und -überwachung werden skizziert. Aspekte der Qualitätsmerkmale und wie diese zu definieren sind, werden nicht detailliert betrachtet. Es existiert kein Meta-Modell für Qualitätsmerkmale und somit keine konzeptionelle Grundlage für die Unterstützung der Service-Auswahl.

Letztendlich kann dieses Modell einen interessanten ersten Ansatz liefern, wie die vielen einzelnen Spezifikationen wie zum Beispiel WS-Policy [176] und das WS-I Basic Profile [10] einzuordnen sind. Somit handelt es sich eher um eine Charakterisierung, die zum Verständnis anderer Modelle beiträgt und weniger um einen eigenständigen Ansatz.

### 6.3.5 Web Services Level Agreements (WSLA)

Das Web Service Level Agreement (WSLA) Projekt wird vor allem von IBM unterstützt und beschreibt das Management von Service Level Agreements im Web Services Kontext. Es wird fokussiert auf eine eindeutige Spezifikation eines SLA's und wie dieses durch Provider, Requestor und Dritte überwacht werden kann. Es ist also in WSLA vorgesehen, Dritte zur Überwachung und zur Messung vereinbarter Qualitätsmerkmale einzubinden.

Weitere Ziele der WSLA sind eine einfache Erzeugung von SLA's, indem Vorlagen (*Templates*) spezifiziert werden können und die automatisierte Verhandlung durch eine standardisierte Sprache für SLA's unterstützt wird. Außerdem betrachtet WSLA, wie ein Provider seine Ressourcen einsetzen muss, um eine vereinbarte Qualität einhalten zu können. Dazu sind die Ressourcen explizit vom Provider zu veröffentlichen. Ein Requestor stellt dann seine Anforderungen an einen Service dar und einem Provider ist bekannt welche Ressourcen zu allokalieren sind. Es werden die Schnittstellen beschrieben, um ein derartiges, gemeinsames Agreement zu erreichen und zu spezifizieren. Eine Architektur zur Umsetzung wird mit CREMONA in [92] vorgestellt.

**Einordnung und Abgrenzung** Das Service-Modell von WSLA unterscheidet sich wesentlich vom Service-Modell dieser Arbeit. Durch WSLA kann ein Requestor Anforderungen an einen Service so spezifizieren, dass die Ressourcen eingesetzt werden, die die erforderlichen Eigenschaften aufweisen. Dadurch wird die zweistufige Abstraktion, die als elementarer Bestandteil des Service-Modells dieser Arbeit herausgearbeitet wurde (siehe Abschnitt 3.3), teilweise aufgehoben. Das Qualitätsmodell dieser Arbeit basiert auf der Annahme, dass die Ressourcen-Verwaltung dem Provider obliegt und dass die zu verwendenden Ressourcen nicht vom Requestor bestimmt werden können. Allerdings ist zu bedenken, dass der Ressourcen-Begriff im Web Services Kontext abweichend definiert ist.

WSLA zeigt einen Weg, wie Service Level Agreements in einer Web Services Technologie umgesetzt werden können. Es wird eine Prototypische Implementierung vorgestellt und eine XML-Spezifikation zur Erstellung von SLA's entwickelt. Dabei werden vertragliche Elemente identifiziert, die über die Betrachtung dieser Arbeit hinausgehen und diesbezüglich somit einen wertvollen Beitrag zur Ergänzung des Qualitätsmodells dieser Arbeit leisten können.

### 6.3.6 Das WS-Policy Framework

Das Web Services Policy Framework ist eine Sammlung von Spezifikationen für XML-Dokumente, die von mehreren Software-Herstellern spezifiziert und verwaltet werden. Hauptsächlich ist es das Ziel mit WS-Policy einen Rahmen zu schaffen, wie bestimmte Eigenschaften spezifiziert und einem Web Service zugewiesen werden können.

Eine einzelne *Policy Assertion* stellt eine solche Eigenschaft dar. Mehrere *Policy Assertions* können zu einer *Policy* zusammengefasst werden. Eine *Policy* kann dann einem Web Service zugewiesen werden. Ebenso kann ein Requestor angeben, welche *Policies* ein gesuchter Web Service erfüllen muss. WS-Policy ist dabei eine allgemeine Spezifikation zur Definition solcher Eigenschaften, die dann von weiteren Standards jeweils für ihren Zweck eingesetzt werden können. In diese Sinne stellt WS-Policy also die konzeptionelle Grundlage für verschiedene Arten von Eigenschaftsbeschreibungen dar. Auf WS-Policy aufbauend sind beispielsweise *WS-SecurityPolicy*, *WS-ReliableMessaging* und *WS-SecureConversation* spezifiziert worden, die auf einen sicheren und zuverlässigen Nachrichten-Austausch fokussieren. Im Rahmen der WS-Policy-Spezifikation wird als Beispiel für eine mögliche Verwendung auch die Formulierung von *QoS Characteristics* angegeben, allerdings werden diese nicht genauer beschrieben.

Die Spezifikation *WS-PolicyAttachment* ist Teil des WS-Policy Frameworks und beschreibt explizit, wie eine *Policy* mit Elementen eines WSDL-Dokumentes verknüpft werden können und wie sie von einem UDDI-konformen Service Broker verwaltet werden können. Eine wiederverwendbare *Policy* hat stellt einen standardisierten Namen dar, so dass ein Requestor angeben kann, welche *Policy* er benötigt. Ein das WS-Policy Framework unterstützender Broker kann auf eine Requestor-Anfrage alle Web Services zurückliefern, welche die angeforderten *Policies* einhalten.

**Einordnung und Abgrenzung** Eine mittels WS-Policy beschriebene und einem Service zugewiesene *Policy* kann mit einer veröffentlichten Ausprägung eines Qualitätsmerkmals nach dem Qualitätsmodell dieser Arbeit verglichen werden. Allerdings sind die Semantik der *Policy* selbst und die bei der Formulierung zu beachtenden Aspekte kaum beschrieben. Es wird vielmehr eine XML-basierte Grammatik spezifiziert, die letztendlich genutzt werden kann um *Policies* an WSDL-Elemente zu knüpfen.

Die Möglichkeiten, die ein Requestor hat, um seine Anforderungen anzugeben, sind ebenfalls eingeschränkt; beispielsweise sind Skalenniveaus nicht betrachtet und Nutzenfunktionen nicht vorgesehen. Geschäftsprozesse und Aggregationen werden nicht betrachtet und die Service-Auswahl unter Berücksichtigung von Policies wird nicht genauer erläutert. Dennoch kann die Grammatik des WS-Policy Frameworks genutzt werden, um Qualitätsmerkmale in einer Web Service Umgebung zu formulieren. Somit stellt WS-Policy also eine Grammatik dar, mit der sich die Ausprägungen von Qualitätsmerkmalen so formulieren lassen, dass die Anwendung des Qualitätsmodells dieser Arbeit für Web Services realisierbar erscheint.

## 6.4 Qualitätsmodelle für zusammengesetzte Services

In diesem Abschnitt werden zwei zentrale Ansätze vorgestellt, wie die Qualitätsmerkmale einzelne Prozessschritte aggregiert werden können, um Aussagen über die Qualität eines zusammengesetzten Services treffen zu können. Es werden jeweils die modellierbaren Qualitätsmerkmale und betrachteten Kontrollflusskonstrukte untersucht.

### 6.4.1 Qualitätsmodelle für das METEOR-S-Projekt

In Rahmen des Projektes *METEOR-S: Semantic Web Services and Processes*<sup>3</sup> ist ein Ansatz zur Modellierung von Qualitätsmerkmalen für Prozessen entwickelt worden [31].

#### Aggregation der Qualitätsmerkmale mit dem SWR-Algorithmus

Ein von Jorge Cardoso et alii entwickelter Ansatz im Rahmen des METEOR-S-Projektes basiert dabei auf einem Algorithmus, der ausgehend von einer graphbasierten Prozessbeschreibung die aggregierten Qualitätseigenschaften des gesamten Prozesses berechnet. Der *Stochastic Workflow Reduction* (SWR) Algorithmus reduziert dabei schrittweise die Prozessdefinition auf eine einzelne Aktivität, indem bestimmte Kontrollfluss-Muster erkannt werden und reduziert werden. Reduktion bedeutet hier, eine Menge von Aktivitäten zu einer Aktivität zusammenzufassen. Je nach Kontrollfluss-Muster werden die Eigenschaften dann nach definierten Formeln errechnet.

**Qualitätsmerkmale** Dieser Ansatz definiert in der neueren Version [31] Zeit, Kosten und Zuverlässigkeit (Reliability) als relevante QoS-Dimensionen. In früheren Versionen wurde noch eine Qualität modelliert, die aus einzelnen, gewichteten und normierten Einflussfaktoren zu einer Zahl zwischen 0 und 1 bestand.

---

<sup>3</sup>Projekt Homepage: <http://lsdis.cs.uga.edu/projects/meteor-s/>

**Kontrollfluss** Für die graphbasierte Prozessbeschreibungssprache sind die Kontrollfluss-Muster *Sequenz*, *parallele Ausführung*, *Conditional System* (entspricht einem XOR-Block) und *Schleifen* spezifiziert. Es wird bei einem Conditional System eine Wahrscheinlichkeit für die Pfade benötigt und bei der Schleife eine erwartete Anzahl an Schleifendurchläufen. Zusätzlich ist noch ein so genanntes *Fault-Tolerant System*, das mit einem AND-Split beginnt und die parallele Ausführung mit einem XOR-Join zusammenführt. Dieses Muster entspricht einem Discriminator nach einem AND-Split (siehe auch [66]).

**Einordnung und Abgrenzung** Der SWR-Algorithmus beschreibt die Aggregation für graphbasierte Prozessbeschreibungssprachen und somit ein ähnliches Verfahren wie die Aggregationsmuster in dieser Arbeit. Es werden allerdings keine Konstrukte mit einem N-out-of-M-Join unterstützt. Weiterhin ist die Metrik der Qualitätsmerkmale unzureichend definiert. Die Qualitätsmerkmale sind statisch vordefiniert und die Aggregationen ebenfalls fest vorgegeben. Anpassungen der Aggregation oder Erweiterung um zusätzliche Qualitätsmerkmale sind nicht vorgesehen. Daher existiert auch kein Meta-Modell für Qualitätsmerkmale, das Aspekte wie Skalenniveau, Wertebereich oder Art des Wertes berücksichtigt. Die Aggregationen sind nur definiert für metrische Qualitätsmerkmale. Außerdem müssen für alle Services dieselben Qualitätsmerkmale definiert sein.

### Semantische Erweiterungen

Im Zusammenhang mit METEOR-S gibt es Überlegungen, die Bedeutung und die Eigenschaften der Aktivitäten genauer zu beschreiben. So wird in [122] das METEOR-S Web Service Annotation Framework vorgestellt. Ziel ist es, die semantische Bedeutung von Web Services an die WSDL-Beschreibung anzuhängen. Dabei wird eine Ontologie benutzt, in der die verwendeten Namen des WSDL-Dokumentes für Service und Operationen nachgeschlagen werden.

In [3] wird darauf aufbauend ein Ansatz zur Prozessoptimierung vorgestellt. Hierbei wird ein Web Service-Modell betrachtet, das jeweils die Operationen eines Services fokussiert, nicht die Services selbst. Für jede Operation können Funktionalität und die Qualitätsmerkmale definiert werden. Die Funktion ist basierend auf einer gemeinsamen Ontologie semantisch beschrieben und berücksichtigt zusätzlich Ein- und Ausgabeparameter, Vor- und Nachbedingungen sowie mögliche Ausnahmen (Exceptions). Als Qualitätsmerkmale werden Zeit, Kosten, Zuverlässigkeit und Verfügbarkeit als allgemeine Qualitätsmerkmale angegeben (siehe auch Abschnitt 4.5.4). Zusätzlich können noch für den einen Anwendungsbereich spezifische Qualitätsmerkmale definiert werden. Es wird hier also als Erweiterung zum SWR-Algorithmus aufgezeigt, wie Qualitätsmerkmale semantisch beschrieben werden können.

Durch diese Erweiterung kann die Service-Auswahl auch semantisch unterstützt werden. Für eine abstrakte Prozessdefinition werden dann Services beziehungsweise deren Operationen ausgewählt, wobei sowohl Funktionalität als auch



Qualitätsmerkmale berücksichtigt werden. Zur Aggregation über den gesamten Prozess wird dann der SWR-Algorithmus eingesetzt. Dann können die Qualitätsmerkmale aggregiert werden, wenn die einzelnen Service ausgewählt sind. Es wird weiter vorgeschlagen, die Optimierung der Service-Auswahl anhand der Qualitätsmerkmale zu automatisieren. Ziel ist dabei die Optimierung unter Berücksichtigung von globalen Constraints. Als Verfahren wird die ganzzahlige lineare Optimierung (*Integer Linear Programming*) vorgeschlagen.

**Einordnung und Abgrenzung** Die Technik der Ontologien wird hier benutzt, um Service-Beschreibungen mit semantischen Informationen zu annotieren. Dies erlaubt dann eine Beschreibung der Funktionalität und der Qualitätsmerkmale. Beides wird in diesem Ansatz nur dadurch getrennt, dass für Qualitätsmerkmale die Aggregationsregeln zu definieren sind. Wie diese Qualitätsmerkmale aussehen können und wie die Aggregationsmuster zu definieren sind, ist nicht genauer erläutert. Es wird in [3] nur erwähnt, dass für die anwendungsbereich-spezifischen Qualitätsmerkmale die Aggregationsmuster anzugeben sind. Ansonsten gelten dieselben Einschränkungen wie beim SWR-Algorithmus.

#### 6.4.2 Blockstruktur-basierte Aggregation von Qualitätsmerkmalen

Von Michael C. Jaeger et alii wird in [72] der Ansatz verfolgt, die Qualitätsmerkmale eines Prozesses aus den Qualitätsmerkmalen der einzelnen Prozessschritte zu berechnen. Grundlage für den betrachteten Kontrollfluss sind hier die Workflow-Patterns von van der Aalst [156].

**Qualitätsmerkmale** Grundlage für die Qualitätsmerkmale sind so genannte *numerical QoS Dimensions*. In [72] werden die Qualitätsmerkmale Ausführungsdauer, Kosten, Durchsatz, Verfügbarkeit und Verschlüsselung spezifiziert, wobei alle möglichen Werte als numerisch angesehen werden. Es ist zusätzlich erwähnt, dass einige Qualitätsmerkmale nicht numerisch sein können und diese anders aggregiert werden müssten.

Weitere Aspekte der Aggregation werden in [71] betrachtet. Unter anderem werden Abhängigkeitsbereiche (*Dependency Domain*) definiert, bei denen mehrere Services sich eine Ausprägung eines Qualitätsmerkmals teilen. Wenn beispielsweise mehrere Services von einem physikalischen Server ausgeführt werden, so ist ein Abhängigkeitsbereich für diese Services definiert und ihre Verfügbarkeit ist anders zu berechnen, als wenn die Services von unterschiedlichen Servern ausgeführt werden.

**Kontrollfluss** Es werden die einzelnen Workflow-Patterns jeweils untersucht und zunächst analysiert, ob sie im Kontext der Aggregation relevant sind oder nicht. Anschließend wird für alle relevanten Workflow-Pattern angegeben, welche Aggregation benutzt werden kann um die Qualitätsmerkmale zu berechnen. Es wird

dann für einen Algorithmus angegeben, wie die definierten Qualitätsmerkmale zur Berechnung der Qualitätsmerkmale des Prozesses benutzt werden. Das Vorgehen ist dabei stark angelehnt an den in Abschnitt 6.4.1 vorgestellten SWR-Algorithmus.

**Einordnung und Abgrenzung** Der SWR-Algorithmus wird zunächst erweitert um N-out-of-M-Joins und auf Workflow-Patterns übertragen. Die Auswahl der relevanten Pattern unterscheidet sich von der Auswahl dieser Arbeit. Die Verwendung eines Kontrollflusskonstruktes mit einem Or-Split und einem N-out-of-M-Join ist bereits in Abschnitt 5.3.2 kritisch betrachtet worden. Weiterhin wird die Annahme einer Gleichverteilung bei XOR-Blöcken getroffen. Welche Aggregationen sich durch die Angabe von Wahrscheinlichkeiten realisieren lassen, wurde in Abschnitt 5.4.5 aufgezeigt. In [71] werden zwar Wahrscheinlichkeiten und Erwartungswerte diskutiert, aber keine konzeptionelle Lösung für die auftretenden Probleme vorgeschlagen. Insbesondere wird nicht betrachtet, dass sich die Bedeutung eines Qualitätsmerkmals nach einer Aggregation ändern kann.

Ansonsten ist erkennbar, dass nur metrische Qualitätsmerkmale (als *Numerical QoS Dimension* bezeichnet) abgebildet werden und keine Unterstützung für die Modellierung von Qualitätsmerkmalen vorgesehen ist. Insbesondere sind Aspekte wie das Skalenniveau, die Art des Werts und die Semantik nicht genauer betrachtet. Bemerkenswert ist allerdings, dass erkannt wird, dass für Kosten und Zeit maximale und minimale Werte berechnet werden können. Diese Werte sind aber nur für den gesamten Prozess relevant, das heißt, es wird auch hier nicht betrachtet, dass Qualitätsmerkmale durch eine Aggregation ihre Bedeutung ändern können. Dies wurde in Abschnitt 5.5 beschrieben.

Auffällig ist insbesondere die Modellierung der Verschlüsselung als numerisches Qualitätsmerkmal: Dieses Merkmal bildet die Schlüssellänge in Bits ab, wird also als Zahl angegeben. Es wird darüber hinaus erwähnt, dass ein berechneter Mittelwert nicht sinnvoll sei. Genauer betrachtet, darf auf dieses Qualitätsmerkmal nur Operationen durchgeführt werden, die bei einer Ordinalskala erlaubt sind. Durch die konzeptionelle Grundlage des Qualitätsmerkmals dieser Arbeit und insbesondere der Skalenniveaus ist eine präzisere Modellierung derartiger Qualitätsmerkmale möglich. Missverständnisse und Mehrdeutigkeiten dieser Art werden dadurch verhindert.

## 6.5 Automatisierte Geschäftsprozessoptimierung

Nachdem im vorherigen Abschnitt die Aggregation von Qualitätsmerkmalen für einen zusammengesetzten Service vorgestellt wurde, wie nun die Optimierung des Geschäftsprozesses betrachtet. Hierbei wird vor allem auf Prozessoptimierung im Kontext serviceorientierter Architekturen unter Berücksichtigung mehrerer Qualitätsmerkmale eingegangen. Zunächst erfolgt ein Überblick über einige in der Literatur vorgeschlagene Verfahren, anschließend wird ein konkretes Projekt vorgestellt.

### 6.5.1 Überblick

Ein Überblick über zwei grundlegende Konzepte und dazugehöriger Verfahren zur automatischen Geschäftsprozessoptimierung durch eine optimale Service-Auswahl wird in [70] aufgezeigt. Ziel ist dabei die Optimierung der Qualität des Prozesses unter Berücksichtigung von globalen Randbedingungen (*Constraints*) und mehreren zu optimierenden Qualitätsmerkmalen.

Zunächst wird davon ausgegangen, dass funktional angemessene Services als Kandidaten für die Prozessschritte ausgewählt wurden. Nun gilt es diejenigen Kandidaten auszuwählen, die zur besten Qualität des Gesamtprozesses führen. Es wird ebenfalls vorausgesetzt, dass die Qualitätsmerkmale quantifizierbar sind oder die Ausprägungen in metrische Werte umgewandelt werden können. Für alle Services sind dieselben Qualitätsmerkmale inklusive der Richtung ihrer Qualitätssteigerung definiert. Es werden folgende Beispiele für derartige Qualitätsmerkmale genannt: Maximale Ausführungsdauer, Kosten, Reputation und Verfügbarkeit. Die Reputation wird von Benutzern des Services vergeben.

Bei der Geschäftsprozessoptimierung wird aus der Menge der Service-Kandidaten jeweils einer für die Prozessschritte ausgewählt. Dabei sind mehrere Randbedingungen einzuhalten (beispielsweise maximale Ausführungsdauer des Prozesses) und über mehrere Qualitätsmerkmale zu optimieren (möglichst geringe Kosten und möglichst geringe Ausführungsdauer). Unter diesen Voraussetzungen wird in [70] aufgezeigt, wie sich dieses kombinatorische Optimierungsproblem auf einen *0/1-Knacksack-Problem* und auf ein *Multi-Mode Resource Constrained Project Scheduling Problem* (MRCPSP) übertragen lässt. Für diese Probleme existieren Algorithmen und Heuristiken zur Lösung, deren Anwendbarkeit für die Optimierung der Service-Auswahl anschließend untersucht wird. In [70] werden verschiedene Heuristiken bezüglich ihrer Effizienz untersucht und verglichen.

Um Optimierungsverfahren anwenden zu können, wird oftmals eine Normierung und Gewichtung der Qualitätsmerkmale vorgenommen. Hierbei kann beispielsweise *Simple Additive Weighting* (SAW) eingesetzt werden. Hier werden die Ausprägungen der Qualitätsmerkmale je nach Richtung so auf  $[0, 1]$  normiert, dass 1 die beste Qualität ausdrückt. Anschließend sind die Qualitätsmerkmale zu gewichten. Die so gewichteten und normierten Qualitätsmerkmale können dann für eine automatische Optimierung benutzt werden. Dies stellt eine Alternative zu der in Abschnitt 4.3.7 beschriebenen Nutzenfunktion dar.

Von San-Yih Hwang et alii wird in [66] ein Verfahren zur Optimierung der Service-Auswahl für Geschäftsprozesse vorgestellt. Es werden die Kontrollfluss-*Sequence*, *Parallel* (entspricht einem AND-Block), *Conditional* (XOR-Block) sowie Schleifen unterstützt. Zusätzlich ist noch das Kontrollflusskonstrukte *Fault Tolerance* spezifiziert, dass aus einem *And-Split* und einem XOR-Join besteht (siehe auch Abschnitt 6.4.1). Hierzu lässt sich aus der Beschreibung der Bedeutung schließen, dass dieses Konstrukt mit einem AndNM-Block mit  $n = 1$  vergleichbar ist.

Als Qualitätsmerkmale werden Kosten, Ausführungsdauer und Zuverlässigkeit

spezifiziert, sowie das Qualitätsmerkmal *Fidelity*, das von Benutzern vergebenes Ansehen beschreibt. Es wird zusätzlich aufgezeigt, wie eine Wahrscheinlichkeitsverteilung der Werte für ein Qualitätsmerkmal angegeben werden kann. Dennoch sind alle Qualitätsmerkmale der kardinalen Skalen zugeordnet. Für die metrischen Merkmale ist dann jeweils die Aggregation für die Kontrollflusskonstrukt angegeben, wobei neben Addition, Multiplikation und Min/Max-Berechnung auch eine konditionale Auswahl verwendet wird.

Es werden zwei Ansätze vorgestellt, die Service-Auswahl zu optimieren: Ein linearer Optimierungs-Ansatz wird vorgestellt, der zusätzlich den Suchraum einschränkt. Als weiterer Ansatz wird eine Heuristik vorgestellt, die auf einer *Greedy Method* basiert. Hier wird eine unterschiedliche Service-Auswahl paarweise verglichen und jeweils die bessere verwendet [66].

### 6.5.2 Ganzzahlige Lineare Optimierung

Als Verfahren für das in Abschnitt 6.4.1 vorgestellte METEOR-S-Projekt wird die ganzzahlige lineare Optimierung *Integer Linear Programming* vorgeschlagen. Bei der linearen Optimierung wird für eine gegebene Matrix  $A \in \mathbb{R}^{m,n}$  und zwei gegebene Vektoren  $b \in \mathbb{R}^m$ ,  $c \in \mathbb{R}^n$  ein Vektor  $x \in \mathbb{R}_{\geq 0}^n$  gesucht, für den gilt, dass alle Komponenten des Vektors  $(b - Ax)$  nicht negativ sind und unter allen diesen Vektoren  $x$ , ist das Skalarprodukt  $c^T x$  maximal. Dieses Optimierungsproblem wird oft abkürzend in der Standardform als  $\max\{c^T x \mid Ax \leq b, x \geq 0\}$  geschrieben, wobei die Bedingung  $Ax \leq b$  komponentenweise zu verstehen ist [1].

Der Vektor  $x$  gibt an, ob der Service verwendet wird, dass heißt eine 1 für  $x_i$  bedeutet, dass Service  $i$  verwendet wird. Bei der ganzzahligen linearen Optimierung gilt, dass  $x \in \mathbb{Z}^n$  sein muss, also nur ganzzahlige Werte für den Vektor  $x$  zugelassen sind, hier sogar nur 0 und 1. Für die Optimierung der Service-Auswahl enthält der Vektor  $b$  die Randbedingungen der  $m$  Qualitätsmerkmale. Die Matrix  $A$  enthält die Ausprägungen für die  $m$  Qualitätsmerkmale der  $n$  Services.

In [186] wird ebenfalls ein Ansatz vorgestellt, Services zur Laufzeit eines Geschäftsprozesses auszuwählen. Hier werden mehrere Qualitätsmerkmale berücksichtigt, wie zum Beispiel der Preis, die Ausführungsdauer und die Zuverlässigkeit. Zusätzlich werden globale Einschränkungen und Bedingungen berücksichtigt. Die Optimierung der Qualitätsmerkmale des serviceorientierten Prozesses wird in [186] durch lineare Optimierung gelöst, mit dem die optimale Auswahl der Services getroffen wird. Hierzu werden alle möglichen Ausführungspfade ermittelt und diese jeweils einzeln optimiert. Eine beweisbar optimale Lösung für ganzzahlige lineare Optimierung ist ein NP-schweres Problem [185].

Zu Ansätzen der linearen Programmierung wird in [185] angemerkt, dass derartige Methoden für einen Einsatz zur Laufzeit nicht effizient realisierbar sind, da sie zu komplex sind. Insbesondere ganzzahlige lineare Optimierung führt dazu, dass Optimierungsproblem in eine Reihe linearer Optimierungen zu zerlegen, die dann alle zu optimieren sind.

### 6.5.3 Das QoS Capable Web Service Composition-Projekt

Die Optimierung von Qualitätsmerkmalen über einen gesamten Geschäftsprozess wird im QoS Capable Web Service Composition (QCWS) Projekt [185] betrachtet und die Qualitätsmerkmale als End-To-End-QoS bezeichnet. Es wird versucht, einen optimalen Prozess zu finden, in dem man aus den unterschiedlichen verfügbaren Services die optimale Service-Auswahl trifft. Es werden Service-Klassen definiert, für die es mehrere Services gibt. Zusätzlich gibt es jeden Service mit unterschiedlichen Ausprägungen der Qualitätsmerkmale, die als unterschiedliche Levels bezeichnet werden. Bei der Optimierung wird neben den End-To-End-QoS auch die Kapazitäten der Services sowie deren aktuelle Auslastung berücksichtigt.

Das Service-Modell des Ansatzes definiert Klassen für Services. Für jede Service-Klasse  $S$  ist ein Schnittstellen Parameter-Satz definiert:  $(S_{in}, S_{out})$ . Ein Service  $s$  ist einer Service-Klasse zugeordnet. Das Konzept der Service-Klassen ist ähnlich dem Konzept der Service-Typen im Service-Modell dieser Arbeit (siehe Definition 14). Allerdings hat ein Service im QCWS-Projekt mehrere Levels in denen er angeboten wird. Für jeden Service-Level  $l$  kann ein QoS-Vektor  $q(s, l)$  angegeben werden, wobei die Elemente dieses Vektors die Ausprägungen der einzelnen Qualitätsmerkmale des Services  $s$  auf dem Level  $l$  sind. Das Konzept im QCWS-Projekt ist in diesem Punkt abweichend vom Modell dieser Arbeit, da einem Service genau ein Service-Level zugeordnet worden ist. Dennoch sind die Sichtweisen insofern ähnlich, als dass für einen Service Ausprägungen der Qualitätsmerkmale veröffentlicht werden.

**Qualitätsmerkmale** Die im QCWS-Projekt modellierten Qualitätsmerkmale unterscheiden sich allerdings in wesentlichen Punkten vom Qualitätsmodell dieser Arbeit. Zunächst ist zu bemerken, dass vier Qualitätsmerkmale exemplarisch betrachtet werden: Antwortzeit, Kosten, Verfügbarkeit und Zuverlässigkeit. Antwortzeit und Kosten sind als metrische Werte mit einer Maßeinheit definiert. Verfügbarkeit und Zuverlässigkeit werden aus den historischen Ausführungsdaten gewonnen und vom Service Broker verwaltet. Hierzu ist anzumerken, dass diese Daten für neue Services nicht vorhanden sind, sondern erst im Laufe der Zeit gewonnen werden. Zusätzlich ist die Verlässlichkeit der Aussagen erst nach mehreren Service-Aufrufen gegeben und benötigt Informationen vom Requestor, der diese dem Service Broker mitteilen muss.

Eine hohe Dynamik und flexible Änderungen verfügbarer Services wird allerdings auch in [185] als generelle Eigenschaft von serviceorientierten Architekturen angenommen. Dieses lässt die Annahmen des QCWS-Projektes in dieser Hinsicht als kritisch erscheinen. Die historischen Daten liegen nur dem Broker vor und es ist im QCWS-Projekt nicht vorgesehen, dass diese Informationen veröffentlicht werden. Dies ist nicht notwendig, da angenommen wird, dass die Optimierung eines Geschäftsprozesses nicht vom Requestor vorgenommen wird, sondern vom Service Broker. Der Requestor erhält auf eine Anfrage an einen Broker nicht wie sonst üblich einen Service, sondern eine optimierte Geschäftsprozessdefinition.

Die Optimierung des Ansatzes basiert auf einer Benutzbarkeits-Funktion (englisch Utility Function), die unter anderem die Systemauslastung berücksichtigt und der Nutzen (englisch benefit function, siehe auch [130]). Darüber hinaus werden die Anforderungen eines Requestors an die Qualitätsmerkmale und deren Gewichtung berücksichtigt. Im QCWS-Projekt haben alle Services dieselben Qualitätsmerkmale, individuelle Qualitätsmerkmale sind ebenso wenig vorgesehen, wie Qualitätsmerkmale anderer Skalen.

**Kontrollfluss** Das in [185] vorgestellte Verfahren zur Optimierung ist auf sequentielle Prozesse beschränkt. Es werden zwar vier grundlegende Kontrollflusskonzepte erwähnt (Sequenz, Schleifen, parallele und konditionale Nebenläufigkeit), aber es wird gesagt, dass aus Sicht von Geschäftsprozessen vor allem die Sequenz relevant ist. Andere Konstrukte lassen sich auch zu sequentiellen Abläufen umformen. Dazu wird vorgeschlagen, alle möglichen Ausführungspfade als sequentielle Prozesse aus einer Prozessdefinition zu extrahieren. Diese Sequenzen werden dann einzeln durch den Algorithmus optimiert.

Diese Reduzierung ist ebenfalls kritisch anzusehen, vor allem, weil nicht vor der Ausführung bekannt sein kann, welcher Pfad letztendlich beschritten wird, siehe dazu auch Abschnitt 5. Aus den alternativen Pfaden werden einige erst zur Laufzeit ausgewählt und das Netzwerk der Prozessbeteiligten birgt gerade für die nebenläufige Ausführung ein großes Optimierungspotential [34].

**Aggregationsmuster** Weiterhin arbeitet das in [185] vorgestellte Optimierungsverfahren nur mit linearen Aggregationen für die Qualitätsmerkmale. Dazu ist es notwendig, die Werte der prozentualen Qualitätsmerkmale (zum Beispiel Verfügbarkeit) zu logarithmieren, um dann eine Addition der Werte anwenden zu können. Auch andere Werte sollen gegebenenfalls so umgewandelt werden, so dass ihre Aggregation eine Summation ist [184]. Dies ist eine notwendige Voraussetzung für die gewählten Optimierungsalgorithmen.

**Zusammenfassung** Das QCWS-Projekt zeigt ein umfangreiches Konzept auf, dass sowohl für Service-Klassen Qualitätsmerkmale definiert, als auch die Service-Auswahl in einem Geschäftsprozess optimiert. Daher wird das Ziel erreicht, für einen Requestor die Auswahl der besten Services unter Berücksichtigung des gesamten Geschäftsprozesses zu realisieren.

Allerdings sind dazu Annahmen zu treffen und das Konzept ist nur mit starken Einschränkungen realisierbar. So gibt es prinzipielle Unterschiede in der Auffassung einer serviceorientierten Architektur: Es wird in den Ansätzen von Tao Yu und Kwei-Jay Lin [185, 183] davon ausgegangen, dass der Service Broker die Optimierung vornimmt und Prozessdefinitionen generiert. Dies ist kritisch, da der Requestor nicht nachvollziehen kann, dass die Prozessdefinition tatsächlich seinen Anforderungen genügt und der Prozess sein Geschäftsziel erreicht. Aus Sicht des QCWS-Projekt ist dies aber notwendig, da nur der Broker über alle Informationen

verfügt, beispielsweise wird die Historie der Service-Aufrufe benötigt für die Qualitätsmerkmale Verfügbarkeit und Zuverlässigkeit. Hier weicht das Projekt deutlich ab von anderen Auffassungen, in denen ein Provider Werte für derartige Qualitätsmerkmale garantiert. Außerdem ist die Erzeugung der optimierten Prozessdefinition durch den Broker notwendig, da die Kapazitäten der Service Provider und ihre momentane Auslastung bekannt sind. Dies wird vom Broker bei der Optimierung ebenso berücksichtigt und widerspricht den in Abschnitt 3.3 untersuchten Abstraktionsebenen der Service-Orientierung, insbesondere der Verwaltung der Ressourcen durch den Provider. Das Service-Modell dieser Arbeit hat diesbezüglich eine abweichende Auffassung: die Lastverteilung der Service-Aufrufe sollte eher vom Service Provider als vom Service Broker vorgenommen werden.

Die Restriktionen der Optimierung auf Summation, metrische Qualitätsmerkmale und sequentielle Prozesse sind ebenso kritisch anzusehen. Wandelt man einen Geschäftsprozess in Sequenzen um und optimiert dann jeweils die Sequenzen, so muss bei der Instanziierung des Geschäftsprozesses eine Sequenz ausgewählt werden. Entscheidungen über alternative Pfade zur Laufzeit sind daher nicht mehr möglich, ohne die optimierte Sequenz zu verlassen. Weiterhin ist anzumerken, dass eine dynamische Optimierung zur Laufzeit als typisch für eine serviceorientierte Umgebung angesehen wird, aber Änderungen im Prozessablauf oder Fehler von Services nicht beachtet werden.

# Kapitel 7

## Diskussion

Das Service-Modell und das Qualitätsmodell werden in diesem Kapitel diskutiert. Zusätzlich wird die konzeptionelle Unterstützung der Modelle für Geschäftsprozesse, sowie die vorgestellten Aggregationsmuster diskutiert. Hierzu werden jeweils Annahmen diskutiert und auf mögliche Erweiterungen eingegangen.

### 7.1 Service-Modell

In Abschnitt 3.1 sind die Annahmen des Service-Modells spezifiziert worden. Die zentralen Annahmen sind notwendig, damit sich eine serviceorientierte Architektur sinnvoll und vorteilhaft für die Beteiligten einsetzen lässt. Zu den zentralen Annahmen zählt, dass eine Leistung als Service bei einem Broker angeboten wird und es Requestoren gibt, die diese Leistung benötigen. Diese Annahmen wurden direkt diskutiert und ihre Notwendigkeit untermauert. Außerdem wurden in Abschnitt 3.4 die Prinzipien der Service-Orientierung diskutiert. Im Folgenden sind einige Annahmen, sowie Erweiterungen und Änderungen des Service-Modells diskutiert.

**Service-Typ** Das Service-Modell manifestiert die Trennung der Funktionalität von den Qualitätsmerkmalen eines Services. Es wurde die grundlegende Annahme getroffen, dass die Funktionalität so beschrieben wird, dass ein Requestor Services der gesuchten Funktionalität identifizieren kann. Insbesondere das Konzept des Service-Typs erlaubt eine lose Kopplung, indem es Funktionalität und Schnittstelle definiert. Ein Service-Typ wird dabei als standardisiert angesehen.

Wird diese Annahme aufgehoben, so kann jeder Provider seine Services mit proprietären Beschreibungen versehen. Dadurch wären Services bezüglich ihrer Funktionalität nicht ohne weiteres vergleichbar und die lose Kopplung ist gefährdet. Ähnliches gilt für den Fall, dass sich die Schnittstelle unterscheidet. Auch diese gefährdet die lose Kopplung, da bei geänderter Schnittstelle eines anderen Services die Aufrufe angepasst werden müssen.



Ist die Beschreibung der Funktionalität nicht standardisiert, ist unklar welche Aspekte der Funktionalität von den Qualitätsmerkmalen abzugrenzen sind. Dies wurde durch die Beispiele in Abschnitt 4.1, die Diskussion der Begriffe in Abschnitt 4.3.1 und die ausgewählte Literatur in Abschnitt 6.3 verdeutlicht. Beispielsweise ist schwer zu beurteilen, ob die Verschlüsselung der Kommunikation ein Teil der Funktionalität eines Services ist oder ob dieses eine Qualitätseigenschaft darstellt. Die konzeptionelle Trennung von Funktionalität und Qualitätsmerkmalen im Qualitätsmodell gewährleistet, dass Services desselben Typs funktional äquivalent sind.

**Beschreibung der Funktionalität** Im Rahmen des in Abschnitt 6.4.1 vorgestellten METEOR-S Projekt wurde in [3] vorgeschlagen, die Funktionalität eines Services semantisch zu beschreiben. Die Notwendigkeit wird auch in [96] betont. Zur semantischen Beschreibung können Ontologien benutzt werden. In einer Ontologie werden die für einen bestimmten Anwendungsbereich relevanten Konzepte und Zusammenhänge abgebildet [143]. Vokabeln und Begriffe sowie ihre Bedeutung werden definiert, indem logikbasierte Techniken verwendet werden. Ziel ist es, die Bedeutung von Aussagen und Vokabeln zur Laufzeit zu ermitteln, so dass Anwendungen Informationen austauschen können. Durch die Logik-Basierung ist es möglich, Inferenzmechanismen anzuwenden, um neues Wissen zu generieren [47]. Es werden nach [59] drei Arten von Ontologien definiert:

- *Top-Level-Ontologien* stellen allgemeingültiges und bereichsübergreifendes Wissen dar.
- *Bereichsontologien* definieren einen bestimmten Bereich durch Einigung einer Expertengruppe auf gemeinsame Vokabeln und Konzepte.
- *Anwendungsontologien* definieren einen festen, eng abgegrenzten Anwendungsbereich.

Zur semantischen Beschreibung der Funktionalität eines Service-Typs kann eine Anwendungsontologie erstellt werden. Mit der Web Ontology Language [141] (OWL) wurde eine Ontologie-Sprache spezifiziert, mit der auch Web Services beschrieben werden können [148]. Unter anderem wird in [114] vorgeschlagen, Web Services Prozesse mit semantischen Informationen aus Ontologien anzureichern. Ziel der semantischen Beschreibung eines Web Services ist es, die Bedeutung der Leistung des Web Services zu beschreiben. Es wird für eine Maschine *verständlich*, was dieser Web Service leistet und gegebenenfalls sogar, wie der Web Service dies leistet.

Die semantische Annotation kann durch einen Provider erstellt werden, von einem Dritten wie beispielsweise dem Service Broker vorgenommen werden oder für einen Service-Typ durch ein geeignetes Gremium standardisiert werden. Somit kann die Technik der semantischen Beschreibungen und Ontologien eingesetzt werden, um die Funktionalität eines Services zu beschreiben. Ontologien stellen daher eine mögliche Realisierung für diese Entität des Service-Modells dar.

**Operationen unterschiedlicher Funktionalität** Für das Service-Modell wurde definiert, dass eine Schnittstelle Operationen definiert und dass die Operationen eine kohärente Funktionalität bilden [178, 180]. Die Operationen werden aufgerufen, wenn die durch die Funktionalität spezifizierte Leistung erbracht werden soll. Es gilt die Annahme, dass die Operationen Varianten des Aufrufs darstellen. Für das in Abschnitt 1.1 skizzierte Beispiel *Buchversand* sei eine Operation definiert, die eine Bestellung durch Angabe der ISBN-Nummer erlaubt, eine andere Operation per Autor und Titel. Zusätzlich sind die Operationen jeweils für Bezahlung per Lastschriftverfahren und per Kreditkarte definiert, so dass insgesamt vier Operationen spezifiziert sind.

In einer alternativen Sichtweise einer serviceorientierten Architektur haben die Operationen unterschiedliche Funktionalität. Für das Buchversand-Beispiel werden dann neben der Buchbestellung weitere Operationen spezifiziert, beispielsweise zur Auflistung aller Bücher eines Autors oder zur Ermittlung der Verfügbarkeit eines Buches. Diese Sichtweise ähnelt dem in Abschnitt 3.4.2 beschriebenen RPC-Stil, bei dem eine bestehende Anwendung mit einer Service-Fassade verpackt wird. Dadurch sind „ähnliche“ Leistungen, die alle von derselben Anwendung erbracht werden können, zu einem Service zusammengefasst. Ein Service bringt also nicht mehr eine Art Leistung, sondern jede Operation erbringt eine eigene Art Leistung. Es ergeben sich Auswirkungen auf die Beschreibung der Funktionalität eines Services (die dann alle Leistungen abdecken muss) und somit auch auf die Service-Suche und Auswahl. Im Abschnitt 7.2 wird die Auswirkung einer derartigen Auffassung auf das Qualitätsmodell diskutiert.

**Granularität der Services** Die in dieser Arbeit betrachteten Services stellen einzelne Leistungen dar, die unter anderem im Rahmen eines Geschäftsprozesses aufgerufen werden. In der Literatur werden auch Grid-Services [139] oder Enterprise Services [79, 86] betrachtet. Bei Grid-Services handelt es sich um eher feingranulare Leistungen (beispielsweise Berechnungsaufgaben), wobei vor allem die Verwaltung der Ressourcen vom Grid automatisiert übernommen wird. Anpassungen des Service-Modells sind nicht notwendig, allerdings beeinflussen Qualitätsmerkmale in der Regel, in welche Ressourcen des Grids wie eingesetzt werden. Ein Qualitätsmodell für Grid-Services ist also ebenfalls sinnvoll. Das Qualitätsmodell dieser Arbeit kann prinzipiell auch für Grid-Services eingesetzt werden, da sich keine konzeptionellen Unterschiede für das Angebot und die Nachfrage nach Services ergeben. Allerdings ist das interne Management der Ressourcen im bisherigen Qualitätsmodell nicht explizit berücksichtigt.

Demgegenüber ist ein Enterprise Service eher von grober Granularität. Hier werden komplette Geschäftsbereiche als ein Service angesehen. Die Operationen stellen dann unterschiedliche Funktionalität dar. Es ist zu beachten, dass ein Enterprise Service nicht direkt in einen Prozess integriert wird, da die Granularität zu grob ist. Es sind wie oben diskutiert die einzelnen Operationen zu betrachten und die Qualität der Operationen ist zu spezifizieren. Alternativ kann im Service-

Modell eine zusätzliche Entität eingeführt werden, die verschiedene Services auf eine höheren Aggregationsebene zu einem Enterprise Service zusammenfasst.

## 7.2 Qualitätsmodell

Die Annahmen des Qualitätsmodells wurden in Abschnitt 4.2 formuliert und bauen auf den Annahmen des Service-Modells auf. Zusätzlich sind Annahmen über die Qualitätsmerkmale und ihre Ausprägungen formuliert worden. Im Folgenden werden einige Annahmen, mögliche Erweiterungen und Änderung des Qualitätsmodells diskutiert. Hierbei werden auch die im vorherigen Abschnitt diskutierten Aspekte des Service-Modells einbezogen.

**Operationen unterschiedlicher Funktionalität** Wie in Abschnitt 7.1 beschrieben, kann ein Service mehrere Arten von Leistungen erbringen, wenn Operationen unterschiedliche Funktionalität realisieren. Bei dieser alternativen Annahme ergeben sich Auswirkungen auf das Qualitätsmodell: Die Qualität ist nicht mehr für einen Service zu spezifizieren, sondern jeweils für eine Operation. Dazu ist dann das Modell entsprechend so anzupassen, dass Qualitätsmerkmale für eine Operation definiert werden und die Ausprägungen ebenfalls nicht für den Service, sondern für die entsprechenden Operationen gelten. Es ist allerdings zu beachten, dass bei der Service-Suche nach Operationen gesucht wird und nicht mehr nach Services.

**Veröffentlichung der Werte** In der Wirtschaft werden nicht immer alle Informationen über die Qualität von Produkten und Dienstleistungen einfach aufbereitet zur Verfügung gestellt. Beispielsweise werden Preise für bestimmte Leistungen nur auf Anfrage mitgeteilt. Es ist dann der ausdrückliche Wunsch, dass diese Informationen für die Konkurrenz nicht offen gelegt werden sollen oder dass ein direkter Vergleich mit der Konkurrenz nicht möglich sein soll. In einem solchen Fall treffen die grundlegenden Prinzipien der serviceorientierten Architektur (siehe Abschnitt 3.4) nicht zu. Die Service-Auswahl kann nicht mehr durch den Broker beziehungsweise durch die beim Broker veröffentlichten Informationen durchgeführt werden. Sind also weitere, nicht veröffentlichte Informationen entscheidungsrelevant, muss vom Requestor eine Anfrage an potentielle Provider durchgeführt werden, um die Informationen zu erhalten.

Eine mögliche Umsetzung auf der Basis des Service-Modells dieser Arbeit sieht wie folgt aus: Die Schnittstelle eines Service-Typen wird um Operationen erweitert, die unveröffentlichte Qualitätsmerkmale zurückgibt. Beispielsweise sei eine Operation „Kalkulation“ für den Service-Typen  $T_{shipping}$  definiert, welche die Transportkosten kalkuliert. Durch den Aufruf der Operation bei einem Service, wird einen Preis ermittelt und zurückgegeben. Eine andere Operation stellt dann den eigentlichen Service-Aufruf dar, das heißt, es wird die Leistung erbracht, für die vorher der Preis ermittelt wurde. Im Kontext eines Geschäftsprozesses, für den

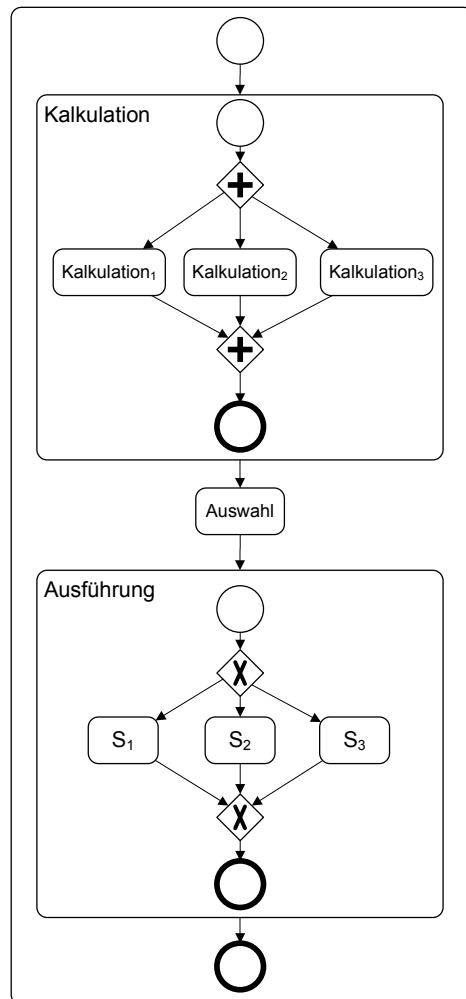


Abbildung 7.1: Explizite Service-Auswahl als Prozessschritt nach vorheriger Preis-anfrage an Provider

der günstigste Service verwendet werden soll, ist dann ein (Teil-)Prozess zu modellieren, der zunächst unterschiedliche Preise einholt und dann den günstigsten Service auswählt. Dies ist in Abbildung 7.1 dargestellt. Hier werden die Kalkulations-Operationen von drei Services parallel aufgerufen (AND-Block *Kalkulation*, wobei *Kalkulation<sub>i</sub>* für Service *S<sub>i</sub>* durchgeführt wird). Anschließend wird explizit ein Service ausgewählt (Prozessschritt *Auswahl*) und der gewählte Service wird aufgerufen (XOR-Block *Ausführung*).

Das Verfahren einer vorherigen Preisermittlung ähnelt der Verhandlung eines Service Level Agreement (siehe Definition 24). Allerdings liegen hier für das Qualitätsmerkmal keine veröffentlichten Werte vor und eine Verhandlung über die Werte muss nicht stattfinden.

**Veröffentlichung einer Funktion** Soll kein fester Wert für ein Qualitätsmerkmal veröffentlicht werden, so kann beispielsweise für eine Preisberechnung eine Funktion veröffentlicht werden. Beispielsweise seien die Kosten für den Transport-Service wie folgt zu berechnen: 100 € Grundgebühr plus 20 € für jedes Kilogramm über 100 Kilogramm Transportware und 1 € für je 10 Kilometer Transport-Entfernung. Eine Funktion zur Berechnung des Beispiels mit Gewicht ( $w$ ) der Transportware und Entfernung ( $d$ ) bildet also Folgendes ab:

$$f_{q_{cost}}(w, d) = \begin{cases} 100 + 20 \cdot (w - 100) + \frac{d}{10} & \text{falls } w > 100 \\ 100 + \frac{d}{10} & \text{sonst} \end{cases}$$

Eine derartige Funktion kann der Broker ausführen, wenn ein Requestor ihm die notwendigen Daten (hier:  $w$  und  $d$ ) mitteilt. Wenn alle Provider ihre Funktion zur Wertberechnung veröffentlichen, sind die Ausprägungen der Qualitätsmerkmale durch den Broker vergleichbar.

Als Alternative zur Veröffentlichung einer Funktion kann auch die Metrik angepasst werden, so dass beispielsweise der Preis pro Kilogramm pro 100 Kilometer angegeben wird. Auch dann sind die Werte der Qualitätsmerkmale direkt durch den Broker vergleichbar.

Wird der Wert eines Qualitätsmerkmals oder die Berechnungsfunktion nicht veröffentlicht, muss eine Anfrage an den Provider gesendet werden. Der Broker kann aufgrund seiner Informationen keine Entscheidung treffen. Daher muss im Geschäftsprozess explizit modelliert werden, dass eine Anfrage bei relevanten Providern durchgeführt wird und anschließend der beste Service ausgewählt wird, wie in Abbildung 7.1 dargestellt.

**Ein Service mit mehreren Qualitätsniveaus** Im Qualitätsmodell ist definiert, dass für einen Service genau eine Qualität spezifiziert ist. Will ein Provider seine Leistung mit unterschiedlichen Ausprägungen für die Qualitätsmerkmale anbieten, so muss er daher mehrere Services anbieten. Um die Flexibilität zu erhöhen, könnte der Provider nur einen Service anbieten und dafür mehrere Qualitätsniveaus festlegen; dies wird unter anderem im QCWS-Projekt [183] vorgeschlagen (siehe Abschnitt 6.5.3).

Im Qualitätsmodell in Abbildung 4.2 ist dann die 1:1 Beziehung in eine 1:n Beziehung zu ändern. Für einen Provider wäre dieses Verfahren einfacher, da weniger Services veröffentlicht werden müssen. Allerdings muss beim Service-Aufruf vom Requestor mitgeteilt werden, welche Qualität gewählt wird. Somit wirkt sich diese Änderung auf den Service-Aufruf aus, was beim ursprünglichen Qualitätsmodell nicht der Fall war. Wird die gewählte Qualität als zusätzlicher Parameter beim Service-Aufruf übermittelt, so ist die Schnittstelle des Services anzupassen.

**Kategorien für Qualitätsmerkmale** Im Kontext serviceorientierter Architekturen wird von Steve Jones in [74] eine (nicht vollständige) Menge von Kategorien

für Service-Eigenschaften vorgeschlagen. Zunächst werden allgemeine Kategorien beschrieben, wie beispielsweise Performanz, Kapazität, Zuverlässigkeit und Sicherheit. Je Kategorie gibt es verschiedene Aspekte, die in einer Service-Beschreibung spezifiziert werden sollten. Die Aspekte werden von Steve Jones jeweils genauer definiert und sind für einen Service anzupassen.

Die Aspekte der Kategorien entsprechen den Qualitätsmerkmalen. Zur besseren Übersicht und insbesondere bei vielen Qualitätsmerkmalen sowie für allgemeine Qualitätsmerkmale ist eine Kategorisierung durchaus sinnvoll, wie auch im Praxis-Projekt des Order-To-Cash-Prozesses in Abschnitt 5.6 dargestellt wurde. Eine Kategorie kann durch eine zusätzliche Entität im Qualitätsmodell abgebildet werden. Allerdings sind die Kategorien für die Service-Auswahl durch einen Requestor ebenso wenig relevant, wie für eine Aggregation der Qualitätsmerkmale. Darüber hinaus werden in [74] Kategorien vorgestellt, die durch andere Konzepte im Qualitätsmodell abgebildet werden, wie Eigentümerschaft (Service Provider) und vertragliche Aspekte (Service Level Agreement). Es werden weitere Aspekte genannt, die über Qualitätsmerkmalen hinausgehen und eher einen beschreibenden Charakter haben: Geschäftsauswirkungen (*business impact*), Geschäftsorganisation (*business organization*), Risikoaspekte (*risks and issues*), sowie Toleranz und Abhängigkeiten (*tolerance* und *dependencies*). Eine genauere Umsetzung derartigen Aspekte ist nicht vorgestellt.

**Überprüfbarkeit der Werte** Das Qualitätsmodell geht nicht auf die Korrektheit der Werte ein. Die Angaben über die Ausprägungen der Qualitätsmerkmale müssen messbar oder überprüfbar sein und die Metrik definiert beispielsweise durch ein technisches Messverfahren, wie dieses geschehen kann. Messverfahren im Kontext logischer und physischer Qualitätsmerkmale wurden in Abschnitt 4.5.5 diskutiert, sowie im Kontext statischer und dynamischer Qualitätsmerkmale im Abschnitt 4.5.6. Problematisch ist, dass für einige Qualitätsmerkmale erst nach Beendigung der Service-Nutzung überprüft werden kann, ob die erbrachte Leistung die zugesicherte Qualität einhält. Außerdem wird in der Literatur vorgeschlagen, eine Überprüfung oder Zertifizierung der Angaben durch eine unabhängige dritte Institution zu realisieren [34, 116].

Auch Shuping Ran schlägt in [126] vor, eine unabhängige Zertifizierungsinstitution zu integrieren. Diese soll ein Zertifikat für einen Service ausstellen, welches dann der Service Beschreibung des Brokers hinzugefügt werden kann. Die Inhalte eines Zertifikates werden vom Provider vorgeschlagen. Hierbei sind Qualitätsmerkmale und Werte frei wählbar, allerdings wird eine Syntax vorgegeben. Die Notwendigkeit einer gemeinsamen Sprache, Definition und Semantik der Qualitätsmerkmale wird nicht beachtet. Die Zertifizierungsinstitution dient hier als Mittler: Wird die angegebenen Qualität anerkannt, so wird davon ausgegangen, dass der Requestor die Qualität verstehen kann. Somit ist neben dem Broker eine weitere Komponente in einer serviceorientierte Architektur und zur Auswahl eines Services zu integrieren.

Sicherheit	Statusüberwachung	Nutzen
–	Telefon	1000
128	Telefon	0
512	Telefon	0
–	Web-Portal	200
128	Web-Portal	400
512	Web-Portal	800
–	EMail	0
128	EMail	700
512	EMail	0

Tabelle 7.1: Nutzen für die Qualitätsmerkmale Sicherheit und Statusüberwachung.

**Unabhängigkeit der Teilnutzen** Um eine Rangfolge der Services zu erstellen, wurde in Abschnitt 4.3.7 das Konzept der Nutzenfunktion vorgestellt. Hierbei wird für jedes Qualitätsmerkmal einzeln spezifiziert, welche Ausprägung welchen Nutzen für den Requestor erzielt. Die Unabhängigkeit der Teilnutzen [53] lässt sich aufheben [12]. Dazu sind nicht die einzelnen Qualitätsmerkmale bezüglich ihres Nutzens zu spezifizieren, sondern beliebige Kombinationen von Qualitätsmerkmalen. Dadurch können bestimmte Kombinationen von Ausprägungen einen hohen Nutzen haben, obwohl die Ausprägungen einzeln betrachtet einen geringen Nutzen haben.

Dies soll in einem Beispiel verdeutlicht werden, bei dem der Transport-Service betrachtet wird (siehe Abschnitt 4.1.2). Ein Requestor möchte nun den besten Service finden, wobei die folgenden Anforderungen zu beachten sind: Die E-Mail-Statusüberwachung kann er nur dann akzeptieren, wenn die E-Mails mit 128-bit Schlüsseln verschlüsselt sind. Für ein Web-Portal gelte, dass eine höhere Verschlüsselung zu bevorzugen ist. Den höchsten Nutzen hätte für den Requestor allerdings eine unverschlüsselte Statusüberwachung per Telefon. Dieser Zusammenhang ist in Tabelle 7.1 durch Angabe von Nutzenwerten je Kombination ausgedrückt. Es wird der Nutzen von 1000 für die Kombination keine Verschlüsselung („–“ in Tabelle 4.2, Seite 70) und „Telefon“ spezifiziert.

Ein solcher Zusammenhang kann mit der in Abschnitt 4.3.7 dargestellten Nutzenfunktion zunächst nicht ausgedrückt werden. Es ist bei abhängigen Teilnutzen notwendig, alle Wert-Kombinationen der jeweiligen Domänen zu bewerten. Ab einer ordinalen Skala können allerdings Wertebereiche bewertet werden. Das Qualitätsmodell ermöglicht prinzipiell eine entsprechende Nutzenfunktion zu formulieren. Allerdings ist der Aufwand für eine einzelne Service-Auswahl vergleichsweise groß und sollte daher als Ausnahme angesehen werden.

## 7.3 Geschäftsprozesse

Die Bedeutung des Qualitätsmodells für Geschäftsprozesse wurde in Kapitel 5 vorgestellt. In Abschnitt 5.2 wurden dazu Varianten der Service-Auswahl untersucht, sowie deren Vorteile und Nachteile diskutiert. Im Folgenden werden die in Abschnitt 5.4 definierten Aggregationsmuster diskutiert, indem die Annahmen und die ausgewählten Kontrollflussblöcke untersucht werden.

**Deferred Choice und Exclusive Choice** Die Workflow-Pattern *Deferred Choice* und *Exclusive Choice* wurden bei der Bildung der Kontrollflussblöcke aus Sicht der Aggregation als äquivalent angesehen. Ein *Deferred Choice* bedeutet, dass während der Ausführung einer Instanz alle folgenden Aktivitäten ausführbar sind. Von der Prozesssteuerung muss gewährleistet werden, dass die anderen nicht mehr ausführbar sind, sobald eine Aktivität gewählt wird. Es wird also exakt eine Aktivität ausgeführt.

Wurde die Variante Service-Auswahl zur Laufzeit gewählt, so sind jeweils alle Services zu ermitteln, die einem *Deferred Choice* folgen. Demgegenüber ist der Prozesssteuerungskomponente bei einem *Exclusive Choice* bekannt, welche Aktivität gewählt wird. Die Auswahl hängt beispielsweise von Bedingungen oder prozessrelevanten Daten ab und kann somit von der Prozesssteuerungskomponente getroffen werden, so dass nur für die gewählte Aktivität ein Service ermittelt werden muss.

Aus Sicht der Service-Auswahl kann es somit zu unterschiedlichem Verhalten kommen. Die Zusammenfassung der beiden Workflow-Pattern ist in diesem Punkt also ungenau. Allerdings wird die Service-Auswahl automatisiert durchgeführt, so dass die Abweichung im Verhalten vernachlässigbar ist. Außerdem ist es der Zweck der Aggregationsmuster, Aussagen über die Qualität des Prozesses treffen zu können, so dass ein Service mit aggregierter Qualität angeboten werden kann. Auch aus dieser Sicht ist der Unterschied in der Ausführung zunächst nicht relevant. Soll die Aktivierung der Prozessschritte und somit die Service-Auswahl bei der Aggregation berücksichtigt werden, so führen die beiden Workflow-Pattern zu unterschiedlichen Kontrollflussblöcken und anderen Aggregationsmustern: Beim *Deferred Choice* sind alle Alternativen zu berücksichtigen, beim *Exclusive Choice* nur der gewählte Pfad. In einem solchen Fall müssen die beiden Workflow-Pattern also separat betrachtet werden und es entsteht ein neuer Kontrollflussblock mit einem *Deferred Choice*.

**Cancellation** Die *Cancellation* Pattern brechen einen (Teil-)Prozess beziehungsweise eine Aktivität ab (siehe Abschnitt 5.3.1). Sie können im Allgemeinen benutzt werden, um zu modellieren, dass in einem Block ein Prozessschritt zu einem Abbruch des Prozesses oder des Blockes führen kann. Wird bei der Prozessmodellierung ein Prozessschritt mit einem entsprechenden Hinweis zu versehen, so kann dies bei der Aggregation der Qualitätsmerkmale ebenfalls berücksichtigt werden.



Es sind beispielsweise nicht alle Prozessschritte zu aggregieren, sondern nur die ersten Prozessschritte bis zu einem möglichen Abbruch; dies gilt insbesondere für die Sequenz, die bei einem Abbruch nicht vollständig ausgeführt wird. Hier kann beispielsweise für das Aggregationsmuster Summe ein minimaler Wert bis zum ersten Abbruch, sowie ein maximaler Wert für alle Prozessschritte der Sequenz (ohne Abbruch) berechnet werden.

In einer serviceorientierten Umgebung ist allerdings zu bedenken, dass nicht klar ist, wie ein Abbruch eines Services durchgeführt werden kann. Die Kontrolle über die Service-Ausführung hat der Provider. Daher muss eine explizite Möglichkeit vorgesehen sein, einen aufgerufenen Service abbrechen. Im Order-To-Cash-Prozess (siehe Abschnitt 5.6) sind Prozessschritte modelliert worden, die zu einem Abbruch des Prozesses führen können. Hierbei stellen die Prozessabbrüche Sonderfälle der Ausführung dar, bei denen die Ausführung aufgrund eines Fehlers abgebrochen werden muss. Für die Aggregation ist hier festgelegt worden, dass diese Abbrüche nicht berücksichtigt werden sollen.

**Anzahl der Schleifendurchläufe** Bei den Aggregationsmustern für Schleifen wurde jeweils die Annahme getroffen, dass die Anzahl der Schleifendurchläufe bekannt ist. Diese Annahme ist in der Praxis nicht immer haltbar, da die Schleifenanzahl in der Regel nicht deterministisch ist.

In [71] wird eine geschätzte Zahl der Schleifendurchläufe verwendet. Der resultierende Wert der Aggregation ist somit ebenfalls als geschätzter Wert zu definieren. In [66] wird eine Wahrscheinlichkeitsfunktion für die Schleifendurchläufe vorgeschlagen. Für jede Anzahl Schleifendurchläufe wird eine Wahrscheinlichkeit angegeben, mit der dieser Fall eintritt. In [57] wird das *Loop Unrolling*-Verfahren vorgestellt, das die Schleife aufrollt, indem der Prozessschritt innerhalb der Schleife mehrfach modelliert wird. Es wird dann jeweils angegeben, mit welcher Wahrscheinlichkeit der Prozessschritt ein weiteres Mal ausgeführt beziehungsweise die Schleife beendet wird. Das auf Markov-Ketten für kontinuierliche Zeit basierende Verfahren gibt somit bei jedem Schleifendurchlauf die Wahrscheinlichkeit an, mit der die Schleife abgebrochen wird beziehungsweise erneut durchlaufen wird.

Eine Wahrscheinlichkeitsfunktion und das Loop Unrolling-Verfahren können genutzt werden, um einen Erwartungswert für die Anzahl der Schleifendurchläufe (*loop*) zu berechnen. Sei  $l$  die maximale Anzahl der Schleifendurchläufe und  $p_i$  die Wahrscheinlichkeit für  $i$  Schleifendurchläufe, dann ist der Erwartungswert für die Anzahl der Schleifendurchläufe  $E(loop) = \sum_{i=1}^l p_i \cdot i$ . Dieser Wert kann dann für die Aggregation verwendet werden, wobei in der Regel die Metrik des aggregierten Qualitätsmerkmals auf einen geschätzten Erwartungswert hinweisen muss.

**Multiple Instances** Die Workflow-Pattern *Multiple-Instances* sind bislang für die Aggregation nicht betrachtet worden. Im Folgenden werden die einzelnen Varianten und ihre Bedeutung diskutiert:

- Da beim Workflow-Pattern *MI with a priori known design time knowledge*

zur Modellierungszeit die Anzahl der parallelen Aktivitäten bekannt ist, ist es äquivalent zu einem AND-Block, bei dem alle Prozessschritte derselben Aktivität entsprechen. Somit sind alle Aggregationsmuster anwendbar, die für einen AND-Block anwendbar sind.

- Da beim Workflow-Pattern *MI with a priori known runtime knowledge* zur Laufzeit die Anzahl der parallelen Aktivitäten ermittelt wird, kann ähnlich wie bei einem Schleifen-Block die Anzahl der Prozessschritte geschätzt werden. Mit dem geschätzten Wert kann ähnlich wie mit einer zur Modellierungszeit bekannten Anzahl aggregiert werden. Allerdings ist die Metrik hier unter Umständen so anzupassen, dass ein geschätzter Wert angegeben wird.
- Da bei der Variante *MI with no a priori runtime knowledge* die Anzahl der Aktivitäten nicht bekannt ist, können auch keine Werte entsprechend der Anzahl der Ausführungen aggregiert werden. Somit sind nur Aggregationsmuster anwendbar, die einen einzelnen Wert beziehungsweise einen Extremwert für diesen Block übernehmen.
- Die Variante *MI without synchronization* ist so zu interpretieren, dass eine parallele Ausführung der Instanzen *neben* dem Prozessverlauf stattfindet [156]. Somit haben diese Aktivitäten keine Auswirkungen auf die Ausführungsdauer und sonstiges Verhalten des restlichen Prozesses. Dies ist bei der Aggregation entsprechender Qualitätsmerkmale (zum Beispiel Ausführungsdauer) zu beachten. Ein derartiger Block wird gestartet, aber sein Ende spielt für die weitere Ausführung keine Rolle. Bezüglich der Anzahl der Prozessschritte gelten die vorher diskutierten Auswirkungen.

**Mittelwerte beim OR-Block** Bei einem OR-Block können mehrere Pfade ausgeführt werden. Daher ist bei der Aggregation bezüglich Erwartungswerten und Mittelwerten eine weitere Vorgehensweise möglich, die alle Kombinationen von Pfaden betrachtet. Bezogen auf das Beispiel in Abbildung 5.10(b) (Seite 148) sind dies:  $S_1$ ,  $S_2$ ,  $S_3$  (jeweils genau einer der Pfade),  $S_1 \wedge S_2$ ,  $S_2 \wedge S_3$ ,  $S_1 \wedge S_3$  (Kombinationen mit je zwei Pfaden) und  $S_1 \wedge S_2 \wedge S_3$  (alle Pfade). Für jede Kombination ist dann jeweils die Wahrscheinlichkeit anzugeben, so dass der Erwartungswert berechnet werden kann. Allgemein ist die Anzahl der Kombinationen bei  $n$  Pfaden offensichtlich  $2^n - 1$ . Mit diesen Wahrscheinlichkeiten sind die Aggregationsmuster nun wie beschrieben anwendbar.

**Ansätze zur automatisierten Geschäftsprozessoptimierung.** Für einen eng abgegrenzten Bereich kann die automatisierte Optimierung eines Geschäftsprozesses angebracht sein. In Abschnitt 6.5 sind dazu Verfahren vorgestellt und diskutiert worden, die dann eine optimale Service-Auswahl für die Prozessschritte berechnen können. Dazu müssen für alle Prozessschritte die zu optimierenden Qualitätsmerkmale identisch definiert sein.

Die Verfahren sind jeweils mit Einschränkungen verbunden: Die unterstützten Kontrollflusskonstrukte sind eingeschränkt (bis hin zu reinen Sequenzen bei den Knapsack-basierten Ansätzen) und es werden in der Regel nur metrische Werte unterstützt. Es ist in Abschnitt 6.5 aufgezeigt worden, dass keines der Verfahren ausreichend ist, um eine Optimierung vorzunehmen, die dem Qualitätsmodell dieser Arbeit gerecht wird. Eine umfassende Unterstützung unter Berücksichtigung der Kontrollflusskonstrukte der Prozessdefinition ist also noch nicht erreicht worden. Allerdings können für einzelne, ausgewählte Qualitätsmerkmale wie Kosten und Ausführungsdauer Optimierungen durchgeführt werden, die menschliche Modellierer bei der Geschäftsprozessoptimierung unterstützen. Dennoch ist eine vollständige Automatisierung der Geschäftsprozessoptimierung in serviceorientierten Umgebungen als offenes Problem anzusehen.

Das Qualitätsmodell dieser Arbeit kann eine konzeptionelle Grundlage dazu bieten, indem Qualitätsmerkmale definiert werden können und gezeigt werden ist, welche Aggregationsmuster verwendet werden können. Ein intuitiver Lösungsansatz aggregiert die Qualität über alle Kombinationen der Service-Auswahl. Die Menge aller verfügbaren Services kann dabei anhand einer Vorauswahl reduziert werden, so dass der Suchraum erheblich verkleinert werden kann. Anschließend können Einschränkungen einzelner Qualitätsmerkmale, beispielsweise Grenzen für Gesamtkosten und Gesamtausführungsdauer, für eine weitere Reduzierung genutzt werden. Im Order-To-Cash-Prozess (siehe Abschnitt 5.6) ist dadurch die Anzahl der gültigen Kombinationen auf vier reduziert worden.

# Kapitel 8

## Zusammenfassung

In diesem Kapitel werden die Ergebnisse der Arbeit zusammengefasst und der wissenschaftliche Beitrag rekapituliert (Abschnitt 8.1). Offene Fragestellungen und der Ausblick auf zukünftige Forschungsthemen werden in Abschnitt 8.2 besprochen. Das Fazit in Abschnitt 8.3 schließt die Arbeit.

### 8.1 Rekapitulation

In dieser Arbeit ist ein Modell von Qualitätsmerkmalen für Services formuliert worden. Es wurde dazu ein Qualitätsmodell entwickelt, mit dem Qualitätsmerkmale definiert und letztendlich ein bester Service ausgewählt werden kann. Als Methodik wurde der Konstruktivismus [49] gewählt. Der erste Schritt eines Ansatzes des Konstruktivismus ist die zirkelfreie und nachvollziehbare Konstruktion von Begriffen. Dazu sind die zentralen Begriffe und Konzepte der Service-Orientierung definiert worden. Anhand von Beispielen wurden die grundlegenden Konzepte illustriert und Anforderungen erarbeitet. Anschließend wurde ein Service-Modell entwickelt und untersucht, in dem Service-Typ und Service zentrale Entitäten darstellen und die Abstraktionsebenen der Service-Orientierung repräsentieren. Unter Einbeziehung der Prinzipien konnte das Paradigma der Service-Orientierung formuliert werden.

Auf der Grundlage des Service-Modells wurden die Anforderungen an das Qualitätsmodell formuliert und das Qualitätsmodell schließlich entworfen. Es definiert als zentrale Entität ein Qualitätsmerkmal, welches Skalenniveau, Wertebereich und Metrik spezifiziert. Das Qualitätsmodell und das Service-Modell wurden integriert, so dass die Qualitätsmerkmale für einen Service-Typ definiert werden können. Dadurch kann die Qualität eines Services spezifiziert werden, indem Ausprägungen der Qualitätsmerkmale angegeben werden. Das integrierte Service- und Qualitätsmodell wurde als relationales Datenmodell realisiert, welches als Basis für eine prototypische, Web-basierte Anwendung verwendet wurde. Dadurch wurde die Umsetzung von Service-Anfragen, inklusive der Verwendung von Nutzenfunktionen, zur qualitätsbasierten Service-Auswahl validiert. Erweiterungen des

Qualitätsmodells um weitere Aspekte wurden ausführlich untersucht und alternative Qualitätsmodelle wurden auch bei der Untersuchung der verwandten Arbeiten diskutiert.

Die Anwendung des Qualitätsmodells wurde im Kontext von Geschäftsprozessen untersucht. Bei der Modellierung eines Geschäftsprozesses wird eine Prozessdefinition erstellt. Durch das Qualitätsmodell dieser Arbeit ist es nun möglich, für die Prozessschritte die besten Services auszuwählen. Es wurden verschiedene Varianten der Service-Auswahl aufgezeigt und dabei untersucht, welche Konsequenzen sich insbesondere in hoch dynamischen serviceorientierten Umgebungen ergeben, in der sich die Menge der Services ständig ändert.

In dieser Arbeit wurden auf dem Qualitätsmodell basierende Aggregationsmuster entwickelt, die wesentlich zur Modellierung von Geschäftsprozessen beitragen können. Es wurden allgemeine Aggregationsmuster formuliert und untersucht, die für viele unterschiedliche Qualitätsmerkmale anwendbar sind. Die Aggregation der Werte wurde mathematisch beschrieben, wobei die Aspekte der Qualitätsmerkmale (insbesondere Skalenniveau und Metrik) berücksichtigt wurden. Demgegenüber wird in vergleichbaren Ansätzen eine statische Menge von Qualitätsmerkmalen fest definiert und beschrieben, wie diese Qualitätsmerkmale zu aggregieren sind. Dadurch sind diese Verfahren nur für die vordefinierten Qualitätsmerkmale anwendbar. Ebenso ist jeweils die Menge der vordefinierten Kontrollflusskonstrukte begrenzt. Wenn ein neues Kontrollflusskonstrukt unterstützt werden soll, so ist für jede einzelne der definierten Qualitätsmerkmale eine entsprechende Aggregation zu definieren. In dieser Arbeit sind demgegenüber die Muster unabhängig von Kontrollflussblöcken spezifiziert worden. Zusätzlich ist jeweils angegeben, für welche Kontrollflussblöcke das Muster angewendet werden kann. Dadurch wird eine einfache Erweiterbarkeit erreicht: für neue Kontrollflussblöcke kann definiert werden welche Aggregationsmuster anwendbar sind. Schließlich wurde das Qualitätsmodell und die Aggregationsmuster in einem Praxisprojekt validiert.

Im Verlauf der Arbeit ist durchgehend gezeigt worden, wie aktuelle Web Service-Technologien eingesetzt werden können um einige Aspekte der Service-Orientierung zu realisieren und welche Aspekte bislang noch nicht ausreichend unterstützt werden [117]. Hierbei sind offene Problemstellungen skizziert worden, die im folgenden Abschnitt zusammengefasst werden.

## 8.2 Ausblick

Die in Abschnitt 4.4 vorgestellte prototypische Realisierung einer Web-basierten Service-Auswahl zeigt die Benutzbarkeit der qualitätsbasierten Service-Auswahl auf. Unter Berücksichtigung der in Abschnitt 5.2 diskutierten Aspekte der Service-Auswahl in einer serviceorientierten Umgebung lassen sich weitere Anwendungen entwickeln. Bei der Geschäftsprozessmodellierung kann beispielsweise ein Modellierer bei der Auswahl eines geeigneten Services durch eine Anwendung unterstützt werden. Zusätzlich ist die Basis geschaffen, mit der sich eine automatisier-

te Service-Auswahl, gegebenenfalls sogar zur Laufzeit eines Geschäftsprozesses, realisieren läßt. In diesem Kontext kann das Qualitätsmodell und die Umsetzung in einer relationalen Datenbank für andere Forschungsprojekte wiederverwendet werden.

Auf der Basis der konzeptionellen Grundlage für die Modellierung von Qualitätsmerkmalen für Services können weiterführende Konzepte entwickelt werden. Die Aggregationsmuster zeigen einen erweiterbaren Ansatz auf, so dass weitere Aggregationsmustern definiert werden können oder die Aggregationsmuster beispielsweise für graphbasierte Prozessbeschreibungssprachen angepasst werden können.

Die Beschreibung der Funktionalität eines Services ist eine offene Problemstellung in der Forschung. Zur Spezifikation einer solchen Semantik werden Techniken aus dem Bereich der Ontologien und logikbasierte Beschreibungssprachen wie Description Logic [8] vorgeschlagen, so dass die Bedeutung eines Services maschinenverständlich beschrieben werden kann. Wenn dies erreicht wurde, ist zu untersuchen, ob ein Bezug zwischen der Beschreibung der Funktionalität und der Beschreibung der Qualität hergestellt werden kann und welche Vorteile sich dadurch realisieren lassen.

Darüber hinaus wurden in Kapitel 6 erste Ansätze für eine automatisierte Geschäftsprozessoptimierung vorgestellt und diskutiert. Eine umfassende Unterstützung unter Berücksichtigung der Kontrollflusskonstrukte der Prozessdefinition und verschiedenen Qualitätsmerkmalen ist allerdings noch nicht erreicht worden. Das Qualitätsmodell dieser Arbeit liefert nun eine konzeptionelle Grundlage für Qualitätsmerkmale und es ist gezeigt worden, welche Aggregationsmuster berücksichtigt werden können. Es ist nun zu untersuchen, ob sich Optimierungsverfahren für Geschäftsprozesse auf Basis des Qualitätsmodells entwickeln lassen.

Eine Umsetzung des Qualitätsmodells auf der Basis von Web Services-Technologie stellt einen weiteren Forschungsbereich dar. Dazu kann die Spezifikation der WSDL in der anstehenden Version 2.0 [181] angewendet werden, da dort voraussichtlich Entitäten wie *Property* und *Feature* spezifiziert werden. Es ist zu untersuchen, wie diese Entitäten für die Definition von Qualitätsmerkmalen benutzt werden können. Wenn die Web Service-Beschreibung um Qualitätsmerkmale erweitert wird, kann eine Anfragesprache formuliert werden, die Qualitätsmerkmale unterstützt. Hier ist unter anderem zu untersuchen, inwieweit sich die Konzepte und Anforderungen für die (Web) Service-Auswahl mittels XQuery [182] realisieren lassen, wenn die Web Service-Beschreibung durch XML-Dokumente erfolgt.

## 8.3 Fazit

Das Qualitätsmodell dieser Arbeit liefert die konzeptionelle Grundlage für die Modellierung von Qualitätsmerkmalen für Services. Unter verschiedenen, funktional äquivalenten Services kann derjenige ausgewählt werden, dessen Qualität den eigenen Anforderungen am besten entspricht und es kann eine Rangfolge passender

Service erstellt werden. Die Qualität eines Services kann präzise und eindeutig spezifiziert werden, so dass qualitativ unterschiedliche Services angeboten werden können. Die Qualität von Services über einen Geschäftsprozess kann aggregiert werden, so dass qualitative Aussagen über einen Geschäftsprozess ebenso möglich sind, wie eine Optimierung der Qualität.

# Literaturverzeichnis

- [1] ADAM, Dietrich: *Produktions-Management*. 7. Auflage. Wiesbaden : Gabler, 1993
- [2] AGARWAL, Vikas ; DASGUPTA, Koustuv ; KARNIK, Neeran ; KUMAR, Arun ; KUNDU, Ashish ; MITTAL, Sumit ; SRIVASTAVA, Biplav: A service creation environment based on end to end composition of Web services. In: *Proceedings of the 14th International Conference on World Wide Web (WWW '05)*. New York, NY, USA : ACM Press, 2005, S. 128–137
- [3] AGGARWAL, Rohit ; VERMA, Kunal ; MILLER, John ; MILNOR, William: Constraint driven Web service composition in METEOR-S. In: *Proceedings of IEEE Conference on Service Computing (SCC)*. Shanghai, China : IEEE, 2004
- [4] ALONSO, Gustavo ; CASATI, Fabio ; KUNO, Harumi ; MACHIRAJU, Vijay: *Web Services - Concepts, Architectures and Applications*. Springer Verlag, 2004
- [5] ALVES, Alexandre ; ARKIN, Assaf ; ASKARY, Sid ; BLOCH, Ben ; CURBERA, Francisco ; FORD, Mark ; GOLAND, Yaron ; GUÍZAR, Alejandro ; KARTHA, Neelakantan ; LIU, Canyang K. ; KHALAF, Rania ; KÖNIG, Dieter ; MARIN, Mike ; MEHTA, Vinkesh ; THATTE, Satish ; RIJN, Danny van d. ; YENDLURI, Prasad ; YIU, Alex: *Web Services Business Process Execution Language Version 2.0*. Version: August 2006. <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-specification-draft.pdf>. – Online-Ressource. – Public Review Draft, 23rd August, 2006
- [6] ANDERSON, Thomas ; LEE, Peter A.: *Fault Tolerance. Principles and Practice*. Wien : Springer Verlag, 1990
- [7] ARSANJANI, Ali: *Service-oriented modeling and architecture - How to identify, specify, and realize services for your SOA*. Version: November 2004. <http://www.ibm.com/developerworks/webservices/library/ws-soa-design1/>. – Online-Ressource



- [8] BAADER, Franz (Hrsg.) ; CALVANESE, Diego (Hrsg.) ; MCGUINNESS, Deborah (Hrsg.) ; NARDI, Daniele (Hrsg.) ; PATEL-SCHNEIDER, Peter (Hrsg.): *The Description Logic Handbook - Theory, Implementation and Applications*. Cambridge University Press, 2003
- [9] BACKHAUS, Klaus: *Industriegütermarketing*. 5. Auflage. München : Verlag Franz Vahlen, 1997
- [10] BALLINGER, Keith ; EHNEBUSKE, David ; GUDGIN, Christopher Ferrisand M. ; LIU, Canyang K. ; NOTTINGHAM, Mark ; YENDLURI, Prasad: *Web Services-Interoperability (WSI) – Basic Profile Version 1.1*. Version: August 2004. <http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html>. – Online-Ressource
- [11] BALZERT, Helmut: *Lehrbuch der Software-Technik*. 2. Auflage. Spektrum Akademischer Verlag, 2001
- [12] BAMBERG, Günter ; COENENBERG, Adolf G.: *Betriebswirtschaftliche Entscheidungslehre*. 13., überarb. Auflage. Verlag Vahlen, 2006
- [13] BARBACCI, Mario ; KLEIN, Mark H. ; LONGSTAFF, Thomas A. ; WEINSTOCK, Charles B.: *Quality attributes*. Version: December 1995. <http://www.sei.cmu.edu/publications/documents/95.reports/95.tr.021.html>. (CMU/SEI-95-TR-021). – Online-Ressource. – Technical Report
- [14] BENATALLAH, Boualem ; SHENG, Quan Z. ; DUMAS, Marlon: The Self-Serv Environment for Web Services Composition. In: *IEEE Internet Computing* 7 (2003), Jan/Feb, Nr. 1, S. 40–48
- [15] BERNERS-LEE, T. ; FIELDING, R. ; MASINTER, L.: *Uniform Resource Identifiers (URI): Generic Syntax*. Version: August 1998. <http://www.ietf.org/rfc/rfc2396.txt>. – Online-Ressource. – IETF RFC 2396
- [16] BERTOLINO, Antonia ; MIRANDOLA, Raffaella: Modeling and Analysis of Non-functional Properties in Component-based Systems. In: PEZZÈ, Mauro (Hrsg.): *International Workshop on Test and Analysis of Component-Based Systems (TACoS'03)* Bd. 82, Elsevier Science B.V., September 2003
- [17] BIRRELL, Andrew D. ; NELSON, Bruce J.: Implementing Remote Procedure Calls. In: *ACM Transactions on Computer Systems* 2 (1998), Nr. 1, S. 39 – 59
- [18] BLIEMEL, Friedhelm ; FILLIP, Stefan: Gestaltung der Produktqualität aus Kundensicht. In: WILDEMAN, Horst (Hrsg.): *Controlling im TQM - Methoden und Instrumente zur Verbesserung der Unternehmensqualität*. Berlin : Springer Verlag, 1996, S. 65–97

- 
- [19] BLOOMBERG, Jason: *How loose is your coupling?* Version: Sept. 2004. <http://www.zapthink.com/>. – Online-Ressource. – ZapFlash 09292004
- [20] BLOOMBERG, Jason: *No, Virginia, there is no SOA Wizard.* Version: Jan. 2005. <http://www.zapthink.com/>. – Online-Ressource. – ZapFlash 2005110
- [21] BLUM, Ulrich: *Volkswirtschaftslehre*. 5. Auflage. München : Oldenbourg, 1994
- [22] BOSWORTH, Adam: *Keynote Talk at the 2nd International Conference on Service Oriented Computing (ICSOC04)*. Version: November 2004. [http://www.adambosworth.net/archives/2004\\_11.html](http://www.adambosworth.net/archives/2004_11.html). – Online-Ressource
- [23] BRAY, Tim ; PAOLI, Jean ; SPERBERG-MCQUEEN, C. M. ; MALER, Eve ; YERGEAU, François: *Extensible Markup Language (XML) 1.0 (Third Edition)*. Version: February 2004. <http://www.w3.org/TR/2004/REC-xml-20040204>. – Online-Ressource
- [24] BRONSTEIN, Ilja N. ; SEMENDJAJEW, Konstantin A. ; MUSIOL, Gerhard ; MÜHLIG, Heiner: *Taschenbuch der Mathematik*. 5. Auflage. Verlag Harry Deutsch, 2000
- [25] BURBECK, Steve: *The Tao of e-business services*. Version: October 2000. <http://www-106.ibm.com/developerworks/library/ws-tao/>. IBM Corporation. – Online-Ressource
- [26] BUSCHMANN, Frank ; MEUNIER, Regine ; ROHNERT, Hans ; SOMMERLAD, Peter ; STAL, Michael: *Pattern-Oriented Software Architecture, A System of Patterns*. Chichester, England : John Wiley & Sons Ltd, Chichester, England, 1996
- [27] BUSSLER, Christoph: *B2B Integration - Concepts and Architecture*. Springer, 2003
- [28] CAPPIELLO, C. ; PERNICI, Barbara ; PLEBANI, Pierluigi: Quality-agnostic or quality-aware semantic service descriptions? In: *W3C Workshop on Semantic Web Service Framework*. W3C
- [29] CAPPIELLO, Cinzia ; MISSIER, Paolo ; PERNICI, Barbara ; PLEBANI, Pierluigi ; BATINI, Carlo: QoS in Multichannel IS: The MAIS Approach. In: MATERA, Maristella (Hrsg.) ; COMAI, Sara (Hrsg.): *ICWE Workshops*, Rinton Press, 2004, S. 255–268
- [30] CARDOSO, Jorge ; SHETH, Amit P. ; MILLER, John A.: Workflow Quality of Service. In: KOSANKE, Kurt (Hrsg.) ; JOCHEM, Roland (Hrsg.) ; NELL,

- James G. (Hrsg.) ; BAS, Angel O. (Hrsg.): *ICEIMT* Bd. 236, Kluwer, 2002, S. 303–311
- [31] CARDOSO, Jorge ; SHETH, Amit P. ; MILLER, John A. ; ARNOLD, Jonathan ; KOCHUT, Krys: Quality of service for workflows and web service processes. In: *Elsevier Journal of Web Semantics* 1 (2004), Nr. 3, S. 281–308
- [32] CASATI, Fabio ; SHAN, Eric ; DAYAL, Umeshwar ; SHAN, Ming-Chien: Business-oriented management of Web services. In: *Communications of the ACM* 46 (2003), October, Nr. 10, S. 55–60
- [33] CHEN, Peter Pin-Shan S.: The Entity-Relationship Model: Toward a Unified View of Data. In: *ACM Transactions on Database Systems* 1 (1976), Nr. 1, S. 9–36
- [34] CHERBAKOV, Luba ; GALAMBOS, George ; HARISHANKAR, Roy ; KALYANA, Shankar ; RACKHAM, Guy: Impact of service orientation at the business level. In: *IBM Systems Journal* 44 (2005), Nr. 4, S. 653–668
- [35] CLEMENT, Luc ; HATELY, Andrew ; VON RIEGEN, Claus ; ROGERS, Tony: *UDDI Version 3.0.2 – UDDI Spec Technical Committee Draft, Dated 20041019*. Version: Okt. 2004. [http://uddi.org/pubs/uddi\\_v3.htm](http://uddi.org/pubs/uddi_v3.htm). – Online-Ressource
- [36] COX, David E. ; KREGER, Heather: Management of the service-oriented-architecture life cycle. In: *IBM Systems Journal* 44 (2005), Nr. 4, S. 709–726
- [37] CURBERA, Francisco ; KHALAF, Rania ; MUKHI, Nirmal ; TAI, Stefan ; WEERAWARANA, Sanjiva: The next step in Web services. In: *Communications of the ACM* 46 (2003), October, Nr. 10, S. 29–34
- [38] DALAL, Sanjay ; TEMEL, Sazi ; LITTLE, Mark ; POTTS, Mark ; WEBBER, Jim: Coordinating Business Transactions on the Web. In: *IEEE Internet Transactions* 7 (2003), Jan/Feb, Nr. 1, S. 30–39
- [39] DALAL, Sanjay ; TEMEL, Sazi ; LITTLE, Mark ; POTTS, Mark ; WEBBER, Jim: Coordinating Business Transactions on the Web. In: *IEEE Internet Transactions* 7 (2003), Jan/Feb, Nr. 1, S. 30–39
- [40] DAN, Asit ; LUDWIG, Heiko ; PACIFICI, Giovanni: *Web Services Differentiation with Service Level Agreements*. Version: May 2003. <http://www-106.ibm.com/developerworks/library/ws-slafram/>. – Online-Ressource
- [41] DORNBUSCH, Rüdiger ; FISCHER, Stanley: *Makroökonomik*. 5. Auflage. München : Oldenbourg, 1992
- [42] ELFATATRY, Ahmed ; LAYZELL, Paul: Negotiating in Service-Oriented Environments. In: *Communications of the ACM* 47 (2004), Nr. 8, S. 103–108

- 
- [43] ELLIS, Clarence ; KEDDARA, Karim ; ROZENBERG, Grzegorz: Dynamic change within workflow systems. In: *Proceedings of conference on Organizational computing systems*, ACM Press, 1995, S. 10–21
- [44] FAHRMEIR, Ludwig (Hrsg.) ; HAMERLE, Alfred (Hrsg.) ; TUTZ, Gerhard (Hrsg.): *Multivariate statistische Verfahren*. 2. Auflage. de Gruyter, 1996
- [45] FEINGOLD, Max (Hrsg.): *Web Services Coordination (WS-Coordination) 1.1*. Version: March 2006. <http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-cd-01.pdf>. (wstx-wscoor-1.1-spec-cd-01). – Online-Ressource
- [46] FENSEL, D. ; BUSSLER, C. ; DING, Y. ; OMELAYENKO, B.: The Web Service Modeling Framework WSMF. In: *Electronic Commerce Research and Applications* 1 (2002), Nr. 2
- [47] FENSEL, Dieter: *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*. 2. Berlin : Springer-Verlag, 2003
- [48] FIROIU, Victor ; BOUDEC, Jean-Yves L. ; TOWSLEY, Don ; ZHANG, Zhi-Li: Theories and Models for Internet Quality of Service. In: *Proceedings of IEEE, special issue on Internet Technology* (2002)
- [49] FISCHER, Hans R. ; PESCHL, Markus: Konstruktivismus (constructivism). In: STRUBE, Gerhard (Hrsg.) ; BECKER, Barbara (Hrsg.) ; FRESKA, Christian (Hrsg.) ; HAHN, Udo (Hrsg.) ; OPWIS, Klaus (Hrsg.) ; PALM, Günther (Hrsg.): *Wörterbuch der Kognitionswissenschaft*. Stuttgart : Klett-Cotta, 1996, S. 329–331
- [50] FREUND, Tom ; GREEN, Alastair ; HARBY, John ; LITTLE, Mark: *Web Services Business Activity (WS-Business Activity) 1.1*. Version: March 2006. <http://docs.oasis-open.org/ws-tx/wstx-wsba-1.1-spec-cd-01.pdf>. (wstx-wsba-1.1-spec-cd-01). – Online-Ressource
- [51] FRØLUND, Svend ; KOISTINEN, Jari: Quality of Service Aware Distributed Object Systems. In: *5th USENIX Conference on Object-Oriented Technologies and Systems (COOTS '99)*, USENIX, San Diego, California, USA
- [52] FRØLUND, Svend ; KOISTINEN, Jari: Quality-of-Service Specification in Distributed Object Systems. In: *Distributed Systems Engineering Journal* 5 (1998), December, Nr. 4
- [53] *Gabler Wirtschafts-Lexikon*. 16. Auflage. Wiebaden : Dr. Th. Gabler Verlag, 2005
- [54] GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995

- [55] GEORGAKOPOULOS, Dimitrios ; HORNICK, Mark F. ; SHETH, Amit P.: An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. In: *Distributed and Parallel Databases 3* (1995), Nr. 2, S. 119–153
- [56] GILLIES, Alan C.: *Software Quality: Theory and Management*. London : Chapman & Hall, 1992
- [57] GILLMANN, Michael ; WEIKUM, Gerhard ; WONNER, Wolfgang: Workflow management with service quality guarantees. In: FRANKLIN, Michael J. (Hrsg.) ; MOON, Bongki (Hrsg.) ; AILAMAKI, Anastassia (Hrsg.): *SIGMOD Conference*, ACM, 2002, S. 228–239
- [58] GRALLA, Preston: *BPM and Web services: A perfect Match ?* Version: Oct. 2004. <http://searchwebservices.techtarget.com/>. – Online-Ressource
- [59] GUARINO, Nicola: Formal Ontology and Information Systems. In: *Proceedings of FOIS'98*,. Amsterdam : IOS Press, 1998, S. 3–15
- [60] HAMMER, Michael ; CHAMPY, James: *Reengineering the corporation*. New York, USA : Harper Collins Publishing, 1993
- [61] HAVEY, Michael: *Essential Business Process Management*. O'Reilly Media, Inc., 2005
- [62] HERWIG, Volker: UDDI in der Praxis. In: *Java Spectrum* (2004), Nov., Nr. 53, S. 34–36
- [63] HOLLINGSWORTH, David: *The Workflow Reference Model*. Version: Jan 1999. <http://www.wfmc.org/standards/docs/tc003v11.pdf>. Workflow Management Coalition. – Online-Ressource. – Document Number WFMC-TC-1003
- [64] HOPCROFT, John E. ; ULLMAN, Jeffrey: *Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie*. 3. Bonn : Addison-Wesley, 1994
- [65] HORN, Stefan ; JABLONSKI, Stefan: An Approach to Dynamic Instance Adaptation in Workflow Management Applications. In: *Conference on Computer Supported Cooperative Work (CSCW)*, 1998
- [66] HWANG, San-Yih ; WANG, Haojun ; SRIVASTAVA, Jaideep ; PAUL, Raymond A.: A Probabilistic QoS Model and Computation Framework for Web Services-Based Workflows. In: ATZENI, Paolo (Hrsg.) ; CHU, Wesley W. (Hrsg.) ; LU, Hongjun (Hrsg.) ; ZHOU, Shuigeng (Hrsg.) ; LING, Tok W. (Hrsg.): *ER*, Springer, 2004 (LNCS 3288), S. 596–609

- [67] IEEE: *Recommended Practice for Architectural Description of Software-Intensive Systems*. Version: 2000. <http://standards.ieee.org/reading/ieee/std/se/1471-2000.pdf>. (1741-2000). – Online-Ressource
- [68] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO): *ISO 9000 2000*. <http://www.iso.org/>. Version: 2000
- [69] JABLONSKI, Stefan ; BUSSLER, Christoph: *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, 1996
- [70] JAEGER, Michael C. ; MÜHL, Gero ; GOLZE, Sebastian: QoS-aware Composition of Web Services: An Evaluation of Selection Algorithms. In: MEERSMAN, Robert (Hrsg.) ; TARI, Zahir (Hrsg.): *Proceedings of the Confederated International Conferences CoopIS, DOA, and ODBASE 2005 (OTM'05)*. Agia Napa, Cyprus : Springer Press, November 2005 (LNCS 3760), S. 646–661
- [71] JAEGER, Michael C. ; ROJEC-GOLDMANN, Gregor ; MÜHL, Gero: QoS Aggregation in Web Service Compositions. In: *2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE-05)*. Hong Kong, China : IEEE Press, March 2005
- [72] JAEGER, Michael C. ; ROJEC-GOLDMANN, Gregor ; MÜHL, Gero: QoS Aggregation for Service Composition using Workflow Patterns. In: *Proceedings of the 8th International Enterprise Distributed Object Computing Conference (EDOC 2004)*. IEEE CS Press, 149–159
- [73] JIN, Li-jie ; MACHIRAJU, Vijay ; SAHAI, Akhil: *Analysis on Service Level Agreement of Web Services*. Version: June 2002. <http://www.hpl.hp.com/techreports/2002/HPL-2002-180.pdf>. (HPL-2002-180). – Online-Ressource. – Technical Report
- [74] JONES, Steve: Toward an Acceptable Definition of Service. In: *IEEE Software* 22 (2005), May, Nr. 3, S. 87–93
- [75] KAMISKE, Gerd F. ; BRAUER, Jörg-Peter: *Qualitätsmanagement von A bis Z – Erläuterungen moderner Begriffe des Qualitätsmanagements*. 5. Auflage. Carl Hanser <http://standards.ieee.org/reading/ieee/std/se/1471-2000.pdf>
- [76] KANO, Makoto ; KOIDE, Akio ; LIU, Te-Kai ; RAMACHANDRAN, Bala: Analysis and simulation of business solutions in a serviceoriented architecture. In: *IBM Systems Journal* 44 (2005), Nr. 4, S. 669–690

- [77] KAVANTZAS, Nickolas ; BURDETT, David ; RITZINGER, Gregory ; FLETCHER, Tony ; LAFON, Yves ; BARRETO, Charlton: *Web Services Choreography Description Language Version 1.0*. Version: November 2005. <http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/>. – Online-Ressource. – W3C Candidate Recommendation 9 November 2005
- [78] KIM, Eunju ; LEE, Youngkon: *Quality Model for Web Services Version 2.0*. Version: September 2005. <http://www.oasis-open.org/committees/download.php/15910/WSQM-ver-2.0.doc>. Organization for the Advancement of Structured Information Standards (OASIS). – Online-Ressource. – Working Draft
- [79] KRAFZIG, Dirk ; BANKE, Karl ; SLAMA, Dirk: *Enterprise SOA : Service-Oriented Architecture Best Practices*. Prentice Hall, 2004
- [80] KRÄMER, Bernd J. ; PAPAOGLOU, Michael P. ; CUBERA, Francisco: Abstracts Collection – Service Oriented Computing (SOC). In: CUBERA, Francisco (Hrsg.) ; KRÄMER, Bernd J. (Hrsg.) ; PAPAOGLOU, Michael P. (Hrsg.): *Service Oriented Computing (SOC)*, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2006 (Dagstuhl Seminar Proceedings 05462)
- [81] LEA, Doug ; VINOSKI, Steve: Middleware for Web Services. In: *IEEE Internet Computing* 7 (2003), Jan/Feb, Nr. 1, S. 28–29
- [82] LEYMANN, Frank: *Web Services Flow Language (WSFL 1.0)*. Version: May 2001. <http://www-306.ibm.com/software/solutions/webservices/pdf/WSFL.pdf>. IBM Software Group. – Online-Ressource
- [83] LEYMANN, Frank ; ROLLER, Dieter: *Production Workflow: Concepts and Techniques*. Prentice-Hall, 2000
- [84] LEYMANN, Frank ; ROLLER, Dieter ; SCHMIDT, M.-T.: Web services and business process management. In: *IBM Systems Journal* 41 (2002), Nr. 2
- [85] LIGGESMEYER, Peter: *Software-Qualität*. Heidelberg, Berlin : Spektrum-Verlag, 2002
- [86] LINTHICUM, David S.: *B2B Application Integration: e-Business-Enable Your Enterprise*. Addison-Wesley, 2001 (Addison-Wesley Information Technology Series)
- [87] LITOIU, Marin: Migrating to web services: a performance engineering approach. In: *Journal of Software Maintenance and Evolution: Research and Practice* 16 (2004), Nr. 1-2, S. 51–70

- [88] LITTLE, Mark (Hrsg.) ; WILKINSON, Andrew (Hrsg.): *Web Services Atomic Transaction (WS-AtomicTransaction) 1.1*. Version: March 2006. <http://docs.oasis-open.org/ws-tx/wstx-wsat-1.1-spec-cd-01.pdf>. (wstx-wsat-1.1-spec-cd-01). – Online-Ressource
- [89] LITTLE, Mark: Transactions and Web services. In: *Communications of the ACM* 46 (2003), October, Nr. 10, S. 49– 54
- [90] LIU, Yutu ; NGU, Anne ; ZHENG, Liangzhao: Qos computation and policing in dynamic web service selection. In: *Proceedings of the WWW 2004*, ACM, 2004, S. 66–73
- [91] LUDWIG, Heiko: Web Services QoS: External SLAs and Internal Policies - Or: How do we deliver what we promise? In: *Keynote Speech at the WISE Workshop on Web Services Quality , Rome, Italy, (WISE 2003)*
- [92] LUDWIG, Heiko ; DAN, Asit ; KEARNEY, Robert: Cremona: An Architecture and Library for Creation and Monitoring of WS-Agreements. In: AIELLO, M. (Hrsg.) ; AOYAMA, M. (Hrsg.) ; CURBERA, F. (Hrsg.) ; PAPA-ZOGLOU, M. (Hrsg.): *Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC 2004)*. New York, USA : ACM Press, December 2004, S. 65–74
- [93] LUDWIG, Heiko ; KELLER, Alexander ; DAN, Asit ; KING, Richard P. ; FRANCK, Richard: *Web Service Level Agreement (WSLA) Language Specification*. Version: January 2003. <http://www.research.ibm.com/wsla>. – Online-Ressource
- [94] MACKENZIE, C. M. ; LASKEY, Ken ; MCCABE, Francis ; BROWN, Peter ; METZ, Rebekah: *Reference Model for Service Oriented Architectures*. Version: December 2005. [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=soa-rm](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm). Organization for the Advancement of Structured Information Standards (OASIS). – Online-Ressource. – Working Draft 11
- [95] MARCHETTI, Carlo ; PERNICI, Barbara ; PLEBANI, Pierluigi: A Quality Model for Multichannel Adaptive Information Systems. In: *Alternate Tracks Proceedings of 13th International World Wide Web Conference (WWW2004)*. New York City, NY, USA : ACM Press, 2004, S. 48–54
- [96] MARGARIA, Tiziana ; STEFFEN, Bernhard: Service Engineering: Linking Business and IT. In: *IEEE Computer* 39 (2006), Nr. 10, S. 45–55
- [97] MARTENS, Axel: Modeling Workflow in Virtual Enterprises. In: WEBER, H. (Hrsg.) ; EHRIG, H. (Hrsg.) ; REISIG, W. (Hrsg.): *2nd International Colloquium on Petri Net Technologies for Modelling Communication Based Systems*. Berlin, September 2001



- [98] MEFFERT, Heribert: *Marketing*. 9. Auflage. Dr. Th. Gabler Verlag, 2000
- [99] MEYER, Harald: *Entwicklung und Realisierung einer Planungskomponente für die Komposition von Diensten*. 2005. – Diplomarbeit, Hasso-Plattner-Institut für Softwaresystemtechnik, Universität Potsdam
- [100] MILANOVIC, Nikola ; MALEK, Miroslaw: Current Solutions for Web Service Composition. In: *IEEE Internet Computing* 08 (2004), Nr. 6, S. 51–59
- [101] MILNER, Robin: *Communicating and mobile systems: the Pi-calculus*. Cambridge University Press, 1999
- [102] MOHAN, C.: Dynamic E-business: Trends in Web Services. In: BUCHMANN, A. (Hrsg.) ; CASATI, F. (Hrsg.) ; FIEGE, L. (Hrsg.) ; HSU, M.-C. (Hrsg.) ; SHAN, M.-C. (Hrsg.): *Proceedings of the third VLDB workshop on Technologies for E-Services, TES 2002*. Hong Kong, China : Springer Verlag, 2002 (LNCS 2444)
- [103] NADALIN, Anthony (Hrsg.) ; KALER, Chris (Hrsg.) ; MONZILLO, Ronald (Hrsg.) ; HALLAM-BAKER, Phillip (Hrsg.): *Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)*. Version: February 2006. <http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>. – Online-Ressource. – OASIS Standard Specification
- [104] OBJECT MANAGEMENT GROUP: *Homepage*. <http://www.omg.org/>. – Online-Ressource
- [105] OBJECT MANAGEMENT GROUP: *MetaObjectFacility(MOF) Specification – Version 1.4*. Version: April 2002. <http://www.omg.org/docs/formal/02-04-03.pdf>. – Online-Ressource
- [106] OBJECT MANAGEMENT GROUP: *UML™ Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms*. Version: June 2004. <http://www.omg.org/docs/ptc/04-06-01.pdf>. – Online-Ressource. – OMG Adopted Specification
- [107] OBJECT MANAGEMENT GROUP: *Business Process Modeling Notation Specification*. Version: February 2006. <http://www.omg.org/docs/dtc/06-02-01.pdf>. – Online-Ressource. – OMG Final Adopted Specification, 21th Dec. 2005
- [108] ORACLE CORPORATION: *Homepage of Oracle Application Express (APEX)*. <http://apex.oracle.com>. – Online-Ressource
- [109] ORACLE CORPORATION: *Homepage of Oracle JDeveloper*. [http://www.oracle.com/tools/jdev\\_home.html](http://www.oracle.com/tools/jdev_home.html). – Online-Ressource

- [110] ORGANIZATION FOR THE ADVANCEMENT OF STRUCTURED INFORMATION STANDARDS (OASIS): *Homepage*. <http://www.oasis-open.org/>. – Online-Ressource
- [111] O’SULLIVAN, Justin ; EDMOND, David ; TER HOFSTEDE, Arthur: What’s in a Service? Towards Accurate description of Non-Functional Service Properties. In: *Distributed and Parallel Databases* 12 (2002), S. 117–133
- [112] O’SULLIVAN, Justin ; EDMOND, David ; TER HOFSTEDE, Arthur H. M.: *Formal description of non-functional service properties*. Version: 2005. <http://www.service-description.com/>. (FIT-TR-2005-01). – Online-Ressource. – Technical Report
- [113] OVERDICK, Hagen: *Entwurf und Implementierung von Prozesssteuerungskomponenten in service-orientierte Umgebungen*. 2005. – Master-Arbeit, Hasso-Plattner-Institut für Softwaresystemtechnik, Universität Potsdam
- [114] PAHL, Claus ; CASEY, Michael: Ontology support for web service processes. In: *Proceedings of the 9th European software engineering conference*, ACM Press, 2003 (Foundations of Software Engineering), S. 208 – 216
- [115] PAPAZOGLU, Michael P.: Extending the Service-Oriented Architecture. In: *Business Integration Journal* (2005), February, S. 18–21
- [116] PAPAZOGLU, Michael P. ; GEORGAKOPOULOS, Dimitros: Service-Oriented Computing. In: *Communications of the ACM* 46 (2003), October, Nr. 10, S. 25–28
- [117] PAPAZOGLU, Michael P. ; TRAVERSO, Paolo ; DUSTDAR, Schahram ; LEYMAN, Frank ; KRÄMER, Bernd J.: Service-Oriented Computing: A Research Roadmap. In: CUBERA, Francisco (Hrsg.) ; KRÄMER, Bernd J. (Hrsg.) ; PAPAZOGLU, Michael P. (Hrsg.): *Service Oriented Computing (SOC)*, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2006 (Dagstuhl Seminar Proceedings 05462)
- [118] PAPAZOGLU, Michael P. ; VAN DEN HEUVEL, Willem-Jan: Web Services Management: A Survey. In: *IEEE Internet Computing* 9 (2005), Nov/Dec, Nr. 6, S. 58–64
- [119] PAPAZOGLU, Mike P. ; KRÄMER, Bernd J. ; YANG, Jian: Leveraging Web-Services and Peer-to-Peer Networks. In: EDER, J. (Hrsg.) ; MISSIKO, M. (Hrsg.): *Proceedings of the CAiSE 2003* Bd. 2681. Berlin Heidelberg : Springer-Verlag, 2003, S. 485–501
- [120] PASLEY, James: How BPEL and SOA Are Changing Web Services Development. In: *IEEE Internet Computing* 09 (2005), May/June, Nr. 3, S. 60 – 67

- [121] PATEL, Chintan ; SUPEKAR, Kaustubh ; LEE, Yugyung: A QoS Oriented Framework for Adaptive Management of Web Service Based Workflows. In: *DEXA*, 2003, S. 826 – 835
- [122] PATIL, Abhijit A. ; OUNDHAKAR, Swapna A. ; SHETH, Amit P. ; VERMA, Kunal: METEOR-S Web Service Annotation Framework. In: FELDMAN, Stuart I. (Hrsg.) ; URETSKY, Mike (Hrsg.) ; NAJORK, Marc (Hrsg.) ; WILLS, Craig E. (Hrsg.): *WWW*, ACM, 2004. – ISBN 1–58113–844–X, S. 553–562
- [123] PENDER, Tom: *UML Bible*. Wiley Publishing, Inc., 2003
- [124] PUHLMANN, Frank: Why do we actually need the Pi-Calculus for Business Process Management? In: ABRAMOWICZ, W. (Hrsg.) ; MAYR, H. (Hrsg.): *9th International Conference on Business Information Systems (BIS 2006)*. Bonn : Gesellschaft für Informatik, 2006 (LNI P-85), S. 77–89
- [125] PUHLMANN, Frank ; WESKE, Mathias: Using the Pi-Calculus for Formalizing Workflow Patterns. In: VAN DER AALST, Wil M. P. (Hrsg.) ; BENATALLAH, Boualem (Hrsg.) ; CASATI, Fabio (Hrsg.) ; CURBERA, Francisco (Hrsg.): *Proceedings of the 3rd International Conference on Business Process Management (BPM 2005)*, Springer-Verlag, 2005 (LNCS 3649), S. 153–168
- [126] RAN, Shuping: A model for web services discovery with QoS. In: *ACM SIGecom Exchanges* 4 (2003), Nr. 1, S. 268 – 277
- [127] REICHERT, Manfred ; DADAM, Peter: ADEPT<sub>flex</sub> -Supporting Dynamic Changes of Workflows Without Losing Control. In: *Journal of Intelligent Information Systems* 10 (1998), Nr. 2, S. 93–129
- [128] REITER, Michael: Serviceorientierung flexibilisiert Software. In: *Computerzeitung* 35 (2004), Nr. 44, S. 16
- [129] RICHTER, Jan-Peter ; HALLER, Harald ; SCHREY, Peter: Serviceorientierte Architekturen. In: *Informatik Spektrum* 28 (2005), Oktober, Nr. 5, S. 413–416
- [130] SABATA, Bikash ; CHATTERJEE, Saurav ; DAVIS, Michael ; SYDIR, Jaroslaw J. ; LAWRENCE, Thomas F.: Taxonomy for QoS specifications. In: *Workshop on Object-Oriented Real-Time Dependable Systems (WORDS)*, IEEE Computer Society
- [131] SAHAI, Akhil ; MACHIRAJU, Vijay ; SAYAL, Mehmet ; VAN MOORSEL, Aad ; CASATI, Fabio: Automated SLA Monitoring for Web Services. In: FERIDUN, M. (Hrsg.) ; KROPF, P. (Hrsg.) ; BABIN, G. (Hrsg.): *13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM*, 2002 (LNCS 2506), S. 28 – 41

- [132] SCHMIDT, Marc-Thomas ; HUTCHISON, Beth ; LAMBROS, Peter ; PHIPPEN, Rob: The Enterprise Service Bus: Making service-oriented architecture real. In: *IBM Systems Journal* 44 (2005), Nr. 4, S. 781–797
- [133] SCHÜTTE, Reinhard: *Grundsätze ordnungsmäßiger Referenzmodellierung*. Dr. Th. Gabler Verlag, 2001
- [134] SCHUBERT, Harald: *Entwicklung eines QoS-Frameworks für Service-orientierte Architekturen*. Oktober 2005. – Master-Arbeit, Hasso-Plattner-Institut für Softwaresystemtechnik, Universität Potsdam
- [135] SCHUSCHEL, Hilmar ; WESKE, Mathias: Integrated Workflow Planning and Coordination. In: *14th International Conference on Database and Expert Systems Applications (DEXA'03)*. Prague, Czech Republic : Springer, 2003 (LNCS 2736), S. 771–781
- [136] SCHUSCHEL, Hilmar ; WESKE, Mathias: Automated Planning in a Service-Oriented Architecture. In: *Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*, IEEE Computer Society Press, 2004, S. 75–80
- [137] SCHUSCHEL, Hilmar ; WESKE, Mathias: Triggering Replanning in an Integrated Workflow Planning and Enactment System. In: *8th East-European Conference on Advances in Databases and Information Systems (AD-BIS'04)*, Springer, 2004 (LNCS 3255)
- [138] SHAIKH ALI, Ali ; RANA, Omer F. ; AL-ALI, Rashid J. ; WALKER, David W.: UDDIe: An Extended Registry for Web Service. In: *SAINT Workshops*, IEEE Computer Society, 2003, S. 85–89
- [139] SINGH, Munindar P. ; HUHS, Michael N.: *Service-oriented Computing – Semantics, Processes, Agents*. John Wiley & Sons, Ltd, 2005
- [140] SMITH, Howard ; FINGAR, Peter: *Business Process Management: The Third Wave*. Meghan-Kiffer Press, 2002
- [141] SMITH, Michael K. ; WELTY, Chris ; MCGUINNESS, Deborah L.: *OWL Web Ontology Language Guide*. Version: 2004. <http://www.w3.org/2004/OWL/>. – Online-Ressource. – W3C Recommendation 10 February 2004
- [142] SNEED, Harry M.: Integrating legacy Software into a Service oriented Architecture. In: LEHNER, F. (Hrsg.) ; NÖSEKABEL, H. (Hrsg.) ; KLEIN-SCHMIDT, P. (Hrsg.): *Multikonferenz Wirtschaftsinformatik 2006, Band 2, XMLABPM Track*. Berlin : GITO-Verlag, 2006, S. 345–360. – Discussion Paper

- [143] STAAB, Steffen (Hrsg.) ; STUDER, Rudi (Hrsg.): *Handbook on Ontologies*. Springer–Verlag, 2004
- [144] STEVENS, S.: On the theory of scales of measurement. In: *Science* 103 (1946), S. 677–680
- [145] TAI, Stefan ; MIKALSEN, Thomas ; WOHLSTADTER, Eric ; DESAI, Nirmal ; ROUVELLOU, Isabelle: Transaction policies for service-oriented computing. In: *Data & Knowledge Engineering* 51 (2004), Nr. 1, S. 59–79
- [146] TANENBAUM, Andrew S.: *Computer networks*. Upper Saddle River, NJ, USA : Pearson, 2003
- [147] THATTE, Satish: *XLANG – Web Services for Business Process Design*. Version: 2001. [http://www.gotdotnet.com/team/xml\\_wsspecs/xlang-c/default.htm](http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm). Microsoft Corporation. – Online–Ressource
- [148] THE OWL SERVICES COALITION: *OWL-S: Semantic Markup for Web Services*. Version: 2004. <http://www.daml.org/services>. – Online–Ressource
- [149] THOME, Rainer: e-Business. In: *Informatik Spektrum* 25 (2002), April, Nr. 2, S. 151–153
- [150] TIAN, Min: *Dienstgüterintegration in Web-Services mit dem WS-QoS Rahmenwerk*. 2005. – Dissertation am Fachbereich Mathematik und Informatik, Freie Universität Berlin
- [151] TIAN, Min ; GRAMM, Andreas ; NAUMOWICZ, Tomasz ; RITTER, Hartmut ; SCHILLER, Jochen: A Concept for QoS Integration in Web Services. In: *1st Web Services Quality Workshop (WQW 2003) in conjunction with IEEE Computer Society 4th International Conference on Web Information Systems Engineering (WISE 2003)*. Rome, Italy, 2004
- [152] TIAN, Min ; GRAMM, Andreas ; RITTER, Hartmut ; SCHILLER, Jochen: A Concept for QoS Integration in Web Services. In: *IEEE/WIC/ACM International Conference on Web Intelligence (WI'04)*, 2004, S. 152–158
- [153] TIAN, Min ; GRAMM, Andreas ; RITTER, Hartmut ; SCHILLER, Jochen: A Survey of current Approaches towards Specification and Management of Quality of Service for Web Services. In: *PIK (Praxis der Informationsverarbeitung und Kommunikation, Fachzeitschrift für den Einsatz von Informationssystemen) Sonderthemenheft Web services* (2004)
- [154] TOSIC, Vladimir ; PAGUREK, Bernard ; PATEL, Kruti ; ESFANDIARI, Babak ; MA, Wei: Management applications of the Web Service Offerings Language (WSOL). In: *Inf. Syst.* 30 (2005), Nr. 7, S. 564–586

- [155] VAN DER AALST, W.M.P.: Don't go with the flow: Web services composition standards exposed. In: *IEEE Intelligent Systems* (2003), Jan./Feb.
- [156] VAN DER AALST, W.M.P. ; TER HOFSTEDÉ, A.H.M.: Workflow Patterns: On the Expressive Power of (Petri-net-based) Workflow Languages. In: JENSEN, K. (Hrsg.): *Proceedings of the Fourth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2002)* Bd. 560. Aarhus : University of Aarhus, August 2002, S. 1–20
- [157] VAN DER AALST, W.M.P. ; TER HOFSTEDÉ, A.H.M.: YAWL: Yet Another Workflow Language. In: *Information Systems* 4 (2005), Nr. 30, S. 245–275
- [158] VAN DER AALST, W.M.P. ; TER HOFSTEDÉ, A.H.M. ; KIEPUSZEWSKI, B. ; BARROS, A.P.: Workflow Patterns / Queensland University of Technology. Brisbane, 2002 (FIT-TR-2002-02). – QUT Technical report
- [159] VAN DER AALST, W.M.P. ; VAN HEE, Kees: *Workflow Management – Models, Methods, and Systems*. The MIT Press, 2002
- [160] VOSSEN, Gottfried: *Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme, 4. Auflage*. München, Wien : Oldenbourg, 2000. – ISBN 3–486–25339–5
- [161] VOSSEN, Gottfried: From Processes via Workflows to Services – An Overview. In: *Transactions of the SDPS Journal of Design & Process science* (2006)
- [162] WEB SERVICES INTEROPERABILITY ORGANIZATION: *Homepage*. Version: Februar 2006. <http://www.ws-i.org/>. – Online-Ressource
- [163] WESKE, Mathias: Object-Oriented Design of a Flexible Workflow Management System. In: *Second East-European Symposium on Advances in Databases and Information Systems ADBIS'98, Poznan Poland*. Berlin : Springer Verlag, Sept 1998 (LNCS 1475), S. 119–130
- [164] WESKE, Mathias: *Workflow Management Systems: Formal Foundation, Conceptual Design, Implementation Aspects*. Universität Münster, 2000 (Habilitationsschrift Fachbereich Mathematik und Informatik)
- [165] WESKE, Mathias: Formal Foundation and Conceptual Design of Dynamic Adaptations in a Workflow Management System. In: SPRAGUE (Hrsg.): *Proceedings of the Thirty-Fourth Annual Hawaii International Conference on System Science (HICSS-34) Minitrack Internet and Workflow Automation: Technical and Managerial Issues*. Los Alamitos : IEEE Computer Society Press, 2001

- [166] WESKE, Mathias ; HÜNDLING, Jens ; KUROPKA, Dominik ; SCHUSCHEL, Hilmar: Objektorientierter Entwurf eines flexiblen Workflow-Management-Systems. In: *Informatik, Forschung und Entwicklung* 13 (1998), Nr. 4, S. 179–195
- [167] WÖHE, Günter ; DÖRING, Ulrich: *Einführung in die allgemeine Betriebswirtschaftslehre*. 22. Auflage. München : Vahlen, 2005 (Vahlens Handbücher der Wirtschafts- und Sozialwissenschaften)
- [168] WHINSTON, Andrew B. ; CHOI, Soon-Yong ; STAHL, Dale O.: *The Economics of Electronic Commerce*. Macmillan Technical Publishing, 1997
- [169] WHITE, Stephen A.: *Process Modeling Notations and Workflow Patterns*. <http://www.bpmn.org/Documents/NotationsandWorkflowPatterns.pdf>. – Online-Ressource
- [170] WHITE, Stephen A.: *Using BPMN to Model a BPEL Process*. <http://www.bpmn.org/Documents/MappingBPMNtoBPELExample.pdf>. – Online-Ressource
- [171] WIKIQUOTE.ORG: *Albert Einstein*. <http://en.wikiquote.org/wiki/Einstein>. – Online-Ressource
- [172] WOODLEY, Thomas ; GAGNON, Stephane: BPM and SOA: Synergies and Challenges. In: NGU, Anne H. H. (Hrsg.) ; KITSUREGAWA, Masaru (Hrsg.) ; NEUHOLD, Erich J. (Hrsg.) ; CHUNG, Jen-Yao (Hrsg.) ; SHENG, Quan Z. (Hrsg.): *WISE*, Springer, 2005 (LNCS 3806), S. 679–688
- [173] WORKFLOW MANAGEMENT COALITION: *Terminology & Glossary – Issue 3.0*. Version: Feb 1999. [http://www.wfmc.org/standards/docs/TC-1011\\_term\\_glossary\\_v3.pdf](http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf). – Online-Ressource. – Document Number WFMC-TC-1011
- [174] WORLD WIDE WEB CONSORTIUM (W3C): *Homepage*. <http://www.w3c.org>. – Online-Ressource
- [175] WORLD WIDE WEB CONSORTIUM (W3C): *SOAP*. <http://www.w3.org/TR/soap/>. – Online-Ressource
- [176] WORLD WIDE WEB CONSORTIUM (W3C): *Web Services Policy 1.2 - Framework (WS-Policy)*. <http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/>. – Online-Ressource. – W3C Member Submission 25 April 2006
- [177] WORLD WIDE WEB CONSORTIUM (W3C) ; MITRA, Nilo (Hrsg.): *SOAP Version 1.2 Part 0: Primer – W3C Recommendation 24 June 2003*. Version: June 2003. <http://www.w3.org/TR/soap/>. – Online-Ressource

- [178] WORLD WIDE WEB CONSORTIUM (W3C) ; BOOTH, David (Hrsg.) ; HAAS, Hugo (Hrsg.) ; MCCABE, Francis (Hrsg.) ; NEWCOMER, Eric (Hrsg.) ; CHAMPION, Michael (Hrsg.) ; FERRIS, Chris (Hrsg.) ; ORCHARD, David (Hrsg.): *Web Services Architecture*. Version: Feb. 2004. <http://www.w3.org/TR/ws-arch/>. – Online-Ressource. – W3C Working Group Note 11 February 2004
- [179] WORLD WIDE WEB CONSORTIUM (W3C) ; HAAS, Hugo (Hrsg.) ; BROWN, Allen (Hrsg.): *Web Services Glossary - W3C Working Group Note 11 February 2004*. Version: Feb. 2004. <http://www.w3.org/TR/ws-gloss/>. – Online-Ressource
- [180] WORLD WIDE WEB CONSORTIUM (W3C) ; BOOTH, David (Hrsg.) ; LIU, Canyang K. (Hrsg.): *Web Services Description Language (WSDL) Version 2.0 Part 0: Primer*. Version: January 2006. <http://www.w3.org/TR/2006/CR-wsdl20-primer-20060106/>. – Online-Ressource. – W3C Candidate Recommendation 6 January 2006
- [181] WORLD WIDE WEB CONSORTIUM (W3C) ; CHINNICI, Roberto (Hrsg.) ; MOREAU, Jean-Jacques (Hrsg.) ; RYMAN, Arthur (Hrsg.) ; WEERAWARANA, Sanjiva (Hrsg.): *Web Services Description Language (WSDL) Version 2.0 Version 2.0 Part 1: Core Language*. Version: March 2006. <http://www.w3.org/TR/2006/CR-wsdl20-20060327/>. – Online-Ressource. – W3C Candidate Recommendation 27 March 2006
- [182] WORLD WIDE WEB CONSORTIUM (W3C) ; BOAG, Scott (Hrsg.) ; CHAMBERLIN, Don (Hrsg.) ; FERNÁNDEZ, Mary F. (Hrsg.) ; FLORESCU, Daniela (Hrsg.) ; ROBIE, Jonathan (Hrsg.) ; SIMÉON, Jérôme (Hrsg.): *XQuery 1.0: An XML Query Language*. Version: June 2006. <http://www.w3.org/TR/2006/CR-xquery-20060608/>. – Online-Ressource. – W3C Candidate Recommendation 8 June 2006
- [183] YU, Tao ; LIN, Kwei-Jay: The Design of QoS Broker Algorithms for QoS-Capable Web Services. In: *International Journal of Web Services Research* 1 (2004), Oct-Dec, Nr. 4
- [184] YU, Tao ; LIN, Kwei-Jay: Service Selection Algorithms for composing Complex Services with multiple QoS Constraints. In: BENATALLAH, Boualem (Hrsg.) ; CASATI, Fabio (Hrsg.) ; TRAVERSO, Paulo (Hrsg.): *International Conference on Service Oriented Computing (ICSOC)*, Springer-Verlag, 2005 (LNCS 3826), S. 130–143
- [185] YU, Tao ; LIN, Kwei-Jay: Service Selection Algorithms for Web Services with end-to-end QoS Constraints. In: *Journal for Information Systems and E-Business Management ISeB* 3 (2005), July, Nr. 2, S. 103–126



- [186] ZENG, Liangzhao ; BENATALLAH, Boualem ; DUAMS, Marlon ; KALAGNANAM, Jayant ; SHENG, Quan Z.: Quality Driven Web Service Composition. In: *The 12th International World Wide Web Conference, Budapest, Hungary*, 2003
- [187] ZHOU, Chen ; CHIA, Liang-Tien ; SILVERAJAN, Bilhanan ; LEE, Bu-Sung: UX- An Architecture Providing QoS-Aware and Federated Support for UDDI. In: ZHANG, Liang-Jie (Hrsg.): *ICWS*, CSREA Press, 2003, S. 171–176
- [188] ZUR MUEHLEN, Michael: *Workflow-based Process Controlling. Foundation, Design and Application of workflow-driven Process Information Systems*. Berlin : Logos, 2004