Institut für Informatik
der Technischen Universität München

Dienstorientierte Architekturen: Eine konzeptuelle Herleitung auf Basis eines formalen Prozessmodells

Philipp J. Torka

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. Johann Schlichter
Prüfer der Dissertation:	
1.	Univ.-Prof. Dr. Dr. h.c. Manfred Broy
2.	Univ.-Prof. Dr. Florian Matthes

Die Dissertation wurde am 22. Mai 2013 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 13. September 2013 angenommen.

Zusammenfassung

Die Einführung dienstorientierter Anwendungsarchitekturen (SOA) zielt häufig auf eine verbesserte Ausrichtung von IT-Anwendungen an der Geschäftsarchitektur ab. Dies ist insbesondere von Vorteil in dynamischen Umgebungen, die eine schnelle Anpassbarkeit von IT-Lösungen an neue geschäftliche Rahmenbedingungen und Abläufe erfordern. Dienste werden hier als modulare funktionale Einheiten mit einer einheitlich beschriebenen Schnittstelle angesehen, die eine Entkopplung und einen flexiblen Einsatz von klar abgegrenzten Funktionalitäten ermöglichen.

Die Herausforderung bei der Umsetzung dienstorientierter Architekturen besteht darin, dass Dienste sowohl auf einer fachlichen als auch auf einer technischen Ebene deckungsgleich realisiert werden müssen, damit eine enge Kopplung erfolgen kann. Dazu ist ein präzises Verständnis von Diensten sowohl in der fachlichen als auch in der technischen Domäne erforderlich. Bestehende Dienstdefinitionen, zum Beispiel Web Services, sind jedoch überwiegend technisch motiviert. Die fachliche Dienstbeschreibung erfolgt dagegen meist informell. Dies kann zu Inkonsistenzen bei der Abbildung führen und damit die Realisierbarkeit der versprochenen Flexibilitätsvorteile gefährden.

Im Rahmen dieser Arbeit führen wir eine präzise Begriffsbildung zur fachlichen Beschreibung geschäftlicher Handlungen in dienstorientierten Architekturen durch. Dazu untersuchen wir Aktionen, die durch Personen oder Rechnersysteme als Akteure durchgeführt werden. Wir erfassen geschäftliche Handlungen als Zusammenstellung von Ereignissen, wobei jedes Ereignis die Ausführung einer Aktion darstellt, und bezeichnen diese als Prozesse. Unser fachliches Begriffsmodell basiert auf einer umfassenden syntaktischen und semantischen Definition solcher Prozesse. Zudem berücksichtigt es ebenfalls die Möglichkeit mehrfacher Ausführungen von Handlungen durch unterschiedliche Akteure, indem die unterschiedlichen Kontexte durch den Begriff der Transaktion voneinander abgegrenzt werden.

Wir betrachten den Prozess als Grundkonzept zur Beschreibung geschäftlicher Handlungen und leiten alle weiteren Konzepte einer dienstorientierten Architektur aus dem Prozessbegriff ab. Dabei begreifen wir Dienste als diejenigen Teile eines Prozesses, in denen Interaktionen zwischen Akteuren stattfinden. Eine an Diensten ausgerichtete Architektur strukturiert demnach geschäftliche Abläufe in Unternehmen anhand der Beiträge einzelner Akteure zu den Geschäftsprozessen. Wir realisieren auf diese Weise die in SOA geforderte modulare Beschreibbarkeit und Zusammensetzbarkeit von Funktionalität auf fachlicher Ebene.

Zur intuitiven Herleitung der Begriffe diskutieren wir diese zunächst im Rahmen eines Praxisbeispiels und leiten daraus sukzessive die für die Modellierung von Dienstarchitekturen relevanten Konzepte ab. Im Folgenden formalisieren wir die Begriffe, um ein präzises Verständnis der Konzepte mit ihren Eigenschaften herzustellen sowie eine gegenseitige Einordnung vorzunehmen. Im letzten Teil der Arbeit ordnen wir unser Modell in bestehende Beschreibungsansätze für

geschäftliche Handlungen ein und diskutieren eine informelle Variante des Modells, um dessen einfache praktische Anwendbarkeit zu gewährleisten.

Danksagung

Ich möchte diese Gelegenheit nutzen, um allen zu danken, die mich in meiner Promotion über die letzten gut drei Jahre hinweg begleitet und so auch wesentlich zum Gelingen dieser Arbeit beigetragen haben.

Mein herzlicher Dank gilt in besonderem Maße dem Betreuer meiner Dissertation, Prof. Broy. Ich blicke zurück auf sehr gute und überaus bereichernde Diskussionen, in denen er sich viel Zeit für mich genommen hat und mich maßgeblich in der thematischen Ausrichtung sowie der inhaltlichen Gestaltung meiner Dissertation unterstützt und geleitet hat. Zugleich hat sein gezieltes und immer konstruktives Feedback entscheidend dazu beigetragen, dass ich mein Forschungsgebiet so präzise und differenziert betrachten konnte. Ich möchte auch herzlich meinem Zweitbetreuer, Prof. Matthes, danken. Er hat eine sehr wichtige zweite Perspektive auf meine Arbeit beigesteuert, die es mir ermöglichte, meine Argumentation aus einem anderen Blickwinkel zu hinterfragen und auf diese Weise abzurunden.

Weiterhin bin ich dankbar, dass ich am Lehrstuhl für Software & Systems Engineering vom ersten Tag an ein sehr freundliches und offenes Umfeld vorgefunden habe, das mir eine wichtige Stütze in der Anfertigung dieser Arbeit war. Ich denke zurück an viele spannende inhaltliche Diskussionen genauso wie schöne Erlebnisse jenseits der Arbeit. Mein besonderer Dank gilt hier Wolfgang, Sascha, Martin und Daniel sowie dem erstklassigen administrativen Team am Lehrstuhl, aber auch allen anderen Mitarbeitern und Doktoranden, mit denen ich zusammenarbeiten durfte.

Zuletzt bin ich sehr froh über all die Unterstützung, die ich durch meine Familie, Freunde und Kollegen erhalten habe. Allen voran danke ich herzlich meinen Eltern. Sie waren mir insbesondere auch in den anstrengenden Zeiten, die zu jeder Promotion dazugehören, eine immense Stütze und haben mir durch ihre motivierenden Worte viel Kraft und Energie mit auf den Weg gegeben, um diese Arbeit fertigzustellen. Mein Dank gilt auch allen Freunden und Kollegen, die mich stets begleitet und dazu beigetragen haben, dass die letzten Jahre auch jenseits der Promotion für mich ein spannender Lebensabschnitt waren, den ich nicht missen möchte.

Inhaltsverzeichnis

1 Einleitung

Die rechnergestützte Informationsverarbeitung in Unternehmen und Organisationen ist in heutigen Zeiten nicht mehr wegzudenken. Waren in den Anfängen des IT-Zeitalters nur punktuell einzelne Aufgaben automatisiert, erfolgen heutzutage große Teile der Wertschöpfung mit substantieller Unterstützung durch Informationstechnologie. Dies resultiert in zumeist historisch gewachsenen Anwendungslandschaften, die die komplexen geschäftlichen Strukturen widerspiegeln und gekennzeichnet sind durch eine Vielzahl von Abhängigkeiten. Zugleich sind solche Anwendungslandschaften selten statisch, sondern unterliegen einem ständigen Wandel bedingt durch veränderte geschäftlichen Rahmenbedingungen und Anforderungen. Die damit einhergehenden Herausforderungen sind mit zunehmender Komplexität nur noch auf Basis einer systematischen Strukturplanung von Unternehmensanwendungen zu bewältigen. [Zac96]

Strukturen von Anwendungslandschaften können in Architekturen erfasst werden. Der Begriff Architektur ist prinzipiell generisch und kann auf jede Form von Artefakt angewendet werden. So haben sich Architekturen als Planungsgrundlage analog zum wohlbekannten Einsatz im Bauwesen auch in der Konzeption von Anwendungslandschaften etabliert. Anwendungsarchitekturen beschreiben dabei die Anordnung einzelner Bestandteile in komplexen Systemen sowie die Beziehungen, die zwischen diesen Bestandteilen bestehen. [HN01] Wie im Bauwesen kann auch bei der Gestaltung von Anwendungslandschaften Kartographie eingesetzt werden, um die durch die Architektur vorgegebene Struktur zu visualisieren. [Mat08]

Im unternehmerischen Kontext können Architekturen auf unterschiedlichen Ebenen betrachtet werden. In betriebswirtschaftlicher Hinsicht finden in Unternehmen eine Vielzahl von Handlungen statt, die teils eng miteinander vernetzt sind. Die strukturierte Beschreibung solcher Abläufe bezeichnen wir auch als Prozesse. Geschäftsarchitekturen definieren Prozesse zusammen mit den an ihnen beteiligten Akteuren. Ferner beschreiben sie die Interaktionen und Abhängigkeiten zwischen den Abläufen. Im Bereich der betrieblichen Informationssysteme wird mit Anwendungs- oder auch Softwarearchitekturen die strukturierte Anordnung von Softwarekomponenten und die Interaktionen zwischen den Komponenten dargestellt. [Bal98] Da Softwaresysteme geschäftliche Abläufe unterstützen, besteht naturgemäß ein enger Zusammenhang zwischen Geschäfts- und Anwendungsarchitekturen.

Die Abhängigkeit zwischen diese beiden architekturellen Ausprägungen motiviert, ein einheitliches Architekturmodell, das die wirtschaftliche und technische Sicht integriert, für Unternehmen anzustreben. Dieses wird auch als Unternehmensarchitektur (Enterprise Architecture) bezeichnet. [Ope09] Unternehmensarchitekturen ermöglichen eine ganzheitliche Betrachtung geschäftlicher Handlungsabläufe zusammen mit ihrer softwareseitigen Unterstützung. Sie erfassen die Gesamtheit der Prozesse und Softwareanwendungen eines Unternehmens und stellen diese in einer Ebenenstruktur dar. Eine Unternehmensarchitektur schreibt jedoch nicht zwingend vor, dass

die geschäftliche und technische Ebene den gleichen Prinzipien folgend strukturiert sind. Häufig sind auf technischer Ebene historisch entstandene übergreifende Anwendungen zu beobachten, die sich nur bedingt mit den sich dynamisch weiterentwickelnden Prozessen auf geschäftlicher Ebene in Deckung bringen lassen.

Die Grundidee der Dienstorientierung (Service Orientation) besteht darin, die in Unternehmensarchitekturen beschriebenen Prozesse und Softwareanwendungen modular aus kleinen standardisierten Bausteinen, den sogenannten Diensten, zusammenzusetzen. Auf diese Weise lässt sich ein einfacher Bezug zwischen Geschäfts- und Anwendungsarchitektur herstellen, da nun nicht mehr komplexe Prozesse auf ganze Anwendungen abgebildet werden müssen, sondern die Beziehungen zwischen den jeweiligen Bausteinen betrachtet werden können.

Der Einsatz von dienstorientierten Architekturen (Service Oriented Architectures, kurz SOA) verspricht eine höhere Agilität in dynamischen Geschäftsumfeldern, da geschäftliche Änderungen dank des Bausteinprinzips einfacher auf der Anwendungsebene nachgebildet werden können. Zudem ermöglicht der Einsatz standardisierter Bausteine auch eine Wiederverwendung und eine verteilte Realisierung von Architekturbestandteilen. Die Innovation von SOA liegt dabei nicht in der Modularisierung von Komponenten, die in der Informationstechnologie längst ein anerkanntes Muster darstellt. Sie drückt sich vielmehr in der einheitlichen Anwendung der Modularisierung in der fachlichen und technischen Architektur aus, die eine bessere Abstimmung von Fachlichkeit und Implementierung ermöglicht und damit die konsistente Durchführung von Änderungen über die Ebenen hinweg vereinfacht.

Dienstorientierte Architekturen haben aus diesem Grund in den letzten Jahren in Wissenschaft und Praxis viel Beachtung erfahren. Nicht zuletzt auf Grund der hohen Praxisrelevanz hat SOA nach ersten Anwendungen in der Telekommunikationsbranche schnell weitere Verbreitung gefunden und ist inzwischen in einer großen Zahl von Unternehmen und Organisationen als Gestaltungsrichtlinie etabliert. [Smi09, Zac05] Jedoch existiert nach wie vor keine einheitliche Definition von SOA und den dazugehörigen Diensten. Auf technischer Ebene werden Dienstarchitekturen in der Regel auf Basis von Web Services realisiert. Entsprechend sind aktuelle SOA-Definitionen stark durch die Eigenschaften von Web Services geprägt (vgl. [NL05, PVDH07]). Die präzise fachliche Einordnung ist dabei nicht im Fokus, was im Widerspruch zum Anspruch von SOA steht, Dienste konsistent auf allen Ebenen zu verankern.

Wir werden uns im Folgenden mit der Begriffsbildung und Modellierung im fachlichen Kontext dienstorientierter Architekturen befassen. Damit ermöglichen wir eine präzise Darstellung von Diensten und Einordnung mit anderen Konzepten, die zur Beschreibung einer Geschäftsarchitektur relevant sind.

1.1 Motivation

Dienstorientierte Architekturen haben in den vergangenen Jahren breite Beachtung erfahren und werden dabei überwiegend als vielversprechendes und zukunftsweisendes Gestaltungsmuster für moderne Unternehmensarchitekturen angesehen (vgl. [Zac05]). Das große Interesse an SOA ist dabei primär auf wirtschaftliche Erwägungen zurückzuführen. Mit der Einführung einer

Dienstorientierung in Anwendungsarchitekturen werden zumeist klare Erwartungen hinsichtlich einer Steigerung der Effektivität und Effizienz der IT-Funktion verbunden. [LLJ+05]

In der Tat lassen sich die Gestaltungsprinzipien von SOA mit wirtschaftlichen Vorteilen assoziieren. So kann durch die konsistent auf fachlicher und technischer Ebene eingeführte Modularisierung die Ausrichtung der IT an der Fachlichkeit verbessert und damit eine höhere Agilität erreicht werden. Genauso vereinfacht eine feingranulare Kapselung von Funktionalität in standardisierten Bausteinen, den Diensten, deren Wiederverwendung. Eine mehrfache Nutzung von Anwendungsteilen steigert die Entwicklungs- und Betriebseffizienz und kann sich durch die damit einhergehende Vereinheitlichung auch positiv auf die Qualität der Anwendungsarchitektur auswirken. [MVAR07]

Eine Reihe von Fallstudien aus unterschiedlichen Industrien, die im Laufe der letzten Jahre durchgeführt wurden, belegen die grundsätzliche Realisierbarkeit der versprochenen Vorteile beim Einsatz von dienstorientierten Architekturen. [HP04, BCHM+05, KDK07] Die Fallstudien heben dabei nicht nur Effizienzgewinne, eine höhere Agilität und eine gesteigerte Architekturqualität innerhalb der betrachteten Unternehmen hervor, sondern verweisen auch auf Vorteile in der Zusammenarbeit zwischen Unternehmen durch eine Ausrichtung der Kommunikation an standardisierten und modularen Diensteinheiten.

Jedoch hängt die Umsetzbarkeit der soeben beschriebenen wirtschaftlichen Vorteile stark von einer präzisen Realisierung von SOA und den damit verbundenen Gestaltungsrichtlinien in der Praxis ab. So kann beispielsweise eine schnelle Umsetzung neuer geschäftlicher Anforderungen nur dann erfolgen, wenn sowohl auf fachlicher als auch technischer Ebene Dienste exakt und konsistent beschrieben sind, so dass eine eindeutige Abbildung zwischen den Ebenen erfolgen kann. Sobald Ambiguitäten in der Dienstbeschreibung entstehen, kann bei Änderungen an Geschäftsabläufen nicht mehr gewährleistet werden, dass diese adäquat auf der Implementierungsebene nachgebildet werden können. Ebenso setzt eine Wiederverwendung von Anwendungsteilen ein klares Verständnis bezüglich deren Funktionalität voraus.

Dienstorientierte Architekturen werden auf technischer Ebene häufig gleichgesetzt mit dem Implementierungskonstrukt Web Services, obwohl dieses genau genommen nur eine mögliche Ausprägung von Diensten auf technischer Ebene darstellt. Die weite Verbreitung von Web Services hat jedoch dazugeführt, dass diese als de facto Standard für die Implementierung von SOA angesehen werden. [Erl08] Die Definition von Web Services fokussiert auf eine syntaktische Schnittstellenbeschreibung der Dienste. Das Verhalten wird dagegen nur in textueller Form beziehungsweise über eine Sammlung von Schlüsselwörtern informell erfasst. [Mel08] Insofern ergibt sich die genaue Semantik der Dienstschnittstelle erst nach erfolgter Bindung der Web Services an eine Implementierung der Schnittstelle.

Auf geschäftlicher Ebene hat sich anders als im Bereich der Implementierung bislang kein Standard für die Dienstbeschreibung durchgesetzt. Folglich orientiert sich die fachliche Definition von Diensten häufig ebenso an Web Services. Entsprechend wird auch auf der Fachebene das Verhalten zumeist nur informell erfasst, so dass sich Unschärfen in der Auslegung der Funktionalität nicht vermeiden lassen. Daraus resultiert ein Interpretationsspielraum in der Abbildung von Diensten auf geschäftlicher Ebene auf Web Services. Zusätzlich ergibt sich die Problematik, dass auf geschäftlicher Ebene auf Grund einer fehlenden etablierten Dienstdefinition keine

klare Einordnung des Konzepts Dienst mit anderen geschäftlichen Konzepten, insbesondere den Prozessen, möglich ist.

Die fehlende Präzision in der Beschreibung und Einordnung von Diensten ist einer der meistgenannten Probleme im Umgang mit SOA. [VM07] Die Unschärfen in der Definition machen dienstorientierte Architekturen fehleranfällig und gefährden damit die Realisierbarkeit des SOA Wertversprechens. [BWM07] Die beschriebenen SOA-Vorteile ergeben sich insbesondere aus einer starken Orientierung an der Fachlichkeit. Insofern ist ein präzises Verständnis von SOA auf der Fachebene für eine erfolgreiche Implementierung von großer Bedeutung. Mit der konzeptuellen Herleitung von Diensten auf Basis eines formalen Prozessmodells sowie der Einordnung mit verwandten Konzepten im geschäftlichen Umfeld adressieren wir diese Herausforderung und ermöglichen eine exakte Beschreibung von dienstorientierten Architekturen jenseits der Implementierungsebene.

1.2 Zielsetzung und Beitrag

Das übergreifende Ziel dieser Arbeit besteht darin, auf fachlicher Ebene ein präzises Konzeptverständnis in Dienstarchitekturen herzustellen. Im Mittelpunkt steht dabei der Aufbau eines formalen Begriffsmodells, das ausgehend von einer formalen Prozesssicht die für SOA zentralen Konzepte umfasst und Beziehungen zwischen diesen darstellt. Das Modell wird auf Basis einer Praxissituation entwickelt und fortlaufend mit Beispielen illustriert, um die Anwendbarkeit des Modells zu gewährleisten. Nach Einführung der Formalismen diskutieren wir Anwendungsaspekte des Modells.

Modelle stellen Abbilder der Realität dar. Die Vielzahl der unterschiedlichen Begriffsmodelle, die im Bereich SOA existiert, belegt eindrücklich, dass die Abbildung in Modellen nicht eindeutig ist, sondern immer auch durch die Wahrnehmung des Modellierenden geprägt wird. Unser Begriffsmodell repräsentiert insofern zweifellos nicht die einzig mögliche Sicht auf SOA. Wir sehen im Modell nicht das Ziel der Arbeit, sondern vielmehr den Ausgangspunkt, um präzise und konsistente Aussagen über Dienstarchitekturen auf fachlicher Ebene treffen zu können. Die gewählten Formalismen stellen dabei einen möglichen Ansatz dar, diese Aussagen exakt abzubilden. Bei der Entwicklung des Modell verfolgen wir den Anspruch, einen fachlichen Dienstbegriff herauszuarbeiten, der zum intuitiven Verständnis von Dienstleistungen in der Praxis passt und sich mit dem etablierten technischen Verständnis von Diensten – in erster Linie Web Services – vereinen lässt.

Im Detail betrachtet umfasst der wissenschaftliche Beitrag dieser Arbeit die folgenden Aspekte:

- *Erarbeitung eines systematischen Überblicks zu bestehenden SOA-Definitionen mit deren Zielsetzungen und Prinzipien*
  Der erste Beitrag der Arbeit besteht in der Erarbeitung eines systematischen Überblicks zum aktuellen Verständnis von dienstorientierten Architekturen. Dazu analysieren wir die Auslegung des Dienstbegriffs, strukturieren die unterschiedlichen Aspekte von Dienstdefinitionen und untersuchen, wie sich Dienste in Dienstarchitekturen einordnen. Zudem

diskutieren wir die Zielsetzungen und Prinzipien, die den gegenwärtigen SOA Initiativen zugrunde liegen.

- *Aufbau eines formalen Modells zur Darstellung geschäftlicher Abläufe auf fachlicher Ebene*
  Der zweite Beitrag liegt im Aufbau eines formalen Modells zur präzisen Darstellung geschäftlicher Abläufe, die wir in ihrer Grundform als Prozess bezeichnen. Das Modell dient uns als Grundlage und Referenzpunkt für alle zu treffenden konzeptuellen Aussagen. Dazu befassen wir uns ausführlich mit der syntaktischen und semantischen Erfassung von Prozessen und lassen unterschiedliche Zeitmodelle zur Ablaufbeschreibung zu. Wir führen das Modell in mehreren Stufen ein und bauen es in einer modularen Weise auf, um eine möglichst flexible und trotzdem präzise konzeptuelle Diskussion zu ermöglichen.
- *Ableitung einer präzisen fachlichen Sicht auf Dienste und dienstorientierte Architekturen aus dem Prozessmodell*
  Im dritten Beitrag leiten wir aus dem eingeführten Prozessmodell eine Dienstsicht ab. Wir fassen Dienste als eine besondere Ausprägung der Prozessbeschreibung auf, die sich an Akteuren ausrichtet. Die formale Ableitung aus dem Prozessmodell ermöglicht dabei eine Gegenüberstellung der Konzepte Prozess und Dienst sowie eine präzise Abgrenzung mit anderen Konzepten, die in Geschäftsarchitekturen eingesetzt werden. Zudem befassen wir uns mit dem architekturellen Aufbau von Prozessen, fokussieren dabei auf Prozessstrukturen, die an Diensten ausgerichtet sind, und ermöglichen somit die Beschreibung dienstorientierter Architekturen.
- *Verknüpfung des formalen Modells mit dem intuitiven Begriffsverständnis*
  Der vierte Beitrag besteht in der konsequenten Verankerung des formalen Modells in der Praxis. Vor Aufbau des Modells führen wir eine Begriffsbildung anhand eines Anwendungsbeispiels durch. Im Zuge der Diskussion des Beispiels extrahieren wir sukzessive die Konzepte, die zur Beschreibung einer dienstorientierten Geschäftsarchitektur benötigt werden, so dass jedes formale Konzept unmittelbar zu einer praktischen Anwendung in Bezug gesetzt werden kann. Zudem halten wir auch während der formalen Modellierung den Praxisbezug aufrecht, indem wir die Anwendbarkeit des Modells anhand des eingeführten Beispiels demonstrieren.
- *Diskussion von informellen Beschreibungsansätzen*
  Der fünfte Beitrag der Arbeit liegt in der Diskussion von vereinfachten Darstellungsoptionen des Modells. Wenngleich das Modell eine hohe Präzision der SOA-Beschreibung ermöglicht, ist dessen Anwendung auch mit hohen Aufwänden verbunden. Daher leiten wir basierend auf den Erkenntnissen des formalen Modells informelle Beschreibungsansätze für Dienstarchitekturen ab. Insbesondere im Fokus stehen dabei graphische Modelle.

1.3 Abgrenzung

Der Aufbau von dienstorientierten Architekturen kann sowohl konzeptuell als auch methodisch beschrieben werden. Dabei stellt ein präzises konzeptuelles Verständnis die Grundlage für eine

aussagekräftige und differenzierte methodische Untersuchung dar. Aus diesem Grund fokussieren wir uns in dieser Arbeit auf eine konzeptuelle Betrachtung dienstorientierter Architekturen als ersten essentiellen Schritt zu einer Fundierung des SOA-Begriffs. Wir sehen die methodische Diskussion von SOA als zweiten Forschungshorizont an, der unmittelbar im Anschluss an diese Arbeit erfolgen kann. Als Ausblick darauf werden wir zum Abschluss unserer Modellierung einige zentrale methodische Fragestellungen skizzieren und darstellen, wie unser konzeptuelles Modell zur Beantwortung dieser beitragen kann.

Die Erstellung eines formalen konzeptuellen Modells erfordert eine Abwägung zwischen Breite und Tiefe der Darstellung. Wir streben in dieser Arbeit eine möglichst präzise Beschreibung und Einordnung der zentralen fachlichen Konzepte in dienstorientierten Architekturen an. Um diese Konzepte umfassend und in Tiefe diskutieren zu können, nehmen wir einige Aspekte aus der Betrachtung aus, die wir nicht als essentielle Bestandteile zum Aufbau des Begriffsverständnisses ansehen.

Wir konzentrieren uns in unserer Betrachtung von dienstorientierten Architekturen auf die Modellierung von geschäftlichen Handlungen und deren Unterstützung durch Softwareanwendungen. Dienste in diesem Kontext sind funktionale modulare Einheiten, die Teilaspekte dieser Handlungen beschreiben. Der Dienstbegriff wird häufig auch auf die IT-Infrastruktur eines Unternehmens ausgeweitet. In diesem Fall sind Dienste als klar definierte und technisch standardisierte Zuweisung von Ressourcen eines Rechenzentrums anzusehen. So wird beispielsweise eine Bereitstellung einer bestimmten Serverkapazität in einer gegebenen Serverarchitektur oder einer bestimmten Menge Speicherplatz ebenso als Dienst eines Rechenzentrums bezeichnet. [Ope09, KBS07] Dienste dieser Art werden auch als Infrastrukturdienste bezeichnet. Wir fokussieren uns jedoch auf die Erfassung von Abläufen und nehmen Infrastrukturdienste aus unserer Betrachtung aus.

In der Modellierung geschäftlicher Handlungen fokussieren wir uns zudem auf funktionale Aspekte, da wir primär das Verhalten erfassen und untersuchen möchten. Zur vollständigen Beschreibung einer Dienstarchitektur gehören auch nicht funktionale Überlegungen wie Aussagen zur Verfügbarkeit, Sicherheit und Skalierbarkeit der beschriebenen Konzepte. Diese Punkte werden auch unter dem Stichwort des SOA- beziehungsweise Dienstmanagements zusammengefasst. [PVDH07] Jedoch betrachten wir diese Aspekte als Attribute der jeweiligen Konzepte, die keinen Einfluss auf die Einordnung mit anderen Bestandteile der Architektur haben. Insofern kann eine nicht funktionale Sicht auf die Architektur zu einem späteren Zeitpunkt problemlos ergänzt werden.

1.4 Aufbau der Arbeit

Im weiteren Verlauf der Arbeit streben wir eine enge Verknüpfung des intuitiven praxisnahen Begriffsverständnisses von SOA mit einer präzisen formalen Sicht an. Dazu erarbeiten wir auf Basis einer informellen, an einem Beispiel ausgerichteten Begriffsbildung der im fachlichen Kontext von SOA relevanten Konzepte ein formales Modell. Wir gehen dabei insbesondere auf den Begriff des Prozesses als Basiskonzept für alle weiteren Elemente einer dienstorientierten

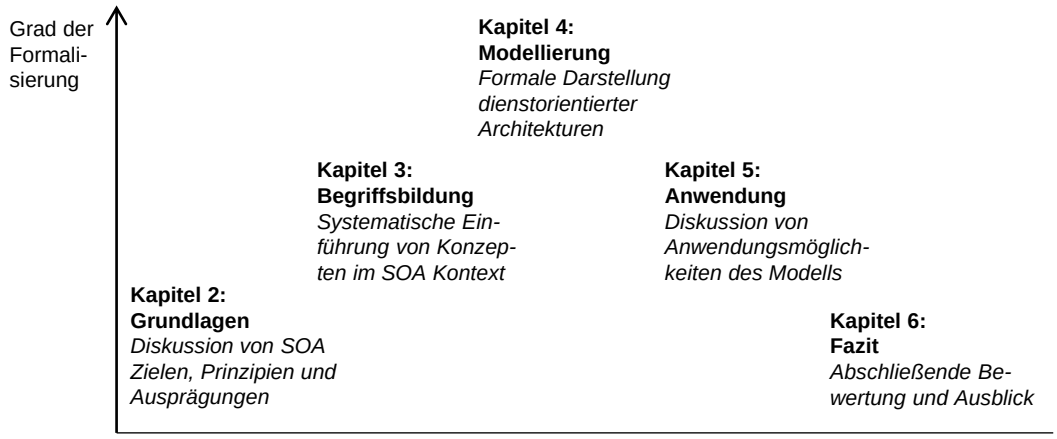

Abbildung 1.1: Struktur der Arbeit

Architektur ein. Im letzten Teil der Arbeit befassen wir uns mit der praktischen Anwendung des Modells, indem wir dessen Einordnung mit anderen Beschreibungsansätzen diskutieren und vereinfachte Darstellungsformen einführen. Abbildung 1.1 zeigt im Überblick die Zielsetzungen der folgenden Kapitel.

Im zweiten Kapitel setzen wir uns mit dem bestehenden Verständnis von Architekturen und Dienstorientierung auseinander. Wir untersuchen dazu geläufige Definitionen und leiten aus diesen die wesentliche Bestandteile und Eigenschaften von Architekturen sowie auch die zugrundeliegende Motivation ab. Wir schließen das Kapitel ab mit einem Überblick zu offenen Fragen und daraus resultierenden Anforderungen, die wir in den Folgekapiteln adressieren.

Im dritten Kapitel nehmen wir eine Begriffsbildung auf fachlich architektureller Ebene vor. Dazu diskutieren wir ein Praxisbeispiel und extrahieren aus diesem sukzessive die für SOA relevanten Konzepte. Der Fokus in diesem Kapitel liegt auf einer intuitiven Begriffsbildung. Durch die konsequente Diskussion anhand des Beispiels streben wir an, unser Konzeptverständnis direkt an der praktischen Anwendung auszurichten.

Im vierten Kapitel formalisieren wir die eingeführten Konzepte und ordnen diese in ein konsistentes mathematisches Modell ein. Ziel des Modells ist eine einheitliche und präzise Beschreibung von Prozessen und daraus abgeleiteten Konzepten, wie zum Beispiel Diensten, sowie deren Beziehungen in einer formalen Notation. Wir diskutieren das Modell ebenso anhand des Praxisbeispiels, um eine Anwendbarkeit der eingeführten Formalismen sicherzustellen.

Im fünften Kapitel befassen wir uns mit der Anwendung des Modells. Das Modell ermöglicht auf Grund der durchgängig formalen Notation zwar eine präzise Darstellung von SOA, ist aber auch mit hohen Modellierungsaufwänden verbunden. Wir fassen daher die aus dem Modell gewonnenen Erkenntnisse in Prinzipien für die Beschreibung von dienstorientierten Architekturen zusammen und besprechen, wie diese Prinzipien auch in einfacheren, beispielsweise graphischen,

Modellen umgesetzt werden können. Zudem nehmen wir eine Einordnung mit vergleichbaren Beschreibungsansätzen vor.

Im sechsten Kapitel fassen wir schließlich die Ergebnisse der Arbeit zusammen und geben einen Ausblick auf mögliche Modellerweiterungen sowie Anknüpfungspunkte für weitergehende Forschung.

2 Grundlagen

In diesem Kapitel behandeln wir die Grundlagen für die Begriffsbildung und Modellierung im Kontext dienstorientierter Architekturen. Dazu geben wir zunächst einen Überblick zu den Prinzipien und der Zielsetzung von Unternehmensarchitekturen im Allgemeinen sowie von Dienstorientierung innerhalb solcher Architekturen. Im Zuge dessen werden wir einen Überblick zu den Begriffen erhalten, die typischerweise zur Beschreibung von SOA benötigt werden.

Wir werden im ersten Abschnitt des Kapitels Unternehmensarchitekturen im Allgemeinen als Ansatz zur Strukturierung betrieblicher Einrichtungen und Abläufe motivieren. Zudem betrachten wir etablierte Referenzmodelle, die derzeit zur Beschreibung von Unternehmenslandschaften eingesetzt werden. Dabei legen wir besonderes Augenmerk auf die Darstellung geschäftlicher Handlungen und Abläufe, da wir Dienste im Folgenden als Konzepte zur Ablaufbeschreibung in Architekturen integrieren werden.

Im zweiten Teil des Kapitels setzten wir uns mit dem aktuellen Verständnis von dienstorientierten Architekturen auseinander. Dazu leiten wir zuerst die Motivation, den Anspruch und die Zielsetzung her, die mit der Einführung von Dienstorientierung in Unternehmensarchitekturen verfolgt werden. Anders als bei den Architekturen im Allgemeinen hat sich noch keine einheitliche Definition von SOA herausgebildet. Wir untersuchen daher die Eigenschaften und Beziehungen, die die einzelnen derzeit geläufigen Definitionen mit SOA assoziieren, und diskutieren, inwiefern sich diese mit dem SOA-Anspruch in Deckung bringen lassen. Wir werden dabei feststellen, dass sich auf technischer Ebene dank des etablierten Konzeptes der Web Services bereits ein konsistentes Bild ergibt, auf fachlicher Ebene jedoch Dienste nur unscharf erfasst werden können. Insbesondere ist auch eine präzise Einordnung mit anderen fachlichen Konzepten, die zur Beschreibung von Unternehmensarchitekturen herangezogen werden, nicht eindeutig durchführbar.

Wir werden daher zum Abschluss des Kapitels die identifizierten offenen Fragen zur Beschreibung von SOA auf fachlicher Ebene zusammenstellen. Diese Fragen bilden die Grundlage für die Begriffsbildung und Modellierung in den folgenden Kapiteln.

2.1 Unternehmensarchitekturen

Wir führen den Begriff der Unternehmensarchitektur (Enterprise Architecture) zunächst unabhängig vom Dienstbegriff ein. Der Architekturbegriff in Unternehmen leitet sich direkt aus dem traditionellen Architekturbegriff ab. Eine Architektur beschreibt die Elemente eines Systems sowie deren Zusammensetzung und die Beziehungen, die zwischen den Elementen bestehen.

[IEE11, HN01, Ope09] Im Kontext eines Unternehmens sind unter solchen Elementen in erster Linie die Subjekte, Objekte des Unternehmens sowie die Handlungen, die die Subjekte mit den Objekten durchführen, zu verstehen. Als Subjekte betrachten wir Mitarbeiter, Gruppen von Mitarbeitern, beispielsweise Organisationseinheiten und auch Rechnersysteme, die als automatisierte Akteure betrachtet werden können. Unter Objekten verstehen wir in voller Allgemeinheit jede Form von materiellen oder immateriellen Sachgegenständen, die von den Subjekten eines Unternehmens benutzt werden.

Wir werden uns im Folgenden primär mit der Informationsverarbeitung in Unternehmen auseinandersetzen. Dazu betrachten wir in erster Linie Architekturen, die den Austausch und die Bearbeitung von Informationen als immaterielle Objekte durch Mitarbeiter oder Rechnersysteme als Unternehmenssubjekte beschreiben.

Architekturen zur automatisierten Informationsverarbeitung können sowohl aus einer fachlichen als auch aus einer technischen Sicht betrachtet werden. Die fachliche Sicht ist eine Anforderungssicht, die aus einer geschäftlichen Perspektive beschreibt, welche Handlungen beziehungsweise Abläufe von den Personen und Rechensystemen erwartet werden. Dazu werden auch Interaktionen anhand des Informationsaustausches zwischen den Subjekten erfasst. Die Subjekte und Objekte werden dabei nur so detailliert beschrieben, dass die Handlungen und Interaktionen adäquat spezifiziert werden können. Die Handlungsbeschreibung auf fachlicher Ebene erfolgt meist ergebnisorientiert. Die fachliche Sicht wird auch als Auftraggebersicht angesehen, da sie sich mit den Anforderungen an Abläufe, beteiligte Entitäten und Objekte befasst.

Die technische Sicht beschreibt dagegen als Realisierungssicht, wie die einzelnen Handlungen und Interaktionen mit Hilfe von Software umgesetzt werden. Dies setzt eine detaillierte Erfassung der Subjekte und Objekte voraus. Die Beschreibung von Handlungen und Abläufen erfolgt vorgehensorientiert. Die technische Perspektive fokussiert dabei auf die Darstellung einer Implementierung, die zum gewünschten Ergebnis führt. Sie stellt somit eine Auftragnehmersicht dar, da sie sich mit der Umsetzung von Abläufen und der damit verbundenen detaillierten Erfassung von Entitäten und Objekten auseinandersetzt.

Wir bezeichnen die geschäftliche Sicht einer Architektur auch als Geschäftsarchitektur und die technische Sicht als IT-Architektur. Die IT-Architektur stellt im engeren Sinne betrachtet die Anwendungslandschaft eines Unternehmens dar. [EHH+08] Zusätzlich kann sie auch ein Daten- und Rollenmodell auf Implementierungsebene umfassen. Teilweise wird die IT-Architektur auch weiter differenziert in eine Anwendungs- und Datenarchitektur sowie Architektur der Infrastruktur, die sich mit der Konfiguration der Hardware befasst.

Die Unternehmensarchitektur fasst die Geschäftsarchitektur und die IT-Architektur in einer einheitlichen Betrachtung zusammen. [EHH+08, WF07] Im Idealfall leitet sich die technische Architektur direkt aus der geschäftlichen Architektur dahingehend ab, dass in der technischen Sicht die Implementierung der geschäftlichen Anforderungen umgesetzt wird. Die Unternehmensarchitektur wird daher häufig auch in einem Ebenenmodell dargestellt, um den Zusammenhang zwischen den Sichten zu unterstreichen. Jedoch gewährleistet eine Unternehmensarchitektur lediglich eine einheitliche Darstellung der fachlichen und technischen Sicht, erzwingt aber nicht Kongruenz über die Ebenen hinweg. Die Herstellung dieser ist vielmehr als Ergebnis einer engen Abstimmung zwischen Geschäfts- und IT-Architekten zu sehen.

Eine Unternehmensarchitektur muss nicht zwingend auf nur zwei Ebenen betrachtet werden. Der Übergang von einer Anforderungssicht auf Geschäftsebene bis hin zu einer Implementierungsperspektive auf technischer Ebene kann auch in mehreren Abstufungen erfolgen. Die soeben erwähnte Differenzierung der IT-Architektur in eine Anwendungs- und Infrastrukturperspektive ist ein Beispiel dafür.

Orthogonal zur Differenzierung nach Fach- und IT-Ebene kann die Architekturbeschreibung auch auf bestimmte Aspekte fokussieren. So kann die Beschreibung auf Unternehmenssubjekte, also die handelnde Entitäten, die Objekte, beispielsweise die Informationen und deren Struktur, oder auch auf die Darstellung der Abläufe an sich konzentriert werden. Entsprechend sind beliebige Kombinationen aus Ebenen- und Aspektbetrachtung möglich. So können beispielsweise ausschließlich Handlungsabläufe auf geschäftlicher Ebene oder die Datenstruktur auf IT-Ebene untersucht werden. Diese einzelnen Ausprägungen der Architektur werden auch als Architektursichten bezeichnet. Sie adressieren die Interessen der unterschiedlichen Stakeholder an der Architektur. [IEE11]

Eine Unternehmensarchitektur kann sowohl den Soll- als auch den Ist-Zustand eines Unternehmens abbilden. [WF07] Im ersten Fall ist die Architektur als Bauplan zu verstehen, aus dem sich die Prinzipien für das Design und die Weiterentwicklung der Unternehmenssysteme ableiten lassen. [IEE11, Ope09] Die Architektur beschreibt die Bestandteile des Systems, so dass die Produktion sich an den Anforderungen der Architektur ausrichten kann. [Zac96, KBS07]

Zur Darstellung von Architekturen und insbesondere auch der einzelnen Architektursichten verwenden wir Architekturmodelle. Ein Modell stellt ein Abbild der Realität in einer strukturierten Beschreibungsweise dar. Im Rahmen dieser Arbeit werden wir uns primär auf die präzise Modellierung von Handlungsaspekten auf der geschäftlichen Ebene konzentrieren.

Ein Architekturframework stellt schließlich eine Zusammenstellung von Modellen für einzelne Architektursichten zur ganzheitlichen und konsistenten Beschreibung einer Unternehmensarchitektur dar. Ein Framework beinhaltet damit Beschreibungsansätze der Architektur für die unterschiedlichen Stakeholder in einem Unternehmer, die ihren Aufgaben entsprechend verschiedene Sichtweisen und Interessen an der Unternehmensarchitektur vertreten. [IEE11]

2.1.1 Zielsetzung

Die Planung, Beschreibung und Weiterentwicklung von Unternehmensarchitekturen (Enterprise Architecture Management, kurz EAM) zielt auf eine ganzheitliche systematische Betrachtung von Geschäfts- und den damit verbundenen Anwendungslandschaften in Unternehmen ab. Die Motivation für den Einsatz von Unternehmensarchitekturen ist vielschichtig.

Ein wesentlicher Mehrwert liegt in der Ganzheitlichkeit der Darstellung, die zu einer deutlich verbesserten Transparenz führt. Diese ermöglicht es, bei der Entwicklung sowie bei Änderungen und Weiterentwicklungen Abhängigkeiten schnell und vollständig zu erkennen sowie auf bestehende Teile der Architektur zurückzugreifen. Zudem verringert die strukturierte Sichtweise die Komplexität, da die Betrachtung auf relevante Teilbereiche der Architektur fokussiert werden kann. Die Integration von geschäftlichen und technischen Aspekten in einer einheitlichen Beschreibung vereinfacht die Aufklärung von Diskrepanzen zwischen fachlicher Intension

und systemseitiger Realisierung sowie die systematische Ableitung von neuer Funktionalität aus der Fachlichkeit.

Der erwartete Nutzen der beschriebenen Aspekte liegt insbesondere in Qualitätssteigerungen und Effizienzgewinnen. Die systematische Beschreibung von geschäftlichen und technischen Aspekten eines Unternehmens über die verschiedenen Architektursichten erhöht die Konsistenz, vereinfacht Anpassungen und führt damit zu einer höheren Agilität. Zudem unterstützen ganzheitliche Architekturmodelle arbeitsteilige Abläufe, da sie Beziehungen und die zu erwartenden Interaktionen zwischen den beteiligten Personen und Rechnersystemen präzise beschreiben. [WBMS10, Zac96, Ope09]

2.1.2 Referenzmodelle

Die Abbildung von Unternehmensstrukturen in Architekturmodellen im Kontext der betrieblichen Informationsverarbeitung hat sich inzwischen in einer Vielzahl von Architekturframeworks manifestiert. Wir werden im Folgenden exemplarisch drei Ausprägungen diskutieren, die unterschiedliche Schwerpunkte in der Architekturbeschreibung setzen und auf diese Weise einen groben Überblick über bestehende Ansätze zur Erfassung von Architekturen geben.

Strukturmodellierung John Zachman hat 1987 mit seiner Publikation [Zac87] die Diskussion um Unternehmensarchitekturen angestoßen und eine erste Schematik zur Strukturierung betrieblicher Anwendungslandschaften auf fachlicher und technischer Ebene vorgeschlagen. In den Folgejahren hat sich daraus sukzessive das Zachman Framework für Unternehmensarchitekturen etabliert. Der wesentliche Beitrag des Frameworks liegt in der Systematisierung der unterschiedlichen Sichten auf Unternehmensarchitekturen. Zachman erstellt damit eine Taxonomie zur Einordnung architektureller Beschreibungsansätze. Die Diskussion konkreter Modelle zur Abbildung der einzelnen Sichten sowie Vorgehensaspekte hinsichtlich der Architekturentwicklung stehen dagegen nicht im Fokus.

In seiner heutigen Form [Zac08] ist das Framework zweidimensional aufgebaut. In der Vertikalen sieht das Framework fünf Ebenen vor, die eine graduelle Überführung von einer fachlichen in eine technische Perspektive vornehmen. In der Horizontalen differenziert es zwischen sechs unterschiedlichen Sichten auf eine Unternehmensarchitektur, die deren unterschiedlichen Bestandteile entlang der W-Fragen Wer?, Wie?, Was?, Wann?, Wo? und Wieso? beleuchten.

Die fünf Ebenen reflektieren die verschiedenen Sichtweisen, die die Mitarbeiter eines Unternehmens auf dessen Architektur haben. Die oberste Ebene stellt die Perspektive des Managements dar, gefolgt von der Sichtweise der Fachseite, der eines Softwarearchitekten, der Perspektive eines Softwareentwicklers und schließlich eines technischen Mitarbeiters, der mit dem Betrieb befasst ist. Dabei erfolgt ein gradueller Übergang von einer rein fachlichen und anforderungsgetriebenen Sicht auf die Architektur zu einer strikt technischen und realisierungsorientierten Perspektive. Die oberen Ebenen sind grobgranular beschrieben und beschränken sich auf eine Erfassung der aus einer wirtschaftlichen Sicht zentralen Elemente und deren Zusammenhänge. Die unteren Ebenen dagegen gehen detailliert auf die einzelnen Elemente ein und befassen sich mit der präzisen Beschreibung von Umsetzungsdetails.

In der Horizontalen erfasst das Zachman Framework die unterschiedlichen Bestandteile einer Architektur. Die erste Spalte befasst sich mit den Objekten des Unternehmens. Diese können materieller Natur sein, zum Beispiel Vorräte, oder auch immateriell, beispielsweise Informationen. Auf fachlicher Ebene werden sie als geschäftliche Gegenstände mit ihren Eigenschaften und Beziehungen beschrieben und dann zunehmend verfeinert. Auf technischer Ebene erfolgt eine Übersetzung in ein entsprechendes Datenmodell.

Die zweite Spalte erfasst Abläufe in Unternehmen, indem sie beschreibt, wie die Subjekte mit den Objekten umgehen und diese verändern. Auf geschäftlicher Ebene werden Abläufe in der Regel als Geschäftsprozesse betrachtet und dann sukzessive in Ablaufvorschriften überführt, die als Software implementiert werden können.

Die dritte Spalte setzt sich mit Orten auseinander und intendiert eine systematische Erfassung der Ausführungsorte für die beschriebenen Abläufe und Algorithmen auf fachlicher und technischer Ebene.

Die vierte Spalte fokussiert auf die Subjekte eines Unternehmens. Sie beschreibt auf fachlicher Ebene die handelnden Entitäten mit den dazugehörigen Rollenbeschreibungen und Verantwortlichkeiten. Auf technischer Ebene werden diese auf Rechnersysteme als automatisierte Akteure abgebildet.

Die fünfte Spalte befasst sich mit der zeitlichen Dimension einer Architektur. Sie definiert eine einheitliche Zeitrechnung, auf die die Abläufe abgebildet werden können. Die sechste Spalte beleuchtet schließlich die Motivation, indem sie der Architektur zugrundeliegende Ziele und Prinzipien auf fachlicher und technischer Ebene erfasst.

Vorgehensmodellierung Das Open Group Architecture Framework (TOGAF), das durch die Open Group herausgegeben wird [Ope09], beschreibt eine Methode zur Entwicklung von Unternehmensarchitekturen. Dazu teilt es den Entstehungsvorgang einer Architektur in mehrere Phasen ein und beschreibt für die einzelnen Phasen erforderliche Voraussetzungen sowie zu erwartende Endprodukte. Die Phasen orientieren sich an der aus dem Zachman Framework bereits bekannten Ebenenstruktur, die einen sukzessiven Übergang von der Fachlichkeit zu einer Implementierungsbetrachtung vorsieht. Der Fokus von TOGAF liegt jedoch im Kontrast zu Zachman nicht in der Strukturierung, sondern in der detaillierten Beschreibung des Vorgehens zur Entwicklung von Architekturen. Entlang der einzelnen Phasen definiert das Framework die für den entsprechenden Abschnitt relevanten Architekturelemente.

Die erste Phase beschäftigt sich mit den Rahmenbedingungen für den Aufbau der Unternehmensarchitektur. Sie hält Unternehmensziele und die daraus für die Architektur ableitbaren Gestaltungsprinzipien fest, klärt Verantwortlichkeiten in der Architekturentwicklung und definiert den Anwendungsbereich und Umfang der Architektur. Die zweite Phase zielt auf eine Modellierung der geschäftlichen Architekturebene ab. Dazu beschreibt sie, wie auf Basis der aufgestellten Rahmenbedingungen die handelnden Entitäten, Unternehmensobjekte sowie die geschäftlichen Abläufe in Form von Geschäftsprozessen erfasst werden. Die ersten beiden Phasen haben einen klaren fachlichen Fokus und befassen sich mit der Abbildung von geschäftlich motivierten Zusammenhängen in der Architektur.

Die Folgephasen nehmen dagegen eine stärker technische Perspektive ein. Die dritte Phase ist zweigeteilt und setzt sich mit der Definition einer Daten- und Anwendungsarchitektur auseinander. Im Rahmen der Datenmodellierung werden die in den Geschäftsprozessen referenzierten Objekte in ein strukturiertes Datenmodell überführt. Mit der Beschreibung der Anwendungslandschaft erfolgt eine Abbildung der geschäftlichen Abläufe auf bestehende oder noch zu erstellende Anwendungssysteme. Modellierungsziel dieser Phase ist nicht die Untersuchung einzelner Softwarelösungen, sondern vielmehr eine Gruppierung der in den geschäftlichen Abläufen geforderten Funktionalität zu Anwendungsgruppen als Grundlage für die Softwareentwicklung sowie eine Beschreibung von Abhängigkeiten zwischen den Anwendungen. Die vierte Phase nimmt schließlich eine Abbildung der Daten- und Anwendungsstruktur auf existierende Hard- und Softwarelösungen vor. Die abschließenden Phasen befassen sich mit Aspekten der Architekturweiterentwicklung und des Managements von Unternehmensarchitekturen.

Sowohl die methodischen Aspekte als auch die adressierten Architekturelemente sind im TOGAF Framework generisch gehalten, um eine breite Anwendbarkeit des Modells in unterschiedlichen geschäftlichen Umfeldern zu gewährleisten. Somit gibt TOGAF in erster Linie Leitplanken für die Entwicklung einer Unternehmensarchitektur vor, abstrahiert jedoch von der Diskussion spezifischer Umsetzungsvarianten. Dies kann komplementiert werden durch Musterkataloge, die eine Sammlung in der Praxis bewährter Architekturmuster (Patterns) bereithalten. [WBF+09] Ein solcher Ansatz wird in [BEL+07] vorgestellt, der zwischen drei Arten von Mustern differenziert. Vorgehensmuster adressieren typische Aufgaben, die im Rahmen einer Architekturentwicklung zu adressieren sind. Viewpoints beschreiben dabei die unterschiedlichen Perspektiven, die die verschiedenen an der Architektur beteiligten Akteure einnehmen. Mit Mustern zum Informationsmodell werden schließlich die Architekturelemente, die in den verschiedenen Sichten benötigt werden, beschrieben. Solche Muster können direkt in der Anwendung des TOGAF Frameworks eingesetzt werden (vgl. [BDMS11] für eine ausführliche Diskussion).

Konzeptmodellierung Die Konzeptmodellierung setzt den Schwerpunkt auf die Beschreibung und Abgrenzung der einzelnen Architekturelemente. Ein Konzeptmodell ist ein Metamodell einer Architektur, das die Konzepte, die den einzelnen Architekturelementen zugrundeliegen, mit ihren Eigenschaften systematisch erfasst und Beziehungen zwischen diesen aufzeigt. Mit den Konzepten wird die Struktur der Architekturbestandteile sowie deren Einbettung definiert. Ein Konzeptmodell wird in der Regel im Kontext eines bestimmten Anwendungsfeldes, zum Beispiel zur Darstellung betrieblicher Informationssysteme erfasst. [MF11]

Konzeptmodelle können in unterschiedlicher Präzision dargestellt werden. Das Spektrum reicht von einer informellen textuellen Beschreibung der Konzepte bis hin zu einer formal mathematischen Modellierung. Wir betrachten zunächst exemplarisch in Abbildung 2.1 einen auf UML basierenden semiformalen Ansatz, der einen Auszug aus den Konzeptmodellen [EHH+08, MF11] darstellt. Das abgebildete Artefaktenmodell fokussiert auf die Geschäftsarchitektur und ist simplifiziert, da es unter anderem Konzepteigenschaften nicht weiter berücksichtigt. Jedoch erhalten wir aus dem Modell einen ersten Überblick zu einigen zentralen Konzepten, die zur Beschreibung einer Unternehmensarchitektur auf geschäftlicher Ebene erforderlich sind.

Im Zentrum des Modell sehen wir eine Reihe von Artefakten zur Erfassung geschäftlicher Hand-

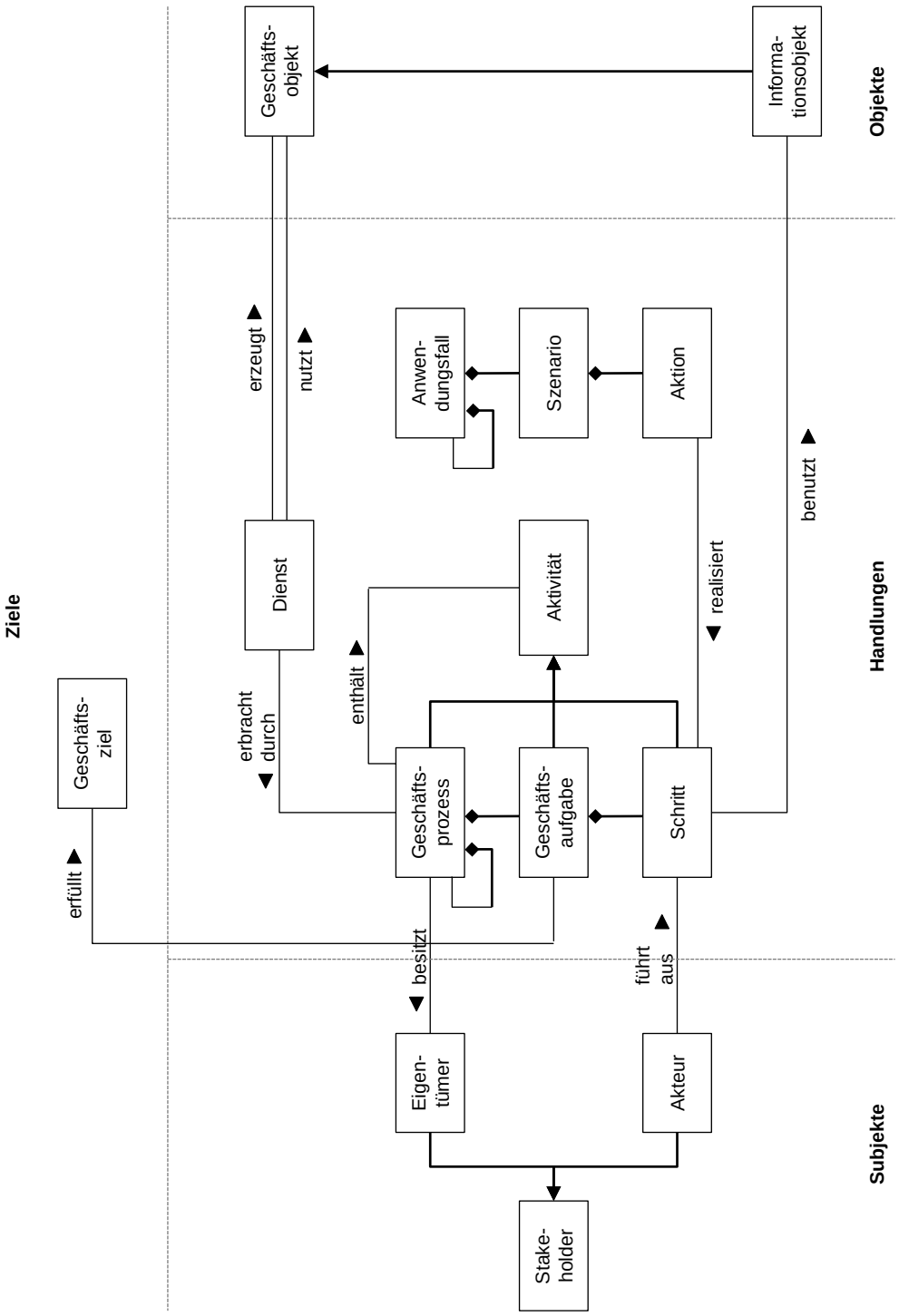

Abbildung 2.1: Vereinfachtes Konzeptmodell für geschäftliche Akteure, Handlungen und Objekte (nach [EHH+08, MF11])

lungen und Abläufe.

Geschäftsprozesse dienen zur Erfassung von geschäftlichen Abläufen und Verhaltensweisen. Diese können sowohl aus einer Innensicht durch systematische Beschreibung einer zu erwartenden Handlung als auch auf Basis von Interaktionen, die zwischen Prozessen erfolgen, erfasst werden. Interaktionen sind dabei im Kontext der betrieblichen Informationssysteme in der Regel als Informationsaustausch zu verstehen. [Ope09, Thu04] Ein Geschäftsprozess kann sich aus Teilprozessen zusammensetzen. Der Vorgang der Zerlegung eines Prozesses in Teilprozesse wird als Dekomposition bezeichnet. Im Zuge der Dekomposition kann die Beschreibung der Prozesse verfeinert werden, indem in den Teilprozessen die Beschreibung der geschäftlichen Handlung präzisiert wird. Die unterschiedliche Detailgrade der Ablaufbeschreibung können in der Konzeptbezeichnung reflektiert werden. Demnach stellen Geschäftsprozesse stark aggregierte ergebnisorientierte Handlungsbeschreibungen dar, während Geschäftsaufgaben als Bestandteile der Prozesse die Abläufe vorgehensorientiert anhand einzelner Schritte erfassen. [MF11] Ein Schritt wird dabei auch als elementare Handlung angesehen, die in einer bestimmten Architekturbetrachtung nicht weiter dekomponiert wird.

Das betrachtete Konzeptmodell verwendet den Begriff der Aktivität als Verallgemeinerung für die Konzepte Geschäftsprozess, Aufgabe und Schritt. Naturgemäß besteht ein enger Zusammenhang zwischen den Konzepten, so dass eine Aktivität auch als Bestandteil eines Prozesses angesehen werden kann. [EHH+08, Mel08, MF11] So kann jeder Geschäftsprozess – als Aktivität betrachtet – wiederum in einen anderen Prozess eingebettet sein.

Mit Anwendungsfällen können ebenfalls geschäftliche Abläufe erfasst werden. Die Beschreibung von Anwendungsfällen orientiert sich strikt an den Interaktionen zwischen den beteiligten Personen und Systemen. Ein Szenario fasst eine Sequenz erfolgter Interaktionen zusammen. Der Anwendungsfall wiederum wird als eine Menge von möglichen Szenarien, also Interaktionsfolgen, angesehen. Anwendungsfälle unterscheiden typischerweise zwischen erfolgreichen Szenarien, bei denen ein zugrundeliegendes geschäftliches Ziel erreicht wurde, und nicht erfolgreichen Szenarien. [Coc01]

Eine Aktion, die in einem Szenario eines Anwendungsfalls beschrieben ist, lässt sich nicht trennscharf von einem Prozessschritt abgrenzen. Wann immer ein geschäftlicher Prozess auf Basis von Interaktionen zwischen Personen oder Systemen erfasst wird, entsprechen die einzelnen Schritte des Prozesses den Aktionen im Sinne eines Anwendungsfalls. [MF11] interpretiert eine Aktion eines Anwendungsfalls als Realisierung eines Prozessschritts. Dabei stellt sich allerdings die Frage, wie sich eine Realisierung konzeptuell abgrenzen lässt, da Anwendungsfälle durchaus auch im fachlichen Kontext betrachtet werden können (vgl. [Coc01]). Letztlich verstehen wir daher unter Prozessschritten und Aktionen das gleiche Konzept, das in unterschiedlichen Bezugssystemen verschiedenen Bezeichnungen trägt.

Ein Dienst stellt schließlich einen aggregierten Geschäftsprozess dar, bei dem die Beschreibung einer im Prozess zu erbringenden geschäftlichen Leistung gegenüber einem oder mehreren Drittakteuren, den Dienstnehmern, im Vordergrund steht. Dienste abstrahieren typischerweise von der Leistungserbringung und werden ergebnisorientiert formuliert. Der Dienst stellt damit eine besondere Ausprägung eines Geschäftsprozesses dar.

Das exemplarische Artefaktenmodell enthält zudem Konzepte zur Beschreibung von Akteuren und Objekten, die zur vollständigen Erfassung einer Unternehmensarchitektur erforderlich sind. Die Akteure auf der linken Seite des Modells können in unterschiedlicher Weise zu den Handlungen in Bezug stehen. Auf der einen Seite werden Prozesse durch Akteure ausgeführt. Andererseits weist das Modell jedem Geschäftsprozess auch einen Eigentümer zu, der verantwortlich für diesen ist. Zudem können geschäftliche Abläufe Geschäftsobjekte sowohl als Grundlage verwenden als auch erzeugen. Da der extern ersichtliche Leistungsumfang eines Prozesses durch Dienste ausgedrückt wird, stehen Geschäftsobjekte zu Diensten im Bezug, indem sie von diesen konsumiert und generiert werden. Geschäftsobjekte können sowohl materieller Natur, zum Beispiel Unterlagen, als auch immateriell, insbesondere Daten, sein. Schließlich sieht das Modell auch die Erfassung von geschäftlichen Zielen vor.

In Summe spannt das vorgestellte Artefaktenmodell den Begriffsraum für die architekturelle Beschreibung betrieblicher Informationssysteme auf und nimmt eine erste Einordnung der Konzepte vor mit Schwerpunkt auf den geschäftlichen Abläufen. Wir erkennen, dass die unterschiedlichen Handlungskonzepte den Fokus auf verschiedene Aspekte legen, können jedoch anhand des Modells keine klare Abgrenzung der Konzepte vornehmen. Vielmehr zeigt sich, dass ein enger Zusammenhang zwischen den Begriffen besteht. Dies gilt insbesondere auch für den Dienstbegriff und motiviert eine Formalisierung der eingeführten Konzepte, um eine präzise Einordnung und Abgrenzung zu ermöglichen.

Zusammenfassung Die in den letzten Abschnitten eingeführten Referenzmodelle repräsentieren unterschiedliche Varianten der Beschreibung von Unternehmensarchitekturen. Das Zachman Framework zielt primär auf die Strukturierung von Architekturbestandteilen ab, indem es entlang einer zweidimensionalen Struktur einerseits zwischen fachlicher und technischer Betrachtung, andererseits zwischen den Arten von Architekturelementen differenziert.

Die Konzeptmodelle befassen sich hauptsächlich mit den Eigenschaften der einzelnen Architekturelemente und deren Beziehungen. Dies unterscheidet sie vom Zachman Framework, das nur eine Taxonomie für die Elemente vorgibt, ohne auf diese im Einzelnen einzugehen. Bei den Konzeptmodellen dagegen steht die Strukturierung nicht im Vordergrund, sondern ist Resultat aus der Beschreibung der Architekturbestandteile.

Vorgehensmodelle wie TOGAF fokussieren schließlich hauptsächlich auf die Methodik zur Erstellung und Weiterentwicklung von Architekturen. Die Konzepte werden hier als Ergebnis der einzelnen Entwicklungsphasen angesehen.

Im Folgenden werden wir uns schwerpunktmäßig mit der Erfassung von Handlungen und Abläufen auf geschäftlicher Ebene befassen. In den bereits eingeführten Referenzmodellen ist die architekturelle Beschreibung geschäftlicher Abläufe in unterschiedlicher Weise thematisiert worden. Das Zachman-Framework sieht dazu eine eigene Sicht, die Wie?-Dimension, vor, entlang der das Verhalten von Unternehmenssubjekten in fachlicher und technischer Hinsicht erfasst wird. Das Framework verweist in dieser Dimension auf den Begriff Prozess zur Beschreibung von Verhaltensweisen und Abläufen. Der Begriff wird sowohl auf fachlicher Ebene im Sinne eines Geschäftsprozesses als auch auf technischer Ebene im Sinne eines technischen Prozesses oder auch Programmablaufs verwendet. [Zac08]

TOGAF sieht ebenfalls in der Phase der Geschäftsarchitekturmodellierung eine Beschreibung von Geschäftsprozessen vor. Prozesse werden dabei interaktionsorientiert modelliert und erfassen den Informationsaustausch zwischen den Akteuren der betrieblichen Informationssysteme. [Ope09] Das vorgestellte Konzeptmodell hat schließlich einen Überblick zu den unterschiedlichen begrifflichen Ausprägungen von Geschäftshandlungen enthalten. Der Prozess nimmt dabei als Grundkonzept zur Beschreibung geschäftlicher Abläufe eine zentrale Rolle ein. Bevor wir uns mit der exakten Begriffsbildung und Formalisierung von Prozessen und daraus abgeleiteten Konzepten befassen, motivieren wir im folgenden Abschnitt zunächst noch die Dienstorientierung in Unternehmensarchitekturen.

2.2 Dienstorientierte Architekturen

Mit der Beschreibung von Unternehmensarchitekturen kann im Kontext der betrieblichen Informationsverarbeitung ein enger Bezug zwischen Fachlichkeit und technischer Realisierung hergestellt werden. Die ganzheitliche architekturelle Betrachtung ermöglicht eine Abbildung von fachlichen Entitäten und Abläufen auf deren systemseitige Realisierung. Dies vereinfacht Anpassungsprozesse, da Abhängigkeiten besser erkannt werden und die Betrachtung so auf den betroffenen Teilbereich der Architektur beschränkt werden kann. Allerdings fordern Unternehmensarchitekturen, wie wir sie bisher kennengelernt haben, keine Kongruenz zwischen der fachlichen und technischen Ebene. So kann es beispielsweise vorkommen, dass eine fachliche Aufgabe durch unterschiedliche Anwendungen unterstützt wird oder eine Anwendung mehrere fachliche Aufgaben gleichzeitig erfüllt. Die Intension von Unternehmensarchitekturen liegt in der konsistenten Darstellung betrieblicher Entitäten und Handlungen, ohne dabei Vorschriften bezüglich deren Anordnung und Ausrichtung zu machen.

Bedingt durch sich ändernde geschäftliche Rahmenbedingungen sowie betriebliche Weiterentwicklungen sind Unternehmensarchitekturen in der Regel dynamischer Natur. Selbst wenn sich die Struktur der technischen Realisierung zu einem Zeitpunkt perfekt mit der Struktur der fachlichen Anforderungen deckt, ist es kaum zu verhindern, dass durch geschäftliche oder auch technisch erforderliche Anpassungen die Kongruenz in der Folge wieder aufgehoben wird. Im Umgang mit Unternehmensarchitekturen ergibt sich also die stetige Herausforderung des Nachhaltens von geschäftlichen Änderungen auf technischer Ebene. Diese Herausforderung ergibt sich insbesondere aus der Tatsache, dass Anwendungslandschaften in Unternehmen häufig historisch gewachsen sind und Softwareeinheiten mit umfassenden Funktionsumfang aufweisen. Dies kann bis hin zu Großanwendungen reichen, die monolithische Züge tragen. Entsprechend gestalten sich Anpassungen solcher umfassenden Anwendungen als komplex und zeitaufwändig. Die Problematik wird zusätzlich dadurch verstärkt, dass eine geschäftliche Modifikation die Anpassung mehrerer Anwendungen zur Folge haben kann.

Die Dienstorientierung in Architekturen zielt darauf ab, die Ausrichtung der Implementierung an der Fachlichkeit deutlich zu vereinfachen und somit das Beibehalten von Kongruenz zwischen den Ebenen auch im Fall von häufigen Anpassungen zu ermöglichen. [EHH+08] Die Grundidee besteht darin, geschäftliche Abläufe auf Basis von kleinen abgrenzten Bausteinen, den Diensten, zu modellieren. Die Dienste werden typischerweise auf Akteure bezogen und beschreiben

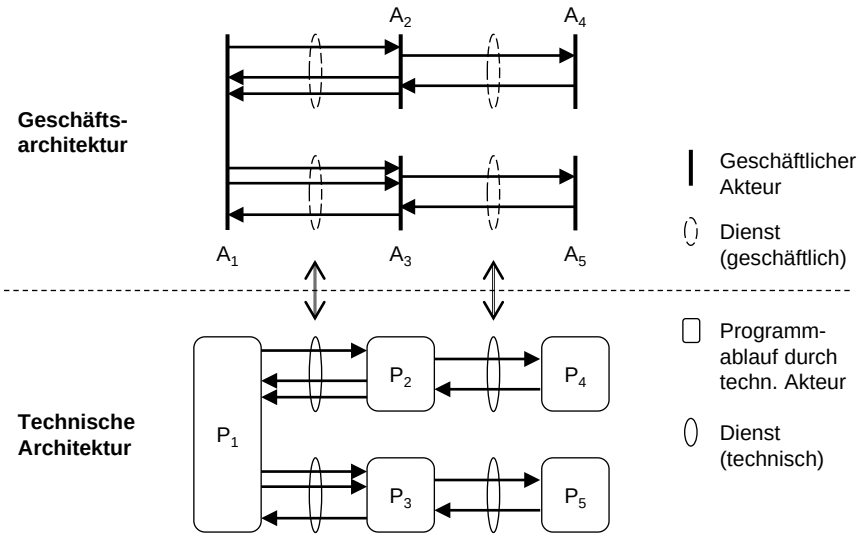

Abbildung 2.2: Ausrichtung von fachlicher und technischer Architektur anhand von Diensten

eine Leistung, die ein gegebener Akteur gegenüber einem anderen Akteur erbringt, anhand zu beobachtender Interaktionen zwischen den Akteuren.

Gleichzeitig wird auf technischer Ebene ein analoges Konstrukt, zum Beispiel Web Services, eingesetzt. Durch das Bausteinprinzip müssen im Fall von Änderungen nicht mehr komplexe Anwendungen modifiziert werden, sondern – zumindest im einfachsten Fall – nur noch die Anordnung der Bausteine auf technischer Ebene der neuen Konstellation der Fachlichkeit entsprechend angepasst werden. Die Dienste per se werden so gestaltet, dass sie funktional einen abgrenzten Teilbereich mit geringen Abhängigkeiten zu anderen Diensten abbilden. Auf diese Weise können Änderungen an den Diensten an sich minimiert werden.

Die Abbildung 2.2 skizziert schematisch den Aufbau einer dienstorientierten Architektur. Auf der fachlichen Ebene sind fünf Akteure dargestellt, die miteinander interagieren – dargestellt durch entsprechende Pfeile. Jedes Interaktionsverhalten zwischen jeweils zwei Akteuren ist dabei in einem entsprechenden Dienst festgelegt. Auf der technischen Ebene nutzen die fünf Softwarekomponenten, die durch Rechnersysteme ausgeführt werden, der gleichen Struktur folgend Dienste, um miteinander zu kommunizieren. Wenn sich auf der fachlichen Ebene Änderungen im Interaktionsverhalten ergeben, können diese so direkt auf die Implementierungsebene abgebildet werden. Dies folgt der Annahme, dass die Handlungen der Akteure als Komposition von Diensten beschrieben werden können, und dass die Dienste feingranular genug definiert sind, so dass fachliche Anpassungen durch eine Modifikation der Dienstkomposition umgesetzt werden können.

SOA hat also sowohl eine fachliche als auch eine technische Perspektive. Es ist Designstil, indem es die konsequente Verwendung einer geschäftlich motivierten Modularität fordert, und bezieht

sich sowohl auf die technische Implementierung als auch die fachliche Prozessbeschreibung. [PVDH07]

Die technische Perspektive von SOA ist nicht neu. Die Schaffung von standardisierten, entkoppelbaren und verteilten Modulen, um eine flexiblere Nutzung und vereinfachte Wiederverwendung von Funktionalität zu ermöglichen, wurde bereits durch Standards wie CORBA [Ope11], RPC [BN84, Thu09] und COM/DCOM [Mic12] adressiert. Im Kontext von SOA wird ein vergleichbarer Leitungsumfang durch Web Services angeboten, die sich als de-facto Standard für die technische Umsetzung von Dienstarchitekturen etabliert haben. [NL05] Die Diskussion von SOA auf technischer Ebene fokussiert dabei primär auf die Beschreibung von Implementierungsstandards zum Aufbau einer für Web Services geeigneten Infrastruktur. Diese umfasst insbesondere eine Schnittstellendefinition für Dienste, Mechanismen zur Verwaltung und Auffindung von Diensten sowie plattformunabhängige Übertragungsprotokolle.

Die fachliche Perspektive von SOA kann dagegen in der Tat als Innovation angesehen werden. Er sieht eine Verankerung von Modularität auf geschäftlicher Ebene vor, indem Prozesse aus fachlichen Diensten flexibel zusammengesetzt werden. Damit überträgt SOA die in den soeben benannten technischen Standards etablierte Modularität auch auf die fachliche Ebene aus. Dies ist Grundlage für die Herstellung und insbesondere Aufrechterhaltung von Kongruenz zwischen der fachlichen und technischen Architektur in dynamischen Geschäftsumgebungen. Dabei ist die Ausrichtung der Ebenen keine inhärente Eigenschaft von SOA, wird aber durch die einheitliche Modularisierung deutlich vereinfacht und unterstützt.

In SOA-Definitionen wird häufig die Technologieunabhängigkeit als charakterisierende Eigenschaft von dienstorientierten Architekturen hervorgehoben. [Mel08] Im Widerspruch dazu werden die Eigenschaften von SOA allerdings zumeist anhand von Web Services oder vergleichbaren technischen Umsetzungskonstrukten diskutiert beziehungsweise erst auf technischer Ebene ausreichend konkretisiert. [PVDH07, NL05]

2.2.1 Zielsetzung

Mit der Einführung von dienstorientierten Architekturen in Unternehmen wird in der Regel eine klare Erwartung hinsichtlich eines Produktivitätsgewinns verbunden. Diese übergreifende Zielsetzung lässt sich wiederum unterteilen in ein Spektrum einzelner Zielsetzungen, die mit SOA assoziiert werden. Wir geben im Folgenden einen Überblick zu diesen Zielsetzungen und gruppieren diese entlang von vier Kategorien: Effizienz, Agilität, Qualität sowie die fachliche Orientierung. Die Gruppen sind nicht gänzlich überschneidungsfrei, decken in Summe aber im Wesentlichen den Erwartungshorizont ab, der mit SOA verknüpft wird.

Effizienz Der Einsatz dienstorientierter Architekturen geht häufig einher mit dem Anspruch auf Effizienzsteigerungen. Dabei steht vor allem die betriebliche Informationsverarbeitung im Fokus, die durch die Verwendung von Diensten den gleichen Unterstützungsgrad mit einem geringeren Ressourcenaufwand erreichen soll. Darüber hinaus kann sich der Effizienzgewinn aber auch auf die Fachseite beziehen.

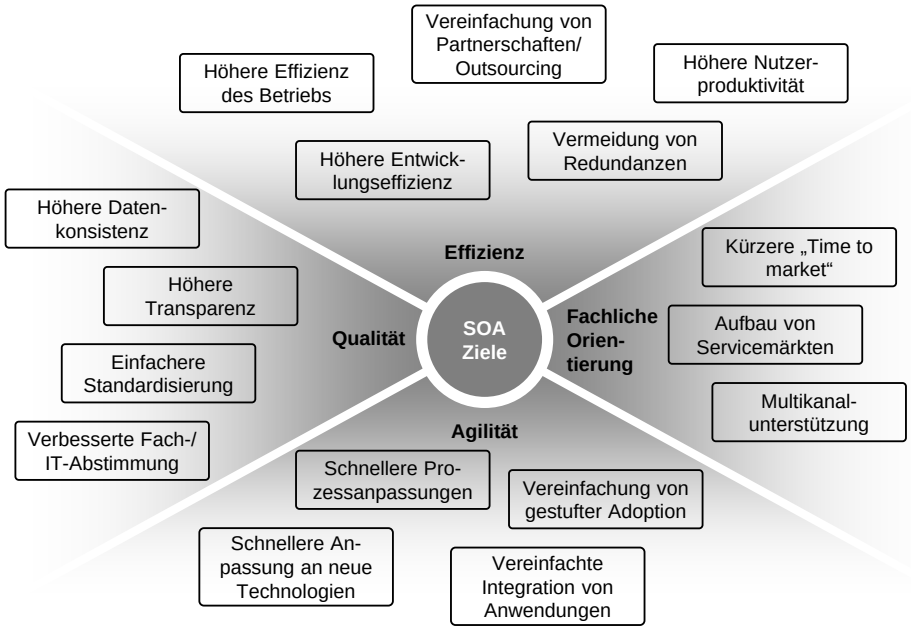

Abbildung 2.3: SOA Zielsetzungen

SOA zielt auf eine Effizienzsteigerung in der Softwareentwicklung ab, indem es durch die Modulbildung die Wiederverwendung von Funktionalität vereinfacht. Zudem erleichtert die konsequente Modularisierung auch den Softwareentwurf und die Abstimmung zwischen den beteiligten Entwicklern. [Erl08] Ein weiterer Effekt ist die Vermeidung von Redundanzen, da durch den feingranularen Schnitt von Diensten Überschneidungen besser erkannt werden können.

SOA kann auch einen positiven Einfluss auf die Betriebseffizienz von Softwareumgebungen haben, indem auftretende Probleme leichter eingegrenzt und adressiert werden können. Zusätzlich können Softwareaktualisierungen einfacher umgesetzt werden, wenn nur noch einzelne Dienste und nicht ganze komplexe Softwarelandschaften neu ausgerollt werden müssen. Dienstorientierte Architekturen unterstützt weiterhin die verteilte Nutzung von Software und ermöglichen durch die gut abgrenzbaren Dienstbausteine eine Auslagerung von Teilfunktionalitäten an externe Dienstleister beziehungsweise die Kooperation mehrerer Unternehmen. Die verbesserte Austauschbarkeit von Diensten leistet dabei auch einen Beitrag zu einer größeren Herstellerunabhängigkeit. [Erl08]

Schließlich kann SOA auch anwenderseitig Effizienzsteigerungen bewirken, wenn durch den Einsatz von standardisierten Dienstumgebungen die Interaktion zwischen unterschiedlichen Systemen vereinfacht und damit manuelle Aufwände reduziert werden.

Agilität Die Einführung einer Dienstorientierung in Unternehmensarchitekturen wird häufig mit einer höheren Agilität und Dynamik verbunden. [PVDH07, EHH+08, RLPB07, Erl08] In

vielen Fällen wird diese auf die schnellere softwareseitige Umsetzung von Änderungen in Geschäftsprozesse bezogen, die wir bereits zuvor diskutiert haben. SOA kann aber auch die Anpassung an neue Technologien begünstigen, da der Leistungsumfang von Diensten unabhängig von der zugrundeliegenden Technologieplattform definiert und damit ein Technologiewechsel vereinfacht wird.

Dienstarchitekturen erleichtern auch die gestufte Adoption von neuen Funktionalitäten. Bedingt durch die Modularität können Softwareaktualisierungen feingranularer ausgerollt werden. Anstehende Aktualisierungen müssen nicht in den Release-Zyklen der übergreifenden Software gebündelt werden, sondern können pro Dienst umgesetzt werden. Auf diese Weise kann die Weiterentwicklung von Funktionalität beschleunigt und dynamisiert werden.

Zuletzt ermöglicht die modulare Struktur einer dienstorientierten Unternehmensarchitektur eine vereinfachte Integration von Anwendungen. So können heterogene Anwendungslandschaften, die sich beispielsweise nach einem Unternehmenszusammenschluss ergeben, besser vernetzt und harmonisiert werden.

Qualität Die einheitliche Strukturierung der fachlichen und technischen Architektur anhand des durch die Dienste vorgegebenen Bausteinprinzips erleichtert die Herstellung und Einhaltung von Konsistenz zwischen Fachseite und Realisierung. Das Bausteinprinzip der Dienste ermöglicht eine feingranulare Zuordnung von fachlichen Teilprozessen zu den entsprechenden Softwarekomponenten, die diese unterstützen. Im Fall von Änderungen können so Abhängigkeiten schneller erkannt und berücksichtigt werden. Dienstarchitekturen unterstützten somit den Abgleich zwischen Informationssystemen und Geschäftsprozessen. [Erl08, NL05]

Ferner kann durch den Einsatz von Diensten auch eine größere Datenkonsistenz erreicht werden. Dies ist der Fall, wenn der Zugriff auf Unternehmensdaten in Diensten gebündelt wird. Wenn alle Zugriffe auf einen bestimmten Datenpunkt durch einen gegebenen Dienst erfolgen, können Konsistenzbedingungen einheitlich überprüft und eingehalten werden. [Erl08]

Schließlich tragen Dienstarchitekturen zu einer höheren Standardisierung bei. Dies bezieht sich sowohl auf die fachliche als auch auf die technische Perspektive. In fachlicher Hinsicht können Routineaufgaben in entsprechend modularen Diensten ausgelagert und dann flexibel eingebunden werden. In technischer Hinsicht unterstützen die Fokussierung auf eine Schnittstellensicht eine verteilte und plattformunabhängige Durchführung von Berechnungen. Zudem erhöht das Modularitätsprinzip der Dienste die Transparenz sowohl auf fachlicher wie auch auf technischer Ebene.

Fachliche Orientierung Dienstorientierte Architekturen richten sich typischerweise stark an den Anforderungen der Fachlichkeit aus. So werden Dienste aus einer fachlichen Sicht heraus definiert, um die in der Architektur beschriebenen Abläufe soweit möglich auf die geschäftlichen Bedürfnisse abzustimmen.

Mit der Einführung von Diensten können zudem neue Geschäftsfelder erschlossen werden. Die Zusammenfassung von Funktionalität in standardisierten, abgegrenzten und plattformunabhängigen Diensten ermöglicht es, diese auf einfache Art und Weise Externen zugänglich zu machen

und auf diese Weise auch zu monetarisieren. Dienste unterstützen die föderale Nutzung von Funktionalität und tragen zu einer verbesserten Arbeitsteilung bei, was spezialisierte Anbieter als neue Absatzchancen für sich begreifen. Durch die erhöhte Flexibilität und die damit einhergehende gesteigerte Effizienz können sich Unternehmen, die dienstorientierte Architekturen anwenden, zudem Wettbewerbsvorteile verschaffen, indem sie schneller auf Marktbedürfnisse reagieren.

Die Kapselung von Funktionalität in Diensten unterstützt auch die Multikanalfähigkeit von Anwendungen, da die Präsentationsschichten der unterschiedlichen Kanäle auf den gleichen Dienst zurückgreifen können. Somit kann ein einheitlicher und effizient wartbarer Funktionsumfang über die Kanäle hinweg gewährleistet werden. Die einfache Erschließbarkeit zusätzlicher Kanäle wiederum kann in der Generierung von neuen Kundengruppen resultieren.

2.2.2 Gestaltungsprinzipien

Den in der Literatur diskutierten Zielsetzungen und Definitionen für dienstorientierte Architekturen liegen drei zentrale Gestaltungsprinzipien zugrunde. Wir werden diese im Folgenden zusammen mit ihren Eigenschaften diskutieren.

Modularität Das erste Gestaltungsprinzip ist die Bündelung von Funktionalität auf fachlicher und technischer Ebene in abgegrenzten Einheiten, den Diensten. Dem zugrunde liegt die Idee eines Bausteinprinzips, das es erlaubt, komplexere Handlungsabläufe flexibel aus feingranularen Teilabläufen zusammenzusetzen. Die Strukturierung dieser Bausteine orientiert sich an den geschäftlichen Gegebenheiten und ist in der Regel nach Akteuren ausgerichtet. [EHH+08, NL05] Der erwartete Mehrwert von SOA beruht auf der Annahme, dass die Kongruenz zwischen der fachlichen und technischen Architekturebene für kleine abgegrenzte Bausteine leichter aufrecht erhalten werden kann als für komplexe übergreifende Prozesse und Anwendungen. Die Modularität von SOA bildet insofern eine Brücke zwischen Fachlichkeit und technischer Realisierung.

Damit die Dienste als Bausteine einer SOA möglichst gut voneinander abgrenzbar sind, werden sie abgeschlossen (self-contained) gestaltet. [PVDH07, Erl08, Ope12b] Dies bedeutet, dass die Informationen, auf deren Basis die in ihnen enthaltene Funktionalität ausgeführt wird, nur über eine definierte Schnittstelle übergeben wird. Genauso werden die Ergebnisse nur über besagte Schnittstelle ausgegeben. Es findet jenseits der Schnittstelle kein Informationsaustausch mit anderen Entitäten oder Datenbanken statt. Daraus folgt, dass sich der Zustand des Dienstes nur aus den Eingaben an der Schnittstelle ergibt, die über Zeit zu betrachten sind. Das Verhalten eines Dienstes ergibt sich aus den Ausgaben an der Schnittstelle, die in Relation zu den Eingaben gesehen werden. Dies wird auch als ergebnisorientierte Architektur bezeichnet, da nicht die Berechnung, sondern das an der Schnittstelle beobachtbare Ergebnis im Vordergrund steht. [Mel08]

Zudem wird als Kriterium für Modularität in SOA häufig gefordert, dass bei Diensten eine lose Kopplung (loose coupling) und eine starke Kohäsion (strong cohesion) besteht. [PVDH07, RLPB07, Mel08, Erl08, NL05] Darunter ist zu verstehen, dass Dienste als Bausteine einer Architektur so geschnitten werden, dass die Abläufe, die im Dienst zusammengefasst sind, eine

hohe Interaktionsdichte aufweisen, der Dienst mit anderen Diensten aber nur so wenig Interaktion wie nötig erfordert. Dies trägt in Verbindung mit der abgeschlossenen Gestaltung dazu bei, dass ein Dienst möglichst geringe Abhängigkeiten zu anderen Abläufen aufweist. Eine solche Gestaltung ist Grundlage für eine vielseitige und flexible Einsetzbarkeit und fördert damit die Wiederverwendbarkeit eines Dienstes, die als charakteristisches Merkmal angesehen wird. [Erl08] Dabei ist zu beachten, dass Kopplung und Kohäsion formal nur schwer messbar sind, da sie nicht nur von der Interaktionsdichte, sondern auch der Qualität der Interaktionen beeinflusst werden. Die Einhaltung einer losen Kopplung und starken Kohäsion sind somit vor allem als Gestaltungsrichtlinie für Dienste zu verstehen. Die Ausrichtung von Diensten an Akteuren begünstigt in der Regel die Einhaltung dieser Kriterien, da Arbeitsteilung in Unternehmen eine der losen Kopplung und starken Kohäsion vergleichbaren Zielsetzung folgt.

Komponierbarkeit Das zweite Gestaltungsprinzip für Dienstarchitekturen ist die Zusammensetzbarkeit von einzelnen Diensten zu neuen übergreifenden Diensten. Dies basiert auf dem in der Informatik häufig angewendetem Prinzip „Divide & Conquer", mit dem komplexe Probleme iterativ in einfacher zu behandelnde Teilprobleme zerlegt und dann auf Teilproblemebene gelöst werden. [CLRS01, BB96] Der Bausteinmetapher folgend ermöglichen Dienstarchitekturen die Beschreibung von komplexen Handlungsabläufen auf fachlicher und technischer Ebene, indem diese sukzessive aus einfacher zu beschreibenden Teilhandlungen zusammengesetzt werden. So können in mehreren Stufen aus elementaren Abläufen komplexe Problemstellungen definiert werden, ohne diese monolithisch beschreiben zu müssen. [PVDH07, Erl08]

Bei der Zusammensetzung von Diensten können verschiedene Kompositionsmuster angewendet werden. Im einfachsten Fall werden zwei Dienste parallel oder sequentiell ausgeführt und die Kombination der beiden wiederum als komplexerer Dienst betrachtet. Bei der Komposition können aber auch zwei Dienste alternativ in Abhängigkeit einer Bedingung oder ein Dienst in iterativer Weise zu einem umfassenderen Dienst komponiert werden.

Arbeitsteilige Erbringung Das dritte Gestaltungsprinzip ist die arbeitsteilige Erbringung von Dienstleistungen. Unternehmerische Abläufe werden bedingt durch betriebliche Spezialisierung häufig arbeitsteilig erbracht. Die Arbeitsteilung erfolgt dabei sowohl auf fachlicher als auch auf technischer Ebene. Auf fachlicher Ebene können unterschiedliche Abteilungen oder auch andere Unternehmen, beispielsweise Lieferanten, an den geschäftlichen Abläufen beteiligt sein. Auf technischer Ebene kann Funktionalität auf unterschiedlichen Plattformen realisiert sein. Häufig bedingt eine fachliche auch eine technische Heterogenität, indem unterschiedlichen Abteilungen und Organisationen auf ihre jeweiligen Systemplattformen zurückgreifen.

Den Prinzipien der Modularität und Komponierbarkeit folgend werden Dienste als Bausteine einer Architektur angesehen und die Handlungen eines Unternehmens modular durch Komposition einzelner Dienste beschrieben. Durch die Bausteinstruktur kann eine arbeitsteilige Erbringung leicht in dienstorientierten Architekturen abgebildet werden. In den Dienstdefinitionen werden die Leistungsumfänge beschrieben, die von den einzelnen Bausteinen erwartet werden. Die einzelnen Dienste können dabei durch unterschiedliche Akteure auf fachlicher und techni-

scher Ebene erbracht werden, solange durch die dienstorientierte Architektur die Interoperabilität zwischen den Diensten gewährleistet ist.

Eine wesentliche Voraussetzung hierfür ist die Standardisierung der Interaktionen zwischen den einzelnen Diensten. [PVDH07, Mel08, NL05] Dazu wird in der Regel zwischen der Dienstschnittstelle, von der aus Interaktionen ausgehen, und der eigentlichen Umsetzung des Dienstes, der Dienstimplementierung, unterschieden. Die Schnittstelle von Diensten wird dabei so gestaltet, dass sie von den plattformspezifischen Details der Implementierung abstrahiert und damit Interaktionen zwischen Diensten ermöglicht unabhängig von der technischen Plattform, auf der diese realisiert sind. [PVDH07, Mel08] Zudem muss die Architektur eine verteilte Realisierung der Dienste zulassen, so dass Dienste orts- und systemunabhängig miteinander kommunizieren können. [PVDH07, Mel08, Erl08]

Eine besondere Ausprägung der arbeitsteiligen Diensterbringung ist die dynamische Bindung. Wir sprechen von dynamisch gebundenen Diensten, wenn mehrere Akteure den gleichen Dienst erbringen können und erst während der Durchführung eine geschäftlichen Handlung eine Entscheidung getroffen wird, welcher der Akteure den Dienst erbringt. [PVDH07, KBS07, Mel08] Die Entscheidung wird dabei häufig durch nicht funktionale Gesichtspunkte geleitet, beispielsweise Kosten für und/oder Qualität der angebotenen Leistung.

2.2.3 Aktuelle Konzeptauslegung

Dienstorientierte Architekturen haben in den letzten Jahren sowohl in der Wissenschaft, als auch in der Praxis breite Beachtung gefunden. Entsprechend hat sich eine Vielzahl unterschiedlicher SOA-Definitionen herausgebildet. Wir werden im Folgenden einen Überblick zum aktuellen Verständnis dienstorientierter Architekturen herausarbeiten und untersuchen, wie die soeben eingeführten Gestaltungsprinzipien in diesen reflektiert sind. Betrachten wir zunächst exemplarisch drei Definitionen von SOA mit praxisorientiertem Fokus:

- „Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. [...] Services are the mechanism by which needs and capabilities are brought together." [OAS06]
- „Service-Oriented Architecture (SOA) is an architectural style that supports service orientation. Service orientation is a way of thinking in terms of services and service-based development and the outcomes of services. [...] An architectural style is the combination of distinctive features in which architecture is performed or expressed." [Ope12b]
- „Service Oriented Architecture (SOA) is a business-centric IT architectural approach that supports integrating your business as linked, repeatable business tasks, or services." [IBM12]

Wir stellen fest, dass die Definitionen die Prinzipien der Modularität, Komposition und Arbeitsteilung auf fachlicher Ebene ansprechen, jedoch viel Deutungsraum für die konkrete Ausgestaltung offen lassen. Um ein genaueres Verständnis zur Auslegung von SOA zu bekommen, ist es erforderlich, sich mit den Merkmalen und Eigenschaften, die die einzelnen Definition mit SOA

assoziieren, auseinanderzusetzen. Dazu differenzieren wir zwischen der fachlichen und technischen Ebene.

Fachliche Auslegung In der fachlichen Auslegung werden Dienste als geschäftliche Leistung betrachtet, die ein Dienstanbieter oder Dienstgeber gegenüber einem Dienstnehmer erbringt. [EHH+08, PVDH07, Mel08, NL05] Der Leistungsumfang wird in einem Vertrag festgelegt, der sowohl funktionale als auch nicht funktionale Aspekte beinhaltet. [KBS07] Dienstverträge werden in der Regel ergebnisorientiert beschrieben. Sie abstrahieren von den internen geschäftlichen Abläufen, die in Prozessen beschrieben sind, und beschränken sich stattdessen auf eine Spezifikation der Interaktionen an der Prozessschnittstelle aus einer Außensicht. [EHH+08, Mel08] Durch die konsequente Schnittstellenfokussierung erfolgt eine Kapselung des Prozesses und es wird damit dem Modularitätsprinzip von Diensten Rechnung getragen.

Der Vertrag beschreibt die Funktionalität des Dienstes, indem er Syntax und Semantik der Dienstschnittstelle festhält. Zudem kann er übergreifende Ziele sowie Beschränkungen für die Nutzung des Dienstes enthalten. [KBS07] Der Umfang und die Art der Funktionalitätsbeschreibung ist im Allgemeinen nicht festgelegt. Dies führt zu einer großen Varianz in der Funktionsbeschreibung und birgt insofern Risiken, als dass nicht vollständige oder eindeutige Leistungsbeschreibungen zu Inkonsistenzen und Überschneidungen führen können. Dies erschwert die konsequente Einhaltung des Modularitätsprinzips. Der Vertrag kann weiterhin nicht funktionale Aspekte, beispielsweise Verfügbarkeit und Zugriffsbestimmungen, enthalten.

Der Vertrag gilt auch als Grundlage für die arbeitsteilige Erbringung eines Dienstes. Er definiert den Leistungsumfang, den der Dienstanbieter gegenüber dem Dienstnehmer zu erbringen hat. Zudem kann der Vertrag auch geforderte Interaktionen mit Dritten umfassen. Der Vertrag ist dabei als minimale Anforderung anzusehen. Jeder Dienst kann durch eine Vielzahl von Prozessen erbracht werden, die diesen Mindestumfang erfüllen, sich aber in Ablaufdetails oder zusätzlichen Funktionalitäten unterscheiden. Die Auswahl des konkreten Prozesses zur Erbringung der Dienstleistung ist dem Dienstgeber überlassen.

Zur Vermittlung zwischen Dienstanbietern und Dienstnehmern werden häufig Dienstmittler eingebunden. Diese verwalten ein Verzeichnis aller verfügbaren Dienste zusammen mit den jeweiligen Anbietern. Dienstgeber können die von ihnen angebotenen Dienste bei den Dienstmittlern registrieren. Anschließend können Dienstnehmer Anfrage beim Dienstmittler zu einem bestimmten Dienst stellen und auf diese Weise eine Auskunft erhalten, welche Dienstgeber die von ihnen nachgefragten Dienste anbieten. [PVDH07, Mel08]

Die Komposition mehrerer Dienste zu einem strukturiertem Ablauf wird überwiegend als Prozess bezeichnet. In den Prozessen wird dabei beschrieben, in welcher Reihenfolge die Dienste ausgeführt werden und welche Abhängigkeiten zwischen den Diensten bestehen. [Mel08, Ope12b, Erl08] Die Herausforderung liegt hier in einer präzisen Abgrenzung zwischen Dienst und Prozess, da jede Ablaufbeschreibung aus einer Außensicht betrachtet auch einen Dienst darstellen und jeder Dienst wiederum als Prozess angesehen werden kann. Bei informellen Beschreibungsansätzen kann es daher zu begrifflichen Überlagerungen und Unschärfen kommen.

Teilweise wird eine Klassifikation der Dienste ja nach Granularität in Basis-, Zwischen- und Unternehmensdienste vorgenommen. [KBS07] Basisdienste stellen feingranulare Grundfunktiona-

litäten zur Verfügung, Zwischendienste aggregieren diese zu komplexeren Funktionsumfängen und Unternehmensdienste fassen die Zwischendienste schließlich zu in sich abgeschlossen Produkten des Unternehmens zusammen. Die Grenzen zwischen den Kategorien sind jedoch nicht präzise festlegbar und liegen somit im Ermessensspielraum des Dienstarchitekten.

Technische Auslegung Auf technischer Ebene werden Dienste prinzipiell unabhängig von einer bestimmten Anbieterlösung, wie zum Beispiel Web Services, definiert. Es wird in der Regel unterschieden zwischen der Dienstschnittstelle und der Dienstimplementierung. Die Dienstschnittstelle wird plattformunabhängig realisiert, so dass sie von unterschiedlichen Systemumgebungen aus aufgerufen werden kann. Einige Definitionen charakterisieren die Dienstimplementierung für den Dienstnutzer als unsichtbar, da der Zugriff auf die Funktionalität des Dienstes ausschließlich über die Schnittstelle erfolgt. [PVDH07, Ope12b, NL05] Dies liegt begründet in der Sichtweise, dass ein Dienst im Wesentlichen durch seine Schnittstelle bestimmt ist und als solcher getrennt von der Komponente zu betrachten ist, die ihn implementiert. [BKM07] Auf der anderen Seite können Dienste aber auch als Ergebnis der Berechnung einer Software-Komponente betrachtet und insofern mit der Komponente als ganzem, nicht nur deren Schnittstelle assoziiert werden. [KBS07]

Zwischen der Implementierung und der Schnittstelle eines Dienstes besteht eine n:m Beziehung. [PVDH07, BKM07] Jede Implementierung kann mehrere Schnittstellen besitzen, über die sie ihre Funktionalität anbietet. Genauso kann eine Schnittstelle auf eine Kombination aus mehreren Implementierungskomponenten zurückgreifen. Wenn ein Dienst nur einen Teilaspekt einer Implementierung abdeckt, wird er auch als partielle Sicht auf die zugrundeliegende Komponente wahrgenommen. [BKM07]

Dienste auf technischer Ebene richten sich in ihrer Funktionalität häufig an den fachlichen Diensten aus, um eine möglichst akkurate technische Unterstützung der geschäftlichen Anforderungen sicherzustellen. [NL05] Im Kontext der SOA-Diskussion gibt es jedoch auch den Gedanken, Dienstinventare anzulegen, in denen Softwaredienste für eine spätere Verwendung vorgehalten werden. Diese Dienste müssen zum Zeitpunkt der Definition keinen direkten Bezug zur Fachlichkeit aufweisen. [Erl08]

Obwohl SOA unabhängig von einer bestimmten Anbieterlösung definiert ist, werden seine Eigenschaften auf technischer Ebene häufig anhand von Web Services beschrieben. Auf diese Weise kann die Gestaltung der Dienste präziser anhand konkreter Implementierungsbeispiele diskutiert werden.

Web Services werden auf Basis ihrer Schnittstelle definiert, die in einer XML-basierten standardisierten Beschreibungssprache WSDL festgelegt wird. [PVDH07, Mel08, NL05] Die funktionale Schnittstelle wird in WSDL systematisch erfasst. Zunächst werden Typen für die Ein- und Ausgaben des Dienstes festgelegt. Die Ein- und Ausgaben werden auch als Aktionen eines Dienstes bezeichnet. [EHH+08] Im nächsten Schritt werden Ports mit einem Interaktionsmuster definiert, das bestimmt, in welcher Reihenfolge Ein- und Ausgaben eines welchen Typs erfolgen müssen. Im letzten Schritt werden diese Ports an eine Webadresse gebunden, so dass der Dienst aufgerufen werden kann. In WSDL wird lediglich eine syntaktische Beschreibung von Diensten vorgenommen. Die semantische Beschreibung der Funktionalität erfolgt dagegen außerhalb

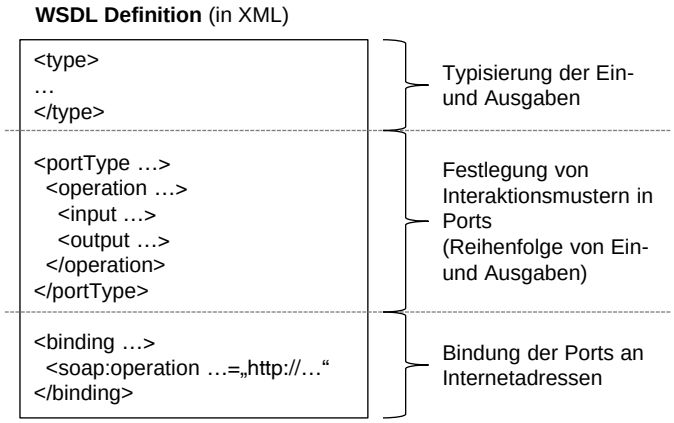

Abbildung 2.4: Schematische Übersicht einer Web Service Definition

von WSDL meist nur informell in Textform. Eine Formalisierung der Semantik auf technischer Ebene ist nur unter zusätzlichem Einsatz von Ontologiesprachen wie OWL-S für Web Services möglich. In diesem Fall wird auch zur Abgrenzung die Bezeichnung Semantic Web Services verwendet (vgl. [CDM+04]). Die Modularität der Dienste wird durch die klare Schnittstellenfokussierung der Web Services unterstützt. In der WSDL Definition wird nur die Schnittstelle zum Anwender beschrieben, Interaktionen mit Dritten stehen nicht im Fokus.

Mit Hilfe von Orchestrierungs- und Choreographiesprachen wie BPEL oder WS-CDL können einzelne Web Services zu zusammenhängenden Abläufen komponiert werden. [PVDH07] In den Sprachen kann die Ablaufreihenfolge der Dienste festgelegt und somit geschäftliche Prozesse auf technischer Ebene abgebildet werden.

Die Suche von Web Services wird unterstützt durch den Verzeichnisdienst UDDI. Dienstanbieter können sich bei einem UDDI Anbieter registrieren und dabei Informationen zu sich selbst, beispielsweise Unternehmensstammdaten, sowie eine Liste der angebotenen Dienste mit funktionalen und nicht funktionalen Eigenschaften hinterlegen. Dienstnehmer können daraufhin UDDI zur Suche nach Dienstanbietern und deren Diensten verwenden. UDDI bietet dazu unterschiedliche Verzeichnisse an. Die weißen Seiten (white pages) ermöglichen Anfragen zum Dienstanbieter vergleichbar mit einem Telefonbuch. Die gelben Seiten (yellow pages) ermöglichen eine Suche nach bestimmten Kategorien von Diensten. Die Kategorisierung basiert dabei auf einer standardisierten Taxonomie. Die grünen Seiten (green pages) schließlich bieten detaillierte technische Informationen zur Einbindung des Dienstes. Häufig wird in diesem Kontext auch auf die WSDL Definition der Schnittstelle verwiesen. [OAS04, Mel08, Erl08, KBS07]

Dienste sind auf technischer Ebene für eine arbeitsteilige Erbringung von Aufgaben ausgelegt, indem sie plattformunabhängig definiert werden und verteilt nutzbar sind. Die arbeitsteilige Erbringung von Diensten wird unterstützt durch eine Dienstplattform, die auch als Enterprise Service Bus (ESB) bezeichnet wird. Diese stellt eine einheitliche Kommunikationsschicht zur

Verfügung, nimmt die technische Integration unterschiedlicher Kommunikationsprotokolle vor und stellt Ortstransparenz für die Dienste her. [KBS07, Mel08, NL05] Web Services nutzen das World Wide Web (WWW) als Kommunikationsmedium und SOAP als XML-basiertes plattformunabhängiges Kommunikationsprotokoll für die Nachrichtenübertragung. [PVDH07]

Zusammenfassung Auf fachlicher Ebene sind Dienste als abgegrenzter geschäftlicher Leistungsumfang, der auf Basis eines Vertrags von einem Dienstanbieter gegenüber einem Dienstnehmer erbracht wird, definiert. Dienste können zu geschäftlichen Abläufen zusammengesetzt werden, die als Prozesse bezeichnet werden. Die fachliche Definition folgt damit prinzipiell den Gestaltungsprinzipien der Modularität, Komposition und arbeitsteiligen Erbringung. Jedoch hat sich noch keine einheitliche präzise Fassung des Dienstbegriffs etabliert und es bestehen Deutungsspielräume insbesondere in der Abgrenzung zu anderen geschäftlichen Handlungskonzepten. Ambiguitäten in der begrifflichen Auslegung erschweren die exakte Abbildung der fachlichen auf die technische Ebene und damit die Erhaltung der Kongruenz zwischen den Ebenen, die aber Voraussetzung für die Realisierung der oben aufgeführten SOA-Vorteile ist.

Auf technischer Ebene materialisiert sich die Diskussion des Dienstbegriffs anhand des weit verbreiteten Web Service Standards. Durch die umsetzungsnahe Diskussion können die Eigenschaften und Gestaltungsmerkmale eines Dienstes im Detail erfasst werden. Jedoch legen Web Services den Dienstbegriff verhältnismäßig eng aus. So wird unter anderem in der Definition auf eine semantische Verhaltensbeschreibung gänzlich verzichtet. Interaktionen über Zeit können nur modelliert werden, wenn das Interaktionsmuster in der WSDL Definition statisch festgelegt wird. Auch bezieht sich die Definition eines Web Service nur auf die Schnittstelle zum Dienstnehmer, Interaktionen mit Dritten werden nicht betrachtet. Auf Grund solcher Limitationen können nur bedingt Rückschlüsse aus der technischen Ebene auf eine fachliche Definition von Diensten in voller Allgemeinheit gezogen werden. Vielmehr ergibt sich ein Bedarf nach einem präzisen fachlichen Modell von Diensten, dass die soeben diskutierten Unschärfen in der begrifflichen Auslegung adressiert.

2.2.4 Anforderungen an ein fachliches Begriffsmodell

In den folgenden Kapiteln werden wir sukzessive ein formales fachliches Begriffsmodell für geschäftliche Handlungen entwickeln, das die präzise Beschreibung und Einordnung von Diensten ermöglicht. Das Modell wird dabei folgende Aspekte adressieren, die wir als Ambiguitäten der fachlichen Auslegung von Geschäftshandlungen in den vorangegangenen Abschnitten identifiziert haben:

- *Formale syntaktische und semantische Beschreibung*
  Das Modell soll eine formale Beschreibung geschäftlicher Abläufe in syntaktischer und semantischer Weise ermöglichen. Dies umfasst sowohl eine formale Erfassung von Geschäftshandlungen (syntaktische Definition), als auch eine formale Verhaltensbeschreibung (Semantik). Wir sehen ein formales syntaktisches und semantisches Verständnis als notwendige Grundlage für die Realisierbarkeit der SOA-Vorteile, zum Beispiel Effizienzgewinne durch Wiederverwendbarkeit.

- *Beschreibung des Zusammenhangs zwischen den Handlungskonzepten*
  Die Diskussion in den letzten Abschnitten hat einen engen Zusammenhang zwischen den unterschiedlichen Konzepten zur Beschreibung geschäftlicher Handlungen gezeigt. Das Modell soll eine präzise Abgrenzung zwischen diesen Konzepten, beispielsweise zwischen Prozess und Dienst, vornehmen und aufzeigen, wie diese ineinander überführt werden können.
- *Erfassung von Abläufen über die Zeit*
  Das Modell soll eine Beschreibung von Abläufen über die Zeit hinweg ermöglichen, um Aktionsfolgen adäquat modellieren zu können. Die Betrachtung von Handlungsabläufen ist Grundlage für eine präzise und vollständige Beschreibung des Verhaltens unter der Annahme, dass Abläufe zustandsbehaftet sind und sich über die Zeit hinweg entwickeln.
- *Differenzierung zwischen Spezifikation und konkretem Ablauf*
  Das Modell soll differenzieren zwischen der Erfassung einzelner konkreter Abläufe und der Spezifikation dieser, die auf eine Menge zulässiger Abläufe zurückgeführt werden kann. Auf diese Weise können Einzelabläufe gegen eine Spezifikation überprüft werden.
- *Differenzierung zwischen Gesamtablauf und Instanzen*
  Das Modell soll die Abbildung mehrere Instanzen unterstützen. Dies beruht auf der Annahme, dass geschäftliche Abläufe in der Regel nicht nur einfach, sondern mehrfach – häufig auch parallel – zur Ausführung kommen. Das Modell soll sowohl die Betrachtung einzelner Instanzen als auch von Instanzmengen, die einen Gesamtablauf repräsentieren, ermöglichen.
- *Fachliche Darstellbarkeit dienstorientierter Architekturen*
  Das Modell soll die Komposition einzelner Geschäftshandlungen zu einem ganzheitlichen fachlichen Architekturmodell unterstützen, das dienstorientiert strukturiert ist. Das fachliche Architekturmodell stellt dabei die Grundlage für eine exakte Übersetzung in die technische Architektur und damit die Implementierung dar.
- *Direkte praktische Anwendbarkeit*
  Das Modell soll dahingehend gestaltet werden, dass reale geschäftliche Abläufe intuitiv im Modell erfasst werden können. Dazu werden wir ein Praxisbeispiel einführen und konsequent alle Bestandteile des Modells anhand dieses Beispiels diskutieren.
- *Technologieunabhängigkeit des fachlichen Modells*
  Die Konzepte des Modells sollen unabhängig von einer bestimmten Implementierungstechnologie beschrieben werden, so dass die Anwendbarkeit nicht eingegrenzt wird.

Wir fokussieren uns in der Modellierung auf eine fachliche Herleitung und Diskussion des Konzepte. Wir werden die Konzepte allerdings in einer Weise definieren, dass sie intuitiv mit der technischen Auslegung in Deckung zu bringen sind und damit eine Übertragung des fachlichen Modells auf die technische Ebene problemlos möglich ist.

3 Begriffsbildung

In diesem Kapitel führen wir anhand eines Praxisbeispiels systematisch Begriffe zur Beschreibung dienstorientierter Architekturen ein und setzen diese zueinander in Beziehung. Auf diese Weise erarbeiten wir ein informelles Begriffsmodell, das wir in UML Notation graphisch darstellen. Wir führen die Begriffsbildung am Beispiel eines repräsentativen Geschäftsablaufs auf fachlicher Ebene durch, um ein intuitives praxisnahes Begriffsverständnis zu ermöglichen. Wir gehen dabei insbesondere auch auf unterschiedliche Sichtweisen von Handlungsabläufen ein, die in der Praxis häufig begrifflich unterschiedlich ausgelegt werden.

In der Diskussion dieses Kapitels stellen wir Konzepte, die in unser Begriffsmodell eingehen und die wir im folgenden Kapitel formalisieren werden, in *kursiver* Schrift dar.

Zum Schluss des Kapitels fassen wir unser intuitives Begriffsverständnis zusammen und spannen damit den Definitionsraum für das formale Begriffsmodell auf. Ziel dieses Kapitels ist dabei, dass jedes zu formalisierende Konzept praktisch eingeordnet und auf diese Weise die formale Darstellung im Modell besser erfasst werden kann.

3.1 Einführung in das Praxisbeispiel

Wir geben zunächst einen groben ersten Überblick zum dem Geschäftsverlauf, anhand dessen wir die Begriffsbildung durchführen. Im Laufe des Kapitels werden wir den Ablauf zunehmend detaillieren. Zudem werden wir auch alle formalen Definitionen anhand dieses Praxisbeispiels illustrieren.

Wir untersuchen den Vorgang einer Buchungsanfrage durch Privatkunden direkt auf der Webseite einer Fluggesellschaft, die wir Holiday Airways nennen. Wir gehen davon aus, dass Holiday Airways eine Vielzahl von Kunden bedient, diskutieren exemplarisch aber nur zwei davon, die wir als Herrn Müller und Frau Huber bezeichnen.

Abbildung 3.1 zeigt in der Notation eines UML Aktivitätsdiagramms (vgl. [Obj05a, Obj05b]) den Ablauf der Buchungsanfrage im Überblick. Dieser gestaltet sich wie folgt: Ein Kunde besucht die Webseite von Holiday Airways und stellt eine Buchungsanfrage, die sich aus dem gewünschten Abflug- und Zielflughafen sowie der gewünschten Flugzeit zusammensetzt. Wir gehen davon aus, dass sich Buchungsanfragen immer nur auf eine Person beziehen. Die Fluggesellschaft bearbeitet die Anfrage eines Kunden in vier Schritten. Sie ermittelt zunächst unabhängig voneinander die nächste verfügbare Flugverbindung (ha_1) und in zwei sequentiellen Schritten den Preis des Fluges (ha_2, ha_3). Nach Abschluss dieser drei Schritte fasst sie den Zeitpunkt und den Preis der ermittelten Flugverbindung zu einem Angebot zusammen und schickt dieses an den Kunden (ha_4).

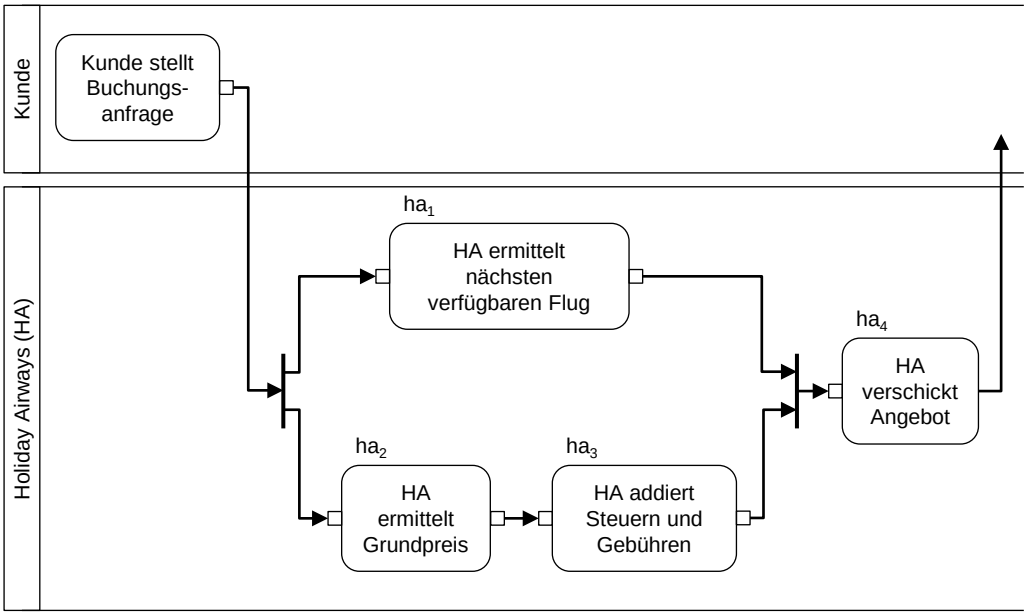

Abbildung 3.1: Praxisbeispiel: Buchungsanfrage bei einer Fluggesellschaft, dargestellt als UML Aktivitätsdiagramm

Bei der Ermittlung der Flugverbindung gehen wir vereinfachend davon aus, dass Holiday Airways nur die nächste verfügbare Verbindung nach dem durch den Kunden angefragten Zeitpunkt ermittelt. Holiday Airways garantiert seinen Kunden die Verfügbarkeit einer angebotenen Verbindung für 5 Minuten und reserviert für diesen Zeitraum einen Sitzplatz. Die Buchung des Fluges muss innerhalb dieser Zeit erfolgen, sonst wird der Platz wieder freigegeben. Entsprechend ergibt sich die Verfügbarkeit eines Fluges aus der aktuellen Buchungslage sowie anderen Kundenanfragen der letzten 5 Minuten für die gleiche Verbindung.

Die Preisberechnung erfolgt parallel zur Ermittlung der nächsten Verbindung. Wir unterstellen dabei simplifizierend, dass der Preis eines Fluges nicht von der Buchungssituation abhängt. Holiday Airways berechnet zunächst den Grundpreis für die angefragte Flugverbindung. Dieser setzt sich zusammen aus einer Entfernungskomponente und zwei möglichen Aufschlägen für Verbindungen in der Hauptzeit und zu besonderen Destinationen. In einem zweiten Schritt addiert Holiday Airways zum ermittelten Grundpreis noch die Flughafensteuern und Gebühren für die Abfertigung.

Die vorgestellte Buchungsanfrage stellt selbstverständlich eine stark vereinfachte Variante im Vergleich zu einer normalen Flugbuchung über das Internet dar. Zusätzlich zu den bereits erwähnten simplifizierenden Annahmen können in unserem Beispiel keine Präferenzen bezüglich Buchungsklasse, Sitzplatz, Mahlzeiten und ähnlichem gestellt werden. Wir wählen aber bewusst diese vereinfachte Darstellung, um die Diskussion des Beispiels – insbesondere auch im formalen Teil – anschaulicher gestalten zu können und nehmen nur solche Aspekte mit auf, die zur

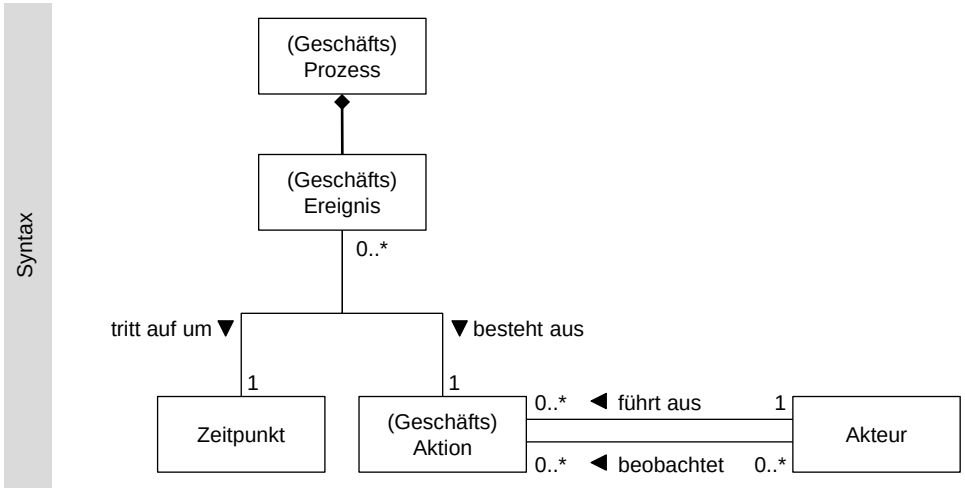

Abbildung 3.2: Grundkonzepte zur syntaktischen Ablaufbeschreibung, strukturiert anhand eines UML Klassendiagramms

Klärung der Begriffe erforderlich sind.

3.2 Syntaktische Ablaufbeschreibung

Für die syntaktische Beschreibung des Ablaufs einer Buchungsanfrage führen wir eine Reihe von Konzepten ein, mit denen wir sowie die beteiligten Personen und Systeme als auch deren Handlungen präzise darstellen können. Wir werden uns dazu im Folgenden mit den Begriffen Akteur, Aktion, Ereignis, Zeitpunkt und Prozess befassen. Abbildung 3.2 zeigt in der Notation eines UML Klassendiagramms (vgl. [Obj05a, Obj05b]) den strukturellen Zusammenhang zwischen diesen Begriffen im Überblick.

Akteur An der beschriebenen Szene sind Personen, namentlich die beiden Kunden Herr Müller und Frau Huber, sowie Rechnersysteme der Fluggesellschaft Holiday Airways beteiligt. Wir bezeichnen jede dieser in einem geschäftlichen Ablauf vorkommenden Entitäten als einen *Akteur*.

Ein Akteur ist im einfachsten Fall eine natürliche Person, wie im Beispiel die beiden Kunden Herr Müller oder Frau Huber. Neben einer natürlichen Person sehen wir aber auch IT-Systeme als Akteure an. Im Fall von Holiday Airways liegt es nahe, dass die Anfrage eines Kunden nicht durch eine Person, sondern durch ein Rechnersystem bearbeitet wird. Wir unterschieden daher zwischen Personen als manuellen und Systemen als automatisierten Akteuren.

Eine Organisation wie Holiday Airways kann auf unterschiedlichen Abstraktionsebenen betrachtet werden. Es ist denkbar, Holiday Airways als die Menge aller Mitarbeiter und IT-Systeme

zu erfassen, die unter dem Firmennamen von Holiday Airways fungieren. Ebenso ist es möglich, das ganze Unternehmen als einen abstrakten Akteur zu betrachten, der alle Handlungen des Unternehmens repräsentiert.

Schließlich sind beliebige Mittelwege aus den beiden dargestellten Optionen realisierbar. So können beispielsweise einzelne Organisationseinheiten der Fluggesellschaft als Akteure angesehen werden. Wir werden in der weiteren Diskussion des Beispiels von drei automatisierten Akteuren bei Holiday Airways ausgehen. Es handelt sich dabei um drei Rechnersysteme, die eine Kundenanfrage bearbeiten. Das erste System ermittelt als Buchungssystem die nächste Flugverbindung, das zweite System führt die Preisberechnung durch und das dritte System ist verantwortlich für die Angebotserstellung.

Aktion Akteure können an geschäftlichen Handlungen beteiligt sein, die wir als *Aktionen* bezeichnen. Wir unterscheiden dabei zwischen einer aktiven und passiven Beteiligung. Im ersten Fall führt ein Akteur eine Aktion selbst durch, im zweiten Fall ist die Handlung eines anderen Akteurs für einen Akteur sichtbar beziehungsweise betrifft diesen. Unsere exemplarische Buchungsanfrage enthält, wie in Abbildung 3.1 dargestellt, fünf Aktionen:

(0) Der Kunde (aktiv) übermittelt eine Anfrage an die Fluggesellschaft (passiv).

(1) Das Buchungssystem (aktiv) ermittelt die nächste Flugverbindung und gibt das Ergebnis an das System zur Angebotserstellung (passiv) aus.

(2) Das System zur Preisberechnung (aktiv) ermittelt den Grundpreis.

(3) Das System zur Preisberechnung (aktiv) ermittelt die Steuern und Gebühren und gibt das Ergebnis an das System zur Angebotserstellung (passiv) aus.

(4) Das System zur Angebotserstellung (aktiv) stellt ein Angebot zusammen und übermittelt dieses an den Kunden (passiv).

Somit geht die Aktion (0) vom Kunden und die Aktionen (1) bis (4) von Rechnersystemen von Holiday Airways aus. Für Aktion (0) gehen wir davon aus, dass die Anfrage dort die beiden Systeme zur Ermittlung der Flugverbindung und zur Preisberechnung erreicht. Die Aktion (2) ist für keinen anderen Akteur sichtbar.

Alle fünf Aktionen bestehen in der Verarbeitung und/oder Übertragung von Daten. Dies ist der Tatsache geschuldet, dass wir Handlungen betrachten, die in Interaktion mit automatisierten Akteuren erfolgen. Wir fokussieren uns in der Beschreibung von Handlungen auf die Datenverarbeitung durch Rechnersysteme, da diese im Kontext betrieblicher Informationssysteme von hervorgehobenem Interesse sind. In voller Allgemeinheit kann unter einer Aktion jedoch jede Form einer Tätigkeit, die durch eine Person durchgeführt wird, verstanden werden.

Ereignis Eine geschäftliche Handlung kann mehrfach durchgeführt werden. So ist in unserem Beispiel davon auszugehen, dass Holiday Airways eine Vielzahl von Kundenanfragen bearbeitet und damit die Aktionen (1) bis (4) immer wieder durchführt, für jede Kundenanfrage einmal. Wir differenzieren zwischen der Beschreibung einer Handlung im Allgemeinen als Aktion und dem konkreten Auftreten einer Aktion in einer bestimmten Situation, das wir als *Ereignis* bezeichnen.

Jedem Ereignis kann eine Aktion zugewiesen werden, die in dem Ereignis zur Ausführung kommt. Im Zuge der Bearbeitung der Kundenanfrage von Herrn Müller treten fünf Ereignisse auf:

- Herr Müller schickt eine Buchungsanfrage für einen Flug von München nach Frankfurt mit gewünschter Flugzeit am 14. Januar um 12 Uhr. (Aktion 0)
- Das Buchungssystem ermittelt die nächste Verbindung am 14. Januar um 12:15 Uhr. (Aktion 1)
- Das Preissystem berechnet den Grundpreis mit 170 EUR. (Aktion 2)
- Das Preissystem berechnet den Gesamtpreis mit 221 EUR. (Aktion 3)
- Das System zur Angebotserstellung übermittelt ein Angebot für den 14. Januar um 12:15 Uhr zum Preis von 221 EUR. (Aktion 4)

Wir gehen zunächst davon aus, dass Frau Huber exakt die gleiche Verbindung anfragt und auch die gleiche Verbindung von Holiday Airways angeboten bekommt.

Jedes Ereignis kann zeitlich eingeordnet werden. So stellt Herr Müller seine Buchungsanfrage beispielsweise zu einem bestimmten Zeitpunkt, vermutlich einige Tage oder Wochen vor der gewünschten Flugzeit. Die im Ereignis durchgeführte Aktion kann sich über einen Zeitraum erstrecken, so dass dem Ereignis ein Start- und ein Endzeitpunkt zugeordnet werden kann. Liegt der Endzeitpunkt eines Ereignisses vor dem Startzeitpunkt eines zweiten Ereignisses, sprechen wir von sequentiellen Ereignissen. Kommt es zu zeitlichen Überschneidungen, betrachten wir die Ereignisse als parallel.

In der weiteren Diskussion unseres Begriffsmodells werden wir simplifizierend davon ausgehen, dass Handlungen zu einem bestimmten *diskreten Zeitpunkt* erfolgen und gehen entsprechend von einem diskreten Zeitmodell aus. Auch in der diskreten Zeitsicht können parallele und sequentielle Ereignisse modelliert werden, indem sie zum gleichen beziehungsweise zu unterschiedlichen diskreten Zeitpunkten stattfinden.

Prozess Wir haben bis jetzt die aufgetretenen Ereignisse in den Buchungsanfragen der beiden Kunden nur einzeln betrachtet. Für die Beschreibung von Abläufen fassen wir nun Ereignisse in Ereignismengen zusammen, die wir als einen *Prozess* bezeichnen. So können wir die soeben diskutierten fünf Ereignisse der Anfrage von Herrn Müller als einen Prozess betrachten. In einem Prozess kann prinzipiell jede beliebige Menge von Ereignissen zusammengefasst werden, ohne dass die Ereignisse zueinander in Beziehung stehen müssen. Allerdings lässt die bisherige Diskussion unseres exemplarischen Buchungsprozesses bereits vermuten, dass in diesem Abhängigkeiten zwischen den Ereignissen bestehen. Dies trifft auf viele Ablaufbeschreibungen in Unternehmen zu. Wir werden im folgenden Abschnitt diskutieren, wie solche Abhängigkeiten in systematischer Weise erfasst werden können.

Definitionsgemäß können auch mehrere Buchungsanfragen, zum Beispiel die Anfragen von Herrn Müller und Frau Huber, in einem Prozess zusammengefasst werden. Dieser Prozess enthält zehn Ereignisse, für jeden Kunden fünf. Schließlich lassen sich als drittes Beispiel auch die beiden Kundenanfragen als ein Prozess mit nur zwei Ereignissen betrachten.

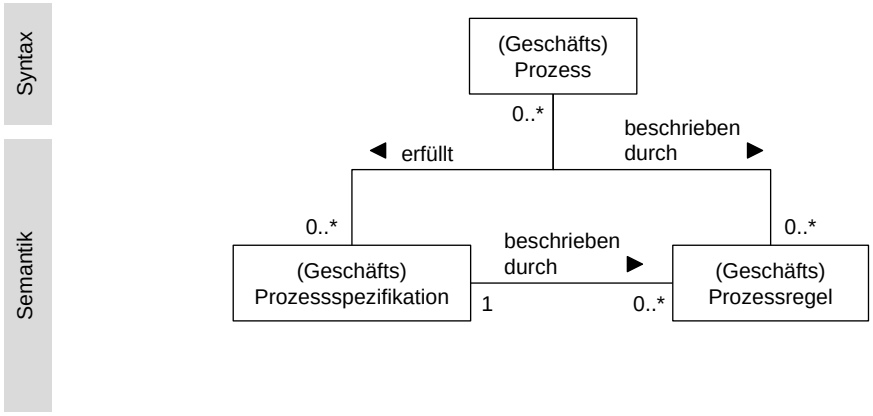

Abbildung 3.3: Grundkonzepte zur semantischen Ablaufbeschreibung, strukturiert anhand eines UML Klassendiagramms

Die drei vorgestellten Prozesse bezeichnen wir als endliche Prozesse, da sie durch eine endliche Menge von Ereignissen repräsentiert werden können. Analog sind auch unendliche Prozesse modellierbar. Dies wäre beispielsweise der Fall, wenn wir die Ereignisse aller Buchungsanfragen von Holiday Airways in einem Gesamtprozess zusammenfassen und davon ausgehen, dass die Fluggesellschaft kontinuierlich Anfragen erhält. Jede Teilmenge eines Prozesses stellt definitionsgemäß auch wieder einen Prozess dar. Zur Abgrenzung bezeichnen wir solche Teilmengen auch als *Teilprozesse*.

3.3 Semantische Ablaufbeschreibung

Die Diskussion der drei Prozessbeispiele hat gezeigt, dass Ereignisse in Prozessen in Abhängigkeit zueinander stehen können. Wir werden nun solche Abhängigkeiten systematisch erfassen, indem wir Mengen von Prozessen betrachten, in denen bestimmte Bedingungen für Ereignisse erfüllt sind. Zudem führen wir mit der Prozessregel ein Konzept ein, wie solche Mengen systematisch hergeleitet werden können.

Prozessspezifikation Während wir bisher nur einzelne Buchungsanfragen bei Holiday Airways untersucht haben, befassen wir uns nun mit der zugrundeliegenden Logik, nach der die Fluggesellschaft Kundenanfragen bearbeitet. Wir beschreiben das Verhalten von Holiday Airways, indem wir eine Menge aller zulässigen Bearbeitungsabläufe einer Kundenanfrage angeben. Jeder Prozess, der in dieser Menge enthalten ist, stellt ein gültiges Verhalten seitens der Fluggesellschaft dar. Entsprechend betrachten wir alle nicht enthaltenen Prozesse als ungültiges Verhalten. Mit dem Aufstellen einer solchen Prozessmenge treffen wir eine semantische Aussage zur Beschaffenheit von Prozessen. Auf diese Weise können Verhaltensaspekte der an den Prozessen beteiligten Akteure modelliert werden. Wir bezeichnen eine derartige Prozessmenge als *Spezifikation* eines Prozesses.

Die semantische Betrachtung von Prozessen ermöglicht es, in systematischer Weise Abhängigkeiten zwischen den Ereignissen beziehungsweise den Aktionen, die in den Ereignissen zur Ausführung kommen, zu erfassen. Wir sprechen von einer Abhängigkeit, wenn das Auftreten eines Ereignisses auf das Auftreten von einem oder mehreren anderen Ereignissen zurückgeführt werden kann. Wenn den Ereignissen ein temporales Modell zugrundeliegt, also jedem Ereignis ein Auftrittszeitpunkt zugeordnet werden kann, fassen wir eine solche Abhängigkeit zwischen zwei oder mehreren Ereignissen als *Kausalität* auf. Eine Kausalität stellt einen Ursache-/Wirkungszusammenhang zwischen Ereignissen beziehungsweise den damit verknüpften Aktionen dar. Dabei wird vorausgesetzt, dass die Ursache zeitlich vor der Wirkung auftritt. [Hum00]

Wir unterscheiden zwei Arten der Kausalität: die notwendige und die hinreichende Voraussetzung. Ein Ereignis ist für ein anderes zeitlich nachgelagertes Ereignis notwendige Voraussetzung, wenn das nachgelagerte Ereignis als Wirkung nur dann eintritt, wenn das erste Ereignis als Ursache auftritt. Ein Ereignis ist hinreichende Voraussetzung für ein anderes Ereignis, wenn das erste als Ursache dazu führt, dass das zweite Ereignis als Wirkung auf jeden Fall zu einem späteren Zeitpunkt auftreten wird. Wir fokussieren uns in der folgenden Diskussion insbesondere auf die erste Form der Kausalität, indem wir untersuchen, welche Ereignisse notwendige Voraussetzung für andere Ereignisse sind.

Kausalitäten zwischen Ereignissen können sowohl extensional als auch intensional erfasst werden. [Bro10] Betrachten wir eine Spezifikation als Menge aller zulässigen Prozesse, so kann eine Kausalität zwischen Ereignissen extensional abgeleitet werden, wenn diese auf alle Prozesse in der Spezifikation zutrifft. Aus der Betrachtung eines einzelnen Prozesses dagegen können im Allgemeinen keine Kausalitäten zwischen Ereignissen festgestellt werden. Dies ist nur dann möglich, wenn zusätzliche Kenntnisse zum Ablaufmuster eines Prozesses vorhanden sind, beispielsweise auf Basis eines formalen Automatenmodells. Wir sprechen in diesem Fall von intensionalen Kausalitäten, die direkt aus dem Ablaufmuster abgeleitet werden können. In unserem Beispiel lässt sich eine solche intensionale Kausalität aus der Beschreibung der Anfragebearbeitung folgern. So stellt beispielshalber die Übermittlung eine Anfrage durch einen Kunden eine kausale Ursache für die Durchführung der Preisberechnung seitens Holiday Airways dar.

Wir gehen für die weitere Diskussion des Beispiels davon aus, dass die im letzten Abschnitt vorgestellten Buchungsanfragen von Herrn Müller und Frau Huber als zwei Prozesse mit jeweils fünf Ereignissen in der Spezifikation der Anfragebearbeitung enthalten sind. Die Spezifikation beinhaltet in analoger Weise auch alle anderen zulässigen Anfragebearbeitungen für beliebige Kunden, die beliebige Anfragen an die Fluggesellschaft stellen können. Zudem enthält sie auch Prozesse, die Vereinigungen mehrerer Anfragebearbeitungen darstellen. Dies ist erforderlich, damit wir Abhängigkeiten zwischen den einzelnen Buchungsanfragen erfassen können. Einen Prozess, in dem die Fluggesellschaft eine Verbindung für eine abweichende Strecke anbietet, würden wir dagegen als unzulässiges Verhalten betrachten und wäre damit nicht in der Spezifikation enthalten.

Eine Spezifikation, wie wir sie gerade skizziert haben, enthält eine Vielzahl von zulässigen Prozessen. Es stellt sich die Frage, wie eine solche Spezifikation in systematischer Weise aufgestellt werden kann. Dazu können sowohl informelle, als auch formale Beschreibungsansätze herangezogen werden. Wir werden unsere folgenden Überlegungen in erster Linie auf Prozessregeln

aufbauen. Mit Regeln entwickeln wir eine Spezifikation vergleichbar mit der Beschreibung einer formalen Sprache, die auf Basis von Produktionsregeln hergeleitet wird.

Prozessregel Mit *Prozessregeln* beschreiben wir Zusammenhänge im Ablauf eines Prozesses, indem wir aus bereits aufgetretenen Aktionen die Durchführung weiterer Aktionen ableiten. Die Regeln richten sich dabei an den Kausalitäten eines Prozesses aus. Sie formulieren Aktionen, die als Wirkungen aus vorgelagerten Aktionen resultieren. So kann eine Regel beispielsweise aus Sicht eines bestimmten Akteurs aufgestellt werden und das Verhalten dieses Akteurs beschreiben, indem durch den Akteur durchzuführende Aktionen in Abhängigkeit von Aktionen festgelegt werden, die für den Akteur beobachtbar sind und dessen Verhalten beeinflussen.

Formal betrachten wir eine Prozessregel als eine Relation zwischen zwei Ablaufstadien eines Prozesses. Das erste Stadium enthält eine Teilmenge der Ereignisse des zweiten Stadiums. Der erste Prozess stellt also einen Teilprozess des zweiten dar. Eine Prozessregel beschreibt die Fortsetzung eines gegebenen Prozesses, indem sie auf Basis der bereits aufgetretenen Aktionen und damit verbundenen Ereignisse festlegt, welche Aktionen aus diesen folgen können oder gar müssen. Wir bezeichnen dies als Entwicklung eines Prozesses. Eine Regel entwickelt einen gegebenen Prozess, indem sie die Ereigniskonstellation in diesem analysiert und dann Aussagen trifft, welche Aktionen sich im Prozess daraus ergeben. Für unser Beispiel der Buchungsanfrage stellen wir folgende vier Regeln auf, die mit den in Abbildung 3.1 eingeführten vier Schritten von Holiday Airways (HA) korrelieren:

- *HA ermittelt nächsten verfügbaren Flug (ha_1)*
  Diese Regel beschreibt die Ermittlung der nächsten verfügbaren Flugverbindung für eine gegebene Kundenanfrage. Dazu reserviert HA zunächst auf Basis der aktuellen Buchungslage Sitzplätze für alle Kundenanfragen der letzten fünf Minuten, die sich auf die gleiche Strecke beziehen, und gibt dann die nächste verfügbare Flugverbindung aus. Damit wird sichergestellt, dass ein angefragter Platz für fünf Minuten dem jeweiligen Kunden gegenüber garantiert werden kann.

- *HA ermittelt Grundpreis (ha_2)*
  Diese Regel beschreibt die Ermittlung des Grundpreises auf Basis einer gegebenen Kundenanfrage. Der Grundpreis setzt sich aus drei Komponenten zusammen. HA berechnet zunächst einen Entfernungspreis anhand einer Entfernungstabelle und eines vorgegebenen Preis-pro-Meile-Faktors. Bezieht sich die Kundenanfrage auf einen Wochentag, so wird ein Aufschlag in Höhe von 20% addiert. Zudem berechnet die Fluggesellschaft Zuschläge für bestimmte beliebte Flugziele auf Basis einer vorgegebenen Tabelle. Der Grundpreis wird schließlich als Summe des Entfernungspreises und der möglichen Aufschläge für Wochentage und die Destination ausgegeben.

- *HA addiert Steuern und Gebühren (ha_3)*
  Diese Regel beschreibt die Ermittlung des Gesamtpreises auf Basis eines bereits berechneten Grundpreises. Holiday Airways bestimmt den Gesamtpreis in pauschaler Weise, indem sie 30% des Grundpreises als Aufschlag für Steuern und Flughafengebühren addiert.

- *HA verschickt Angebot (ha_4)*
  Diese Regel beschreibt die Angebotserstellung. Holiday Airways fasst, sobald für eine Kundenanfrage die nächste verfügbare Flugverbindung und der Gesamtpreis ermittelt wurde, Flugzeit und Preis zu einem Angebot zusammen und schickt dieses an den Kunden, von dem sie die Anfrage erhalten hat.

Regeln können in unterschiedlicher Granularität definiert werden. So kann eine Regel einen einfachen Zusammenhang zwischen zwei einzelnen Ereignissen erfassen, kann aber auch eine umfassende Aussage treffen, wie ein Prozess mit einer Vielzahl zu beobachtender Ereignisse weiterzuentwickeln ist. Dabei kann es zweckmäßig sein, komplexere Regeln durch Zusammensetzung einfacherer Regeln herzuleiten. Wir können beispielsweise das Verhalten von Holiday Airways zur Bearbeitung einer Kundenanfrage beschreiben als Zusammensetzung der soeben eingeführten vier Prozessregeln. Wir bezeichnen eine solche Zusammensetzung von Regeln als *Komposition*. Dabei unterscheiden wir mehrere Arten der Komposition. Die Regeln zur Berechnung des Grund- und des Gesamtpreises wenden wir nacheinander an, wir sprechen hierbei von einer sequentiellen Komposition. Die Regeln zur Ermittlung des nächsten Fluges und zur Preisberechnung können gleichzeitig angewendet werden, wir bezeichnen dies auch als parallele Komposition.

Durch sequentielle Komposition, die einer Relationenkomposition entspricht, können Prozessregeln iterativ eingesetzt werden. Dies ist sinnvoll, wenn sich aus der Anwendung einer Regel neue Ereignisse in den Prozessen ergeben, auf die wiederum die Regel angewendet werden kann. Auf diese Weise kann eine Spezifikation sukzessive weiterentwickelt werden. Wir sprechen von der *transitiven Hülle* einer Prozessregel, wenn diese so lange iterativ angewendet wird, bis ein *Fixpunkt* erreicht ist und sich die Spezifikation durch Regelanwendung nicht mehr weiter entwickeln lässt. Auf diese Weise kann ein Verhalten ausgehend von einer einfachen Basisspezifikation, die beispielshalber die Menge aller möglichen Kundenanfragen darstellt, durch Hüllenbildung einer Prozessregel beschrieben werden.

Dem intuitiven Verständnis folgend fordern wir, dass eine Regel bereits aufgetretene Ereignisse nicht modifizieren darf und folglich ausschließlich Aussagen bezüglich weiterhin durchzuführender Aktionen trifft. Auf diese Weise kann die regelbasierte Entwicklung von Prozessen entlang einer vollständigen Halbordnung, der Teilprozessordnung, ausgerichtet und somit die Existenz eines Fixpunktes in der Hüllenbildung einer Prozessregel nachgewiesen werden.

In der Anwendung einer Regel gehen wir grundsätzlich von einem vorhandenen Prozess aus, der anhand der Regellogik fortgesetzt wird. Wir bezeichnen den bereits durchgeführten Prozess auch als *Prozessannahme*, da dessen Ausführung nicht durch die Regel festgelegt, sondern nur erfasst werden kann. In unserem Beispiel gehen wir davon aus, dass Holiday Airways keine genaueren Kenntnisse bezüglich des Kundenverhaltens besitzt und daher eine Annahme treffen muss, welche Anfragen seitens der Kunden zu erwarten sind. Durch iterative Anwendung der Prozessregeln von Holiday Airways auf Basis einer solchen Prozessannahme kann eine Spezifikation abgeleitet werden, die alle zulässigen Anfragebearbeitungen durch die Fluggesellschaft repräsentiert.

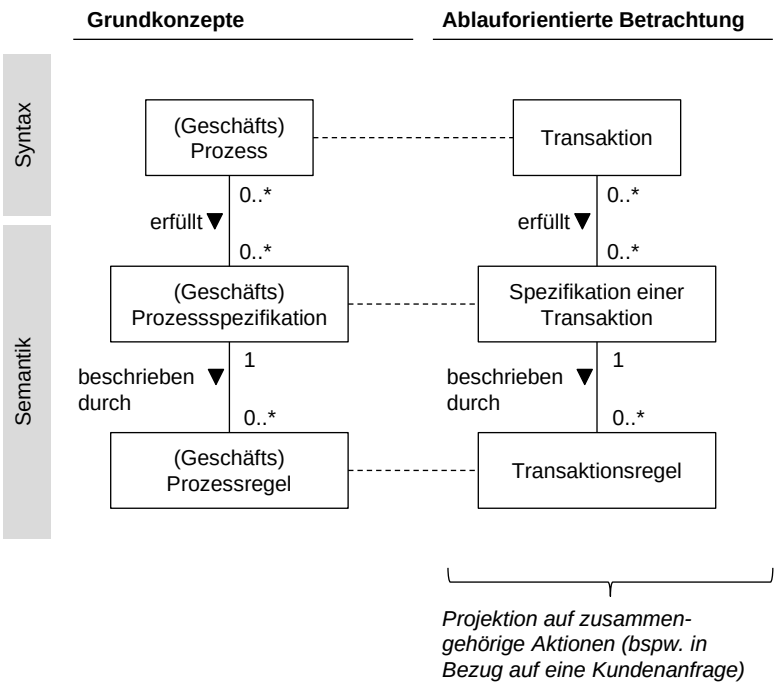

Abbildung 3.4: Konzepte zur ablauforientierten Vorgangsbeschreibung, strukturiert anhand eines UML Klassendiagramms

3.4 Strukturierung geschäftlicher Abläufe

Wir haben bis jetzt das Beispiel der Flugbuchung in voller Allgemeinheit in syntaktischer und semantischer Weise beschrieben. Um eine differenziertere Perspektive auf die Erfassung geschäftlicher Abläufe zu erhalten, befassen wir uns im Folgenden mit unterschiedlichen Strukturierungen geschäftlicher Handlungen.

3.4.1 Ablauforientierte Beschreibung

Es ist davon auszugehen, dass Holiday Airways eine Vielzahl von Kundenanfragen bearbeitet, die sowohl sequentiell als auch parallel auftreten können. Im Beispiel haben wir bereits zwei Kunden eingeführt. Für eine differenzierte Beschreibung der Geschäftstätigkeit von Holiday Airways möchten wir sowohl einzelne Kundenanfragen als auch die Gesamtheit aller Anfragen erfassen. Dabei müssen auch in der Betrachtung mehrerer Abläufe die einzelnen Anfragen präzise voneinander abgrenzbar sein, so dass beispielsweise eine Preisberechnung der Fluggesellschaft eindeutig einer Kundenanfrage zugeordnet werden kann.

Wir gehen davon aus, dass die zuvor aufgestellten Prozessregeln nicht nur für einzelne Kundenanfragen, sondern auch für eine Menge von Kundenanfragen anwendbar sind. In diesem Fall werden die Regeln auf jede einzelne Anfrage in gleicher Weise angewendet. Entsprechend kann

aus den Regeln eine Spezifikation abgeleitet werden, die Prozesse mit einer Vielzahl bearbeiteter Kundenanfragen enthält. Solche Prozesse bezeichnen wir als *Transaktionsfluss*. Ein Transaktionsfluss setzt sich aus mehreren Transaktionen zusammen. Eine *Transaktion* fassen wir als einen Prozess mit zusammengehörigen Aktionen auf. Die Auslegung der Zusammengehörigkeit hängt dabei vom Kontext ab und kann in unterschiedlicher Weise erfolgen. Wir modellieren Transaktionen anhand von *Instanzen*. Wir gehen davon aus, dass die Fluggesellschaft jede Anfrage in einer eindeutig referenzierbaren Instanz bearbeitet und auf diese Weise jede Aktion während der Bearbeitung auf einen Kunden bezogen werden kann. Folglich enthält eine Transaktion in unserer Auslegung nur Aktionen, die dahingehend als zusammengehörig betrachtet werden, dass sie sich der gleicher Kundenanfrage zuordnen lassen. In unserem Beispiel untersuchen wir zwei exemplarische Transaktionen für die Bearbeitung der beiden Kundenanfragen.

Aus einem Transaktionsfluss können durch Projektion einzelne Transaktionen abgeleitet werden. Da Transaktionen ebenfalls Prozesse darstellen, kann auch für jede Transaktion eine Spezifikation formuliert werden. Unserem Beispiel folgend würde eine solche Spezifikation für einen bestimmten Kunden die Menge aller zulässigen Anfragebearbeitungen für diesen Kunden umfassen. Eine *Spezifikation einer Transaktion* kann anhand von Prozessregeln hergeleitet werden, die wir auch als *Transaktionsregeln* bezeichnen. Transaktionsregeln lassen sich ebenfalls durch Projektion aus Prozessregeln, die einen gesamten Transaktionsfluss beschreiben, ableiten. Da es in voller Allgemeinheit zu Wechselwirkungen zwischen den Transaktionen kommen kann, müssen während der Projektion einer Regel auf eine Transaktion Annahmen bezüglich der anderen Transaktionen getroffen werden.

Betrachten wir dies anhand unseres Praxisbeispiels. Wir fassen die Buchungsanfrage von Frau Huber als eine Transaktion auf und formulieren eine Transaktionsregel, die die Bearbeitung der Anfrage von Frau Huber beschreibt. Die Transaktionsregel bezieht sich im Kontrast zu den vorhin diskutierten Prozessregeln ausschließlich auf diese eine spezifische Kundenanfrage. Wir wissen, dass die nächste verfügbare Flugverbindung in Abhängigkeit aller Kundenanfragen der letzten fünf Minuten ermittelt wird. Da diese jedoch außerhalb des Betrachtungsraums der Transaktionsregel liegen, muss eine Annahme bezüglich der Existenz weiterer Anfragen getroffen werden.

Wir gehen für den Moment davon aus, dass auf dem Flug von München nach Frankfurt am 14. Januar um 12:15 Uhr nur noch ein Platz verfügbar ist und der nächste Flug um 13:15 Uhr angeboten werden kann. Für diesen Flug nehmen wir simplifizierend unbegrenzte Kapazität an. Zudem gehen wir davon aus, dass neben der Anfrage von Frau Huber beliebige weitere Anfragen existieren können. Dies führt zu einer nicht deterministisch definierten Transaktionsregel, die für die Anfrage von Frau Huber sowohl ein Flugangebot um 12:15 Uhr, als auch um 13:15 Uhr ableiten kann. Aus Sicht der Transaktionsregel kann nicht entschieden werden, welche der beiden Verbindungen die nächste verfügbare Verbindung ist. Dies liegt darin begründet, dass andere Kundenanfragen der letzten fünf Minuten, beispielsweise die von Herrn Müller, nicht im Betrachtungsraum der Regel liegen und damit die Reservierungssituation nicht vollständig rekonstruiert werden kann.

Sofern sich Transaktionen nicht gegenseitig beeinflussen, bezeichnen wir diese als *unabhängige Transaktionen*. In diesem vereinfachten Fall können Transaktionsregeln per Projektion abgeleitet

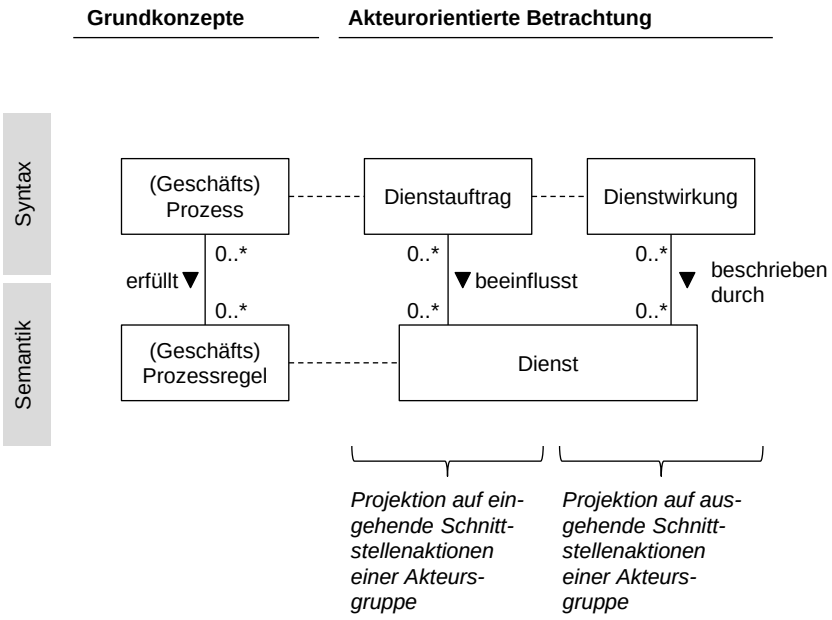

Abbildung 3.5: Konzepte zur akteurorientierten Vorgangsbeschreibung, strukturiert anhand eines UML Klassendiagramms

werden, ohne dass dabei Annahmen bezüglich anderer Transaktionen getroffen werden müssen.

3.4.2 Akteurorientierte Beschreibung

Die Aktionen eines Prozesses werden durch Akteure ausgeführt. Wir haben mit unserem Beispiel bereits eine Praxissituation diskutiert, in der mehrere Akteure an einem geschäftlichen Vorgang beteiligt sind: die Kunden Herr Müller und Frau Huber sowie drei Rechnersysteme der Fluggesellschaft Holiday Airways. In einer solchen arbeitsteiligen Konstellation ist insbesondere auch von Interesse, welcher Akteur welchen Beitrag zu einem Prozess leistet. Wir untersuchen daher eine weitere Betrachtungsweise geschäftlicher Abläufe, die sich an Akteuren oder Gruppen von Akteuren orientiert.

Um den Beitrag, den eine Gruppe von Akteuren in einem Prozess erbringt, präzise zu erfassen, untersuchen wir Interaktionen zwischen den Akteuren der Gruppe und anderen am Prozess beteiligten Personen und Systemen. Unter einer *Gruppe von Akteuren* verstehen wir dabei eine zusammengehörige Menge von Akteuren. In unserem Beispiel betrachten wir die drei Rechnersysteme von Holiday Airways als eine Gruppe, die zusammen die Anfragen der beiden Kunden Herr Müller und Frau Huber bearbeiten. Wir unterscheiden dabei zwischen *internen Aktionen* und *Schnittstellenaktionen* einer Akteursgruppe. Interne Aktionen sind nur innerhalb der Gruppe beobachtbar. Dies sind alle Aktionen während der Buchungsanfrage, an denen die Kunden nicht

beteiligt sind. Schnittstellenaktionen sind Aktionen, die entweder durch die Gruppe durchgeführt oder von dieser beobachtet werden, an denen aber auch mindestens ein Akteur außerhalb der Gruppe beteiligt ist. Im Beispiel ist dies einerseits die Kundenanfrage, andererseits der Versand des Flugangebots an den Kunden.

Ein *Dienst* beschreibt das externe Verhalten einer Gruppe von Akteuren, indem er die durch diese Gruppe zu erbringenden Schnittstellenaktionen in Abhängigkeit von beobachteten Schnittstellenaktionen definiert. Wir bezeichnen dabei die beobachteten Schnittstellenaktionen anderer Akteure auch als *Dienstauftrag* und die durch die Akteursgruppe selbst durchgeführten Schnittstellenaktionen als *Wirkung des Dienstes*. Mit Diensten erfassen wir den Beitrag einer Akteursgruppe zu einem Prozess, indem wir basierend auf einem von anderen Akteuren erhaltenen Dienstauftrag die durch die Gruppe durchzuführenden Handlungen beschreiben und diese als Wirkungen des Dienstes auffassen.

So ist beispielsweise der Dienst, den Holiday Airways – vertreten durch die Rechnersysteme – gegenüber Herrn Müller erbringt, der Versand eines Angebots für den Flug von München nach Frankfurt am 14. Januar zum Preis von 221 EUR als Reaktion auf die entsprechende Anfrage von Herrn Müller. Der Dienst abstrahiert von allen internen Aktionen, beispielsweise der Preisberechnung, die zur Erbringung des Dienstes erforderlich sind. Dienste wenden eine strikte Außensicht im Bezug auf eine Akteursgruppe an, die wir auch als *Black-Box-Perspektive* bezeichnen.

Eine solche Dienstdefinition für Holiday Airways kann aus der Komposition der vorhin aufgestellten Prozessregeln für die Fluggesellschaft abgeleitet werden, indem diese auf die Schnittstellenaktionen projiziert wird. Der Dienst kann damit als eine Art Leistungsvereinbarung zwischen den Kunden und der Fluggesellschaft angesehen werden. Dabei besteht definitionsgemäß ein enger Zusammenhang zwischen der Innensicht auf die Buchungsanfrage und der Black-Box-Perspektive, die durch die Schnittstellenbetrachtung erzeugt wird.

3.4.3 Zielorientierte Beschreibung

Wir haben bislang Spezifikationen als Mengen von Prozessen diskutiert, die ein zulässiges Verhalten darstellen. Die Frage nach Zulässigkeit orientiert sich hierbei primär an den Fähigkeiten der beteiligten Akteure und hat noch keine wirtschaftlichen Aspekte im Fokus. Für eine differenziertere Beschreibung geschäftlicher Vorgänge ist es jedoch auch von Interesse, ob diese aus einer wirtschaftlichen Sicht als erfolgreich anzusehen sind oder nicht. Wir unterstellen dabei, dass ein geschäftlicher Vorgang auf die Erfüllung eines Geschäftsziels ausgerichtet ist. [EHH+08, Coc01, MF11] Unter einem *Geschäftsziel* verstehen wir eine Bedingung, auf deren Basis der Erfolg eines Prozesses bestimmt werden kann. Im Fall der Buchungsanfrage können wir beispielsweise als Ziel definieren, dass Holiday Airways für eine vorliegende Kundenanfrage am gleichen Tag eine Verbindung anbieten kann.

Im Kontext der zielorientierten Vorgangsbetrachtung verwenden wir synonym zum Prozess auch den Begriff des *Szenarios*. Wir differenzieren Szenarien in *Erfolgsszenarien* und *nicht erfolgreiche Szenarien*. Ein Szenario wird als erfolgreich angesehen, wenn es ein gegebenes Geschäftsziel erfüllt. [Coc01] In unserem Beispiel wäre die Buchungsanfrage durch den Kunden Herr

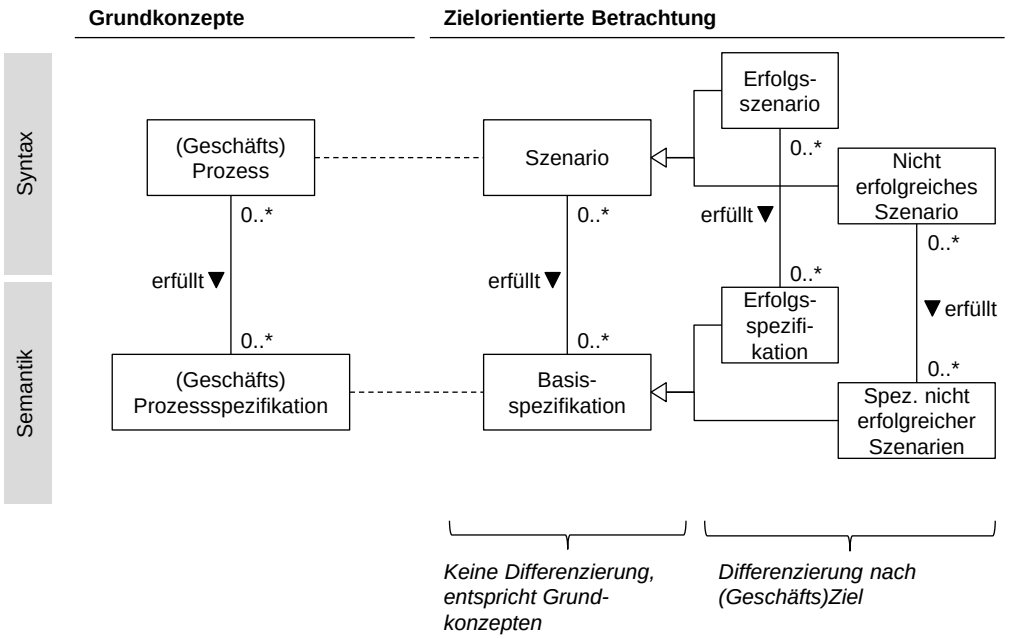

Abbildung 3.6: Konzepte zur zielorientierten Vorgangsbeschreibung, strukturiert anhand eines UML Klassendiagramms

Müller ein Erfolgsszenario, da ihm eine Verbindung am gleichen Tag angeboten werden kann. Nicht erfolgreiche Szenarien können sich insbesondere auch aus unzulässigen Anfragen – beispielsweise für Daten in der Vergangenheit – ergeben.

In der semantischen Auslegung betrachten wir Prozessspezifikationen in Kombination mit einem gegebenen Geschäftsziel. Daraus ergibt sich eine Menge von Prozessen, die der Spezifikation entsprechen und eine Teilmenge solcher Prozesse, die zusätzlich die im Geschäftsziel formulierte Bedingung erfüllen. Wir bezeichnen eine Semantik, die nach Zielerreichung differenziert ist, auch als einen *Anwendungsfall*. Ein Anwendungsfall beschreibt das Verhalten in einem geschäftlichen Vorgang als Menge von erfolgreichen und nicht erfolgreichen Szenarien. [Coc01] Unserem Beispiel folgend können wir die vorhin aufgestellte Spezifikation aller Anfragebearbeitungen durch Holiday Airways als Basisspezifikation des Anwendungsfalls heranziehen und betrachten eine Teilmenge dieser mit allen Prozessen, in denen die Fluggesellschaft eine Flugverbindung am gleichen Tag anbietet, als Erfolgsspezifikation des Anwendungsfalls.

3.5 Architekturbeschreibung

Zum Abschluss der Begriffsbildung werden wir uns noch mit der Struktur von Prozessen auseinandersetzen. Die Prozesse zur Bearbeitung von Kundenanfragen, die wir in den letzten Abschnitten untersucht haben, folgen einem bestimmten Ablaufmuster. Dieses lässt sich bereits aus

der Anschauung der Überblicksdarstellung in Abbildung 3.1 deutlich erkennen.

Wir bezeichnen die Struktur eines Prozesses als *Prozessarchitektur*. Eine Architektur erfasst im Allgemeinen die Bestandteile eines Betrachtungsobjekts und deren Beziehungen. Entsprechend legen wir eine Prozessarchitektur als Beschreibung der Aktionen (Bestandteile eines Prozesses) sowie der Kausalitäten zwischen den Ereignissen (Beziehungen zwischen den Prozessbestandteilen) aus. Wird eine Spezifikation auf Basis von Prozessregeln definiert, so ergeben sich naturgemäß die Kausalitäten aus den Regeln. Im Fall von Holiday Airways bedeutet dies, dass die Prozessarchitektur für Buchungsanfragen aus den fünf Aktionen besteht, die wir bereits diskutiert haben. Die Abhängigkeiten zwischen den Aktionen folgen aus den vier aufgestellten Prozessregeln. So stehen die Aktionen „Grundpreis berechnen" und „Gesamtpreis berechnen" in kausaler Abhängigkeit zueinander, während die Preisberechnung und die Verfügbarkeitsprüfung unabhängig voneinander durchgeführt werden können.

Prinzipiell können in einer regelbasierten Spezifikation beliebige Kausalitäten zwischen Aktionen bestehen. Insbesondere müssen Prozessregeln nicht aus der Sicht eines Akteurs formuliert werden. Um die Komplexität einer umfassenden Spezifikation jedoch in Grenzen zu halten, ist es wünschenswert, die Abhängigkeiten in Prozesse zu beschränken.

Wir betrachten dazu *dienstorientierte Architekturen* als eine spezielle Architekturausprägung. In dienstorientierten Prozessen hängen alle Aktionen eines beliebigen Akteurs ausschließlich von Aktionen des gleichen Akteurs oder von Schnittstellenaktionen ab, die der Akteur beobachten kann. Ein solcher Prozess lässt sich durch eine Komposition von Regeln beschreiben, die das Verhalten der einzelnen Akteure erfassen. Wir bezeichnen solche Regeln auch als *akteurorientierte Regeln*. Die für unser Beispiel eingeführten Prozessregeln von Holiday Airways sind bereits akteurorientiert formuliert.

Der Vorteil einer dienstorientierten Prozessarchitektur liegt, wie in Kapitel 2 diskutiert, in der Modularität und Komponierbarkeit. Da interne Aktionen eines Akteurs definitionsgemäß nicht für andere Akteure relevant sind, können diese ausgeblendet und Interaktionen zwischen Akteuren aus einer Black-Box-Perspektive anhand von Diensten betrachtet werden. Bei Holiday Airways ist so beispielsweise der Ablauf der Preisberechnung für das System zur Angebotserstellung nicht relevant. Dieses bezieht sich lediglich auf das Ergebnis, den ermittelten Gesamtpreis. Somit können die Beiträge der einzelnen Akteure zum Prozess als Module aufgefasst werden. Zudem gelten diese Überlegungen in gleicher Weise für Gruppen von Akteuren. So gilt wiederum, dass für die Kunden von Holiday Airways nur die Übermittlung des Angebots relevant ist und alle internen Aktionen der drei Systeme aus der Betrachtung ausgenommen werden können.

Eine dienstorientierte Architektur ermöglicht eine zunehmende Aggregation von Akteuren in mehreren Stufen, wobei in jeder Stufe die interne Aktionen der zu einer Gruppe zusammengefassten Akteure ausgeblendet werden können. Damit lässt sich ein Schichtenmodell konstruieren, das verschiedene *Sichten* auf einen Prozess zulässt. Die Sichten stellen dabei unterschiedliche Detaillierungsgrade des Prozesses dar. In der untersten Schicht werden alle Interaktionen zwischen allen am Prozess beteiligten Akteuren erfasst. In den höheren Schichten werden Gruppen von Akteuren, beispielsweise die gesamte Organisation von Holiday Airways, betrachtet und von internen Aktionen dieser Gruppe abstrahiert. Die Etablierung eines solchen Schichten-

modells, das an Diensten ausgerichtet ist, vereinfacht komplexe Spezifikationen deutlich. Zudem erweist es sich auch in der Entwurfsphase als vorteilhaft. Die Einführung von Schichten ermöglicht eine stufenweise Verfeinerung der Dienstarchitektur und gewährleistet gleichzeitig Konsistenz zwischen den Verfeinerungsstufen.

3.6 Zusammenfassung

Wir haben in diesem Kapitel einen systematischen Überblick über die Beschreibung geschäftlicher Abläufe erarbeitet. Zunächst lassen sich geschäftliche Vorgänge in syntaktischer und semantischer Weise definieren. Dazu haben wir als Grundkonzepte den Prozess und die Spezifikation eines Prozesses eingeführt.

Syntaktisch sehen wir einen Prozess als eine Menge aufgetretener Ereignisse an. Ein Ereignis stellt dabei die einmalige Durchführung einer Aktion dar. Jeder Aktion können wir einen Akteur zuweisen, der die Aktion ausführt, sowie eine Menge von Akteuren, für die die Ausführung der Aktion sichtbar ist. Auf semantischer Ebene beschreiben wir ein Verhalten auf Basis einer Prozessspezifikation als eine Menge von Prozessen, die als zulässiges Verhalten betrachtet werden. Prozesse in einer Spezifikation können durch systematische Anwendung von Prozessregeln entwickelt werden. Die transitive Hülle einer solchen Regelentwicklung resultiert in der Spezifikation.

Für eine präzisere Beschreibung geschäftlicher Abläufe können die Grundkonzepte Prozess und Spezifikation eines Prozesses in unterschiedlicher Weise betrachtet werden (siehe auch Abbildung 3.7). Wir unterscheiden zwischen Projektionen, die die Betrachtung der Konzepte auf bestimmte Ausschnitte fokussieren, und einer Partition, die eine Differenzierung der Abläufe vornimmt.

Wir haben zwei gängige Projektionen untersucht. Zum einen beschränken wir mit Transaktionen die Betrachtung auf einen Ablauf eines geschäftlichen Gesamtvorgangs mit zusammengehörigen Aktionen, beispielsweise alle eine Kundenanfrage betreffenden Aktionen. Ein Transaktionsfluss im Kontrast dazu umfasst eine Menge paralleler und sequentieller Abläufe. Eine Transaktion kann als Instanz eines Transaktionsflusses angesehen werden.

Ein Dienst fokussiert die Betrachtung auf die Schnittstellenaktionen eines Akteurs beziehungsweise einer Gruppe von Akteuren. Auf diese Weise kann der Beitrag, den bestimmte Akteure zu einem Prozess leisten, präzise erfasst werden. Ein Dienst erfasst in Abhängigkeit von beobachteten Schnittstellenaktionen, dem Dienstauftrag, die durch die Akteure zu erbringenden Schnittstellenaktionen als Wirkung des Dienstes.

Schließlich haben wir noch eine Partitionierung von Prozessen in erfolgreiche und nicht erfolgreiche Szenarien betrachtet. Die Unterscheidung in die beiden Partitionen erfolgt anhand einer Bedingung, die wir als Ziel oder Geschäftsziel bezeichnen. Entsprechend können wir anhand des Ziels die Prozessspezifikation in eine Erfolgsspezifikation als Menge aller erfolgreichen Szenarien und eine Spezifikation nicht erfolgreicher Szenarien differenzieren. Die Menge aller erfolgreichen und nicht erfolgreichen Szenarien bezeichnen wir in diesem Kontext auch als Anwendungsfall.

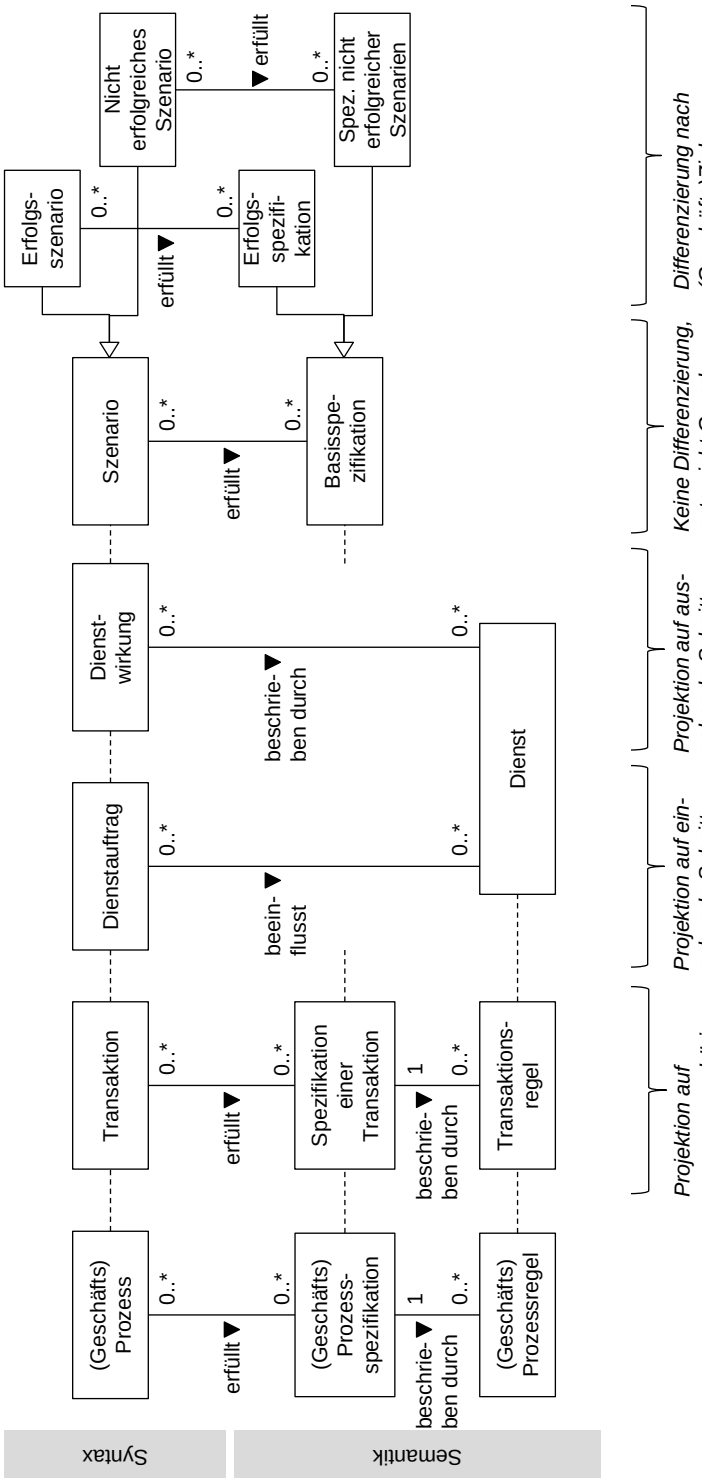

Abbildung 3.7: Konzepte zur Ablaufbeschreibung im Überblick, strukturiert anhand eines UML Klassendiagramms

4 Modellierung

In diesem Kapitel leiten wir nun sukzessive ein formales Modell zur Beschreibung der im vorangegangenen Kapitel hergeleiteten Konzepte her. Ziel des Modells ist eine formale und damit präzise Darstellbarkeit der einzelnen Konzepte sowie eine klare Abgrenzung und Einordnung. Wir werden die formale Darstellung weiter im Kontext des eingeführten Praxisbeispiels diskutieren, um den Praxisbezug aufrecht zu erhalten. Die Modellierungsziele richten sich nach den Anforderungen, die wir im Abschnitt 2.2.4 aufgestellt haben.

4.1 Modellierungsansatz

Geschäftliche Abläufe, wie wir sie im letzten Kapitel eingeführt haben, können durch eine Vielzahl von Modellen dargestellt werden. Jedes Modell bedeutet dabei eine Einschränkung der Konzepte, da im Zuge der formalen Modellierung Annahmen bezüglich der Realität getroffen werden müssen. Entsprechend sind solche konzeptuellen Einschränkungen auch im hier vorgestellten Modell unvermeidbar. Wir geben daher zunächst einen Überblick über die Annahmen und Prinzipien, die dem Modell zugrundeliegen.

Dazu gehen wir zuerst auf die Grundsätze unseres Modellierungsansatzes ein und stellen anschließend den Aufbau des Modells vor. Ziel dieses Vorgehens ist es, eine möglichst große Transparenz über die getroffenen Modellierungsannahmen und deren Auswirkungen zu schaffen. Zudem vereinfacht die modulare Einführung des Modells auch Anpassungen an individuelle Anforderungen.

4.1.1 Modellierungsgrundsätze

Mit den folgenden Grundsätzen stellen wir die Gestaltungsprinzipien des Modells dar und adressieren dabei den strukturellen Aufbau des Modells, die gewählte syntaktische Repräsentation von Abläufen sowie den semantischen Beschreibungsansatz des Modells.

Systematische Verknüpfung von theoretischer Fundierung und praktischer Anwendung

Unser Modell zielt auf eine präzise Beschreibung geschäftlicher Abläufe ab. Gleichzeitig intendieren wir eine exakte Abgrenzung der unterschiedlichen Konzepte zur Beschreibung solcher Abläufe. Dies erfordert eine systematische theoretische Fundierung aller vorgestellten Begriffe. Wir setzen das Modell zunächst auf einem schlanken, sehr eng an etablierten Konzepten

der theoretischen Informatik ausgerichteten abstrakten Grundmodell auf und erweitern dieses anschließend sukzessive. Im Grundmodell beschränken wir uns auf die Untersuchung von Prozessen und dazugehörigen Spezifikationen, die wir als Basiskonzepte zur Beschreibung von Abläufen betrachten. Alle weiteren Konzepte leiten wir konsequent als Sichten auf einen Prozess ab und stellen so einen präzisen Bezug zwischen den Konzepten her.

Eine starke theoretische Fundierung des Modells birgt jedoch auch das Risiko, das intuitive Verständnis der diskutierten Begriffe zu erschweren und damit die praktische Anwendbarkeit des Modells einzuschränken. Um diesem entgegenzuwirken, führen wir bereits im abstrakten Modell die Diskussion der eingeführten Konzepte anhand unseres Praxisbeispiels, das wir im letzten Kapitel vorgestellt haben. Zudem verfolgt unsere Modellierung den Anspruch, dass alle Konzeptbezeichnungen sowie Beziehungen, die zwischen den Konzepten hergestellt werden, dem intuitiven, in der Praxis etablierten Verständnis folgen und gerecht werden.

Das Modell zielt damit insgesamt darauf ab, eine Brücke zwischen theoretischer Fundierung und praktischer Anwendung zu schlagen. Durch die theoretische Fundierung wird eine präzise Einordnung und Beschreibbarkeit aller vorgestellten Begriffe gewährleistet. Die Verankerung der eingeführten Konzepte im existierenden praktischen Verständnis ermöglicht eine Anwendung der im Modell gewonnenen Erkenntnisse auch jenseits der formalen Ebene, die im Modell zum Einsatz kommt.

Ereignisbasierte Repräsentation von Abläufen

Es existiert eine Vielzahl unterschiedlicher Ansätze, wie Abläufe formal dargestellt werden können. Dabei stellt sich prinzipiell die Frage, wie die einzelnen Schritte, aus denen sich Abläufe zusammensetzen, mathematisch repräsentiert werden. Gängige Ansätze sind Graphenmodelle, Netzstrukturen (beispielsweise Petri-Netze), Ein-/Ausgaberelationen und ereignisbasierte Ablaufdarstellungen. Weiterhin können Abläufe auch als Termsysteme erfasst werden. [Gla90] Wir wählen in unserem Modell eine ereignisbasierte Repräsentation von Abläufen. Wir fassen dabei ein Ereignis als die einmalige Durchführung einer geschäftlichen Handlung auf. Ein Ablauf lässt sich in dieser Sichtweise als eine Menge von sequentiell und parallel auftretenden Ereignissen erfassen. Jedem Ereignis kann ein Auftrittszeitpunkt zugewiesen werden, woraus sich eine temporale Ordnung der Ereignisse in einem Ablauf ergibt.

Die Ereignisorientierung erweist sich für unser Modell als vorteilhaft, da auf Grund der feingranularen Natur der Ereignisse die unterschiedlichen Projektionen und Partitionen, die wir im Zuge der Modellierung diskutieren werden, einfach abgebildet werden können. Zudem sehen wir Ereignisse als einen guten Kompromiss an als ein Konzept, das sich einerseits präzise formalisieren lässt, andererseits aber leichtgewichtig genug ist, um auch komplexe Ablaufstrukturen intuitiv erfassen zu können.

Regelbasierte Beschreibung von Abläufen

Neben der syntaktischen Erfassung von Abläufen befassen wir uns umfassend mit deren semantischer Beschreibung. Wir stellen dazu Spezifikationen auf, die Prozessmengen definieren

und auf diese Weise eine semantische Aussage bezüglich der in der Menge enthaltenen Prozesse treffen. Spezifikationen können auf Basis unterschiedlicher Beschreibungsansätze formuliert werden.

Wir befassen uns primär mit der regelbasierte Spezifikation von Prozessen. Wir betrachten einen Ablauf als eine Folge sequentieller und paralleler Schritte. Der Grundgedanke einer regelbasierten Beschreibung besteht darin, entlang eines Ablaufs Aussagen zu treffen, die einen Zusammenhang zwischen bereits stattgefundenen und weiterhin durchzuführenden Aktionen herstellen. Es handelt sich dabei also um eine relative Beschreibungsform. Wir formulieren Regeln für Prozesse, mit denen wir festlegen, nach welcher Logik ein bestehender Ablauf fortzusetzen ist.

Dies basiert auf der Annahme, dass an komplexen Abläufen eine Vielzahl von Akteuren beteiligt sind, die zu den Abläufen beitragen. Eine ganzheitliche Beschreibung eines solchen arbeitsteiligen Ablaufs setzt eine akteurübergreifende Perspektive voraus. Im geschäftlichen Umfeld ist eine Gesamtperspektive insbesondere in arbeitsteiligen Konstellationen jedoch nicht immer realisierbar und auf Grund der hohen Komplexität auch nicht zweckmäßig. Wir erfassen daher stattdessen den Beitrag einzelner Akteure zu Abläufen in Form von Regeln und entwickeln einen Ablauf inkrementell, indem wir diesen aus den Akteurskontributionen zusammensetzen und potenzielle Wechselwirkungen berücksichtigen.

Wir gehen dabei grundsätzlich von einem reaktiven Verhaltensmuster der am Ablauf beteiligten Akteure aus. Die Akteure wählen ihre Handlungen auf Basis von Beobachtungen fremder Handlungen und setzen so einen bestehenden Ablauf in einer bestimmten Weise fort. Selbstverständlich kann auch ein proaktives Vorgehen modelliert werden, indem Handlungen unabhängig von einer Beobachtung beschrieben werden. Wir betrachten dieses jedoch als eine besondere Ausprägung der regelbasierten Ablaufbeschreibung.

4.1.2 Modellierungsvorgehen

Wir führen das Modell in drei Stufen ein. Die mehrstufige Vorgehensweise erweist sich dabei in mehrfacher Hinsicht als vorteilhaft. Zum einen ermöglicht sie eine klare mathematische Fundierung des Modells, da wir ein umfassendes fachliches Modell auf Basis eines schlanken, formal gehaltenen Grundmodells entwickeln werden. Gleichzeitig führt der mehrstufige Aufbau zur einer Reduktion der Modellkomplexität, da die Eigenschaften der Stufen aufeinander aufbauen und zugleich ein klarer Bezug aller Modelleigenschaften zu etablierten mathematischen Konzepten hergestellt werden kann.

Weiterhin verdeutlicht der Stufenansatz die getroffenen Modellierungsannahmen dahingehend, dass wir aus einer generischen mathematischen Perspektive schrittweise ein Anwendungsmodell entwickeln. Auf diese Weise können Annahmen und deren Auswirkungen auf die Konzeptbeschreibung leicht nachvollzogen werden. Zudem vereinfacht das Vorgehen die Anpassbarkeit des Modells. Durch den stufenweisen Aufbau des Modells können Auswirkungen von Änderungen einfach erfasst und entsprechend nachgehalten werden.

Die Abbildung 4.1 zeigt den Aufbau des Modells im Überblick. Die drei Stufen gestalten sich – ausgehend von der abstraktesten Stufe – wie folgt:

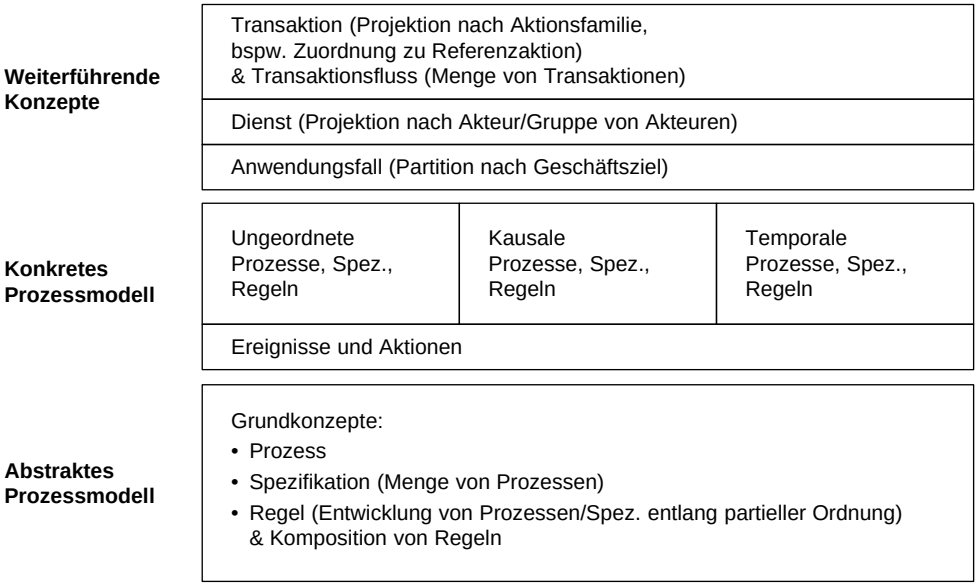

Abbildung 4.1: Modellübersicht

- *Abstraktes Prozessmodell*
  Im abstrakten Teil des Modells führen wir den Prozessbegriff zunächst unabhängig von einer bestimmten Repräsentation ein. Dies ermöglicht eine schlanke Formalisierung und eine Vermeidung von Redundanzen in der Diskussion grundlegender Eigenschaften. Wir befassen uns primär mit der Semantik von Prozessen. Dazu betrachten wir Spezifikationen als Prozessmengen mit einer zugrundeliegenden semantischen Aussage und etablieren Prozessregeln als strukturierten Beschreibungsansatz solcher Spezifikationen. Wir führen eine Ordnung ein, entlang der Prozesse regelbasiert entwickelt werden können, und diskutieren die formalen Eigenschaften der Prozessentwicklung.
- *Konkretes Prozessmodell*
  In der zweiten Stufe führen wir Ereignisse und Aktionen ein und stellen drei verschiedene Repräsentationen eines Prozesses vor, die alle auf Ereignissen basieren, sich aber in der Auslegung von Kausalität unterscheiden. Wir fokussieren dabei weiterhin auf Prozesse und deren regelbasierter Spezifikation. Dabei zeigen wir, dass sich die Erkenntnisse des abstrakten Modells unmittelbar anwenden lassen. Zudem führen wir Akteure als Subjekte der Aktionen ein, um auch Abläufe exakt darstellen zu können, an denen eine Vielzahl von Akteuren beteiligt sind.
- *Weiterführende Konzepte*
  In der letzten Stufe führen wir schließlich zusätzliche Sichten auf die bislang etablierten Begriffe Prozess, Spezifikation und Prozessregel ein, indem wir Projektionen und Partitionen auf die in den Prozessen dargestellten Ereignismengen vornehmen. Dies ermöglicht

die systematische Herleitung und Einordnung der unterschiedlichen Konzepte, die in der Beschreibung von dienstorientierten Architekturen zum Einsatz kommen.

Zum Schluss des Kapitels werden wir uns mit der Modellierung von Architekturen auseinandersetzten und damit zeigen, wie die eingeführten Konzepte zu einer dienstorientierten Architektur zusammengefügt werden können. Durch das gewählte Vorgehen spannen wir für die Beschreibung von geschäftlichen Abläufen einen Bogen von einer abstrakten formalen algebraischen Struktur hin zu einer konkreten betriebswirtschaftlichen Auslegung. Wir betrachten unser Modell dabei lediglich als *eine* mögliche formale Grundlage von vielen, die uns als Werkzeug dient, um eine präzise Einordnung und Abgrenzung betrieblicher Handlungskonzepte vorzunehmen. Die gewonnen Erkenntnisse lassen sich prinzipiell auch in andere Modelle übertragen.

Wir modellieren die Konzepte einer dienstorientierten Architektur rein aus einer fachlichen Perspektive und abstrahieren dabei vollständig von technischen Aspekten. So kann das Modell unabhängig von einer bestimmten Implementierungstechnologie in jeder dienstorientierten Umgebung eingesetzt werden. Durch die konsequente formale Fundierung ist jedoch eine Überführung in ein Automatenmodell und damit eine technische Abbildung durch den Modellierungsansatz gewährleistet.

4.2 Abstraktes Prozessmodell

Wir betrachten Prozesse zunächst abstrakt und unabhängig von einer bestimmte Repräsentation. Wir fokussieren auf die semantische Erfassung von Prozessen sowie allgemeingültige Eigenschaften. Die folgenden Überlegungen schaffen dabei die formale Grundlage für die Diskussion unterschiedlicher Prozessrepräsentationen im nächsten Abschnitt. Um jedoch die Anschaulichkeit der vorgestellten Konzepte zu gewährleisten, werden wir die folgenden Beispiele bereits im Kontext des Geschäftsvorgangs der Flugbuchung diskutieren.

4.2.1 Prozess

Mit dem Konzept des Prozesses erfassen wir einen Ablauf. Wir gehen dabei davon aus, dass Abläufe komponierbar sind, also die Vereinigung zweier Abläufe wieder einen Ablauf ergibt. Umgekehrt kann ein Ablauf auch dekomponiert werden in Teilabläufe. Die Teilabläufe bezeichnen wir dabei auch als Schritte des Ablaufs. Wir werden uns im Folgenden ausführlich mit der Entwicklung von Abläufen auseinandersetzen. Dabei verstehen wir unter einer Entwicklung eines Ablaufs, wenn dieser um weitere Schritte ergänzt wird, also zusätzliche Handlungen durchgeführt werden. Somit kann eine Ordnung bezüglich des Entwicklungsstandes eines Ablaufs definiert werden, die wir als Entwicklungsordnung bezeichnen. Wir setzen damit Abläufe zueinander in Beziehung, wenn der eine Ablauf durch Ergänzung weiterer Schritte aus dem anderen Ablauf entstanden ist.

Definition 1. Prozess

Wir verstehen unter einem Prozess die Beschreibung eines Ablaufs. Zwischen zwei Prozessen lässt sich ein Bezug herstellen, wenn der eine Prozess einen Teilablauf des anderen Prozesses

im Sinne eines Zwischenstadiums darstellt. Wir beschreiben solche Zusammenhänge auf Basis einer Halbordnung, die wir als Entwicklungsordnung bezeichnen. Wir gehen davon aus, dass jede Prozessentwicklung entlang dieser Ordnung beschränkt ist und folglich ein Supremum angegeben werden kann.

Formal erfassen wir die Menge aller Prozesse zusammen mit der Entwicklungsordnung wie folgt:

$$(\mathit{Processes}, \preceq)$$

Die Halbordnung $\preceq \subseteq \mathit{Processes} \times \mathit{Processes}$ ist vollständig und setzt zwei Prozesse $P, Q \in \mathit{Processes}$ zueinander in Beziehung $P \preceq Q$, falls der in Q beschriebene Ablauf alle Schritte des Ablaufs P enthält. □

Da wir in diesem Abschnitt Prozesse abstrakt betrachten, setzen wir die Existenz einer vollständigen Halbordnung über den Prozessen zunächst voraus, ohne diese formal zu definieren. Wir werden die formale Einführung der Entwicklungsordnung im weiteren Verlauf unserer Diskussion im Abschnitt 4.3 anhand der jeweiligen Prozessrepräsentation vornehmen.

Als Halbordnung ist $\preceq$ reflexiv, antisymmetrisch und transitiv. Die Ordnung besitzt ein Nullelement $\emptyset$, das einem Prozess ohne Handlung entspricht. Somit kann für jede Teilmenge von Prozessen ein Infimum konstruiert werden. Das Infimum entspricht dabei einem Prozess mit einer größtmöglichen Anzahl von Schritten, die in allen Prozessen der Teilmenge in gleicher Weise durchgeführt werden. Mit der Ordnung $\preceq$ lässt sich die Entwicklung eines beliebigen Prozesses $Q \in \mathit{Processes}$ anhand einer vollständig geordneten Teilmenge $\{P_1, P_2, \ldots, P_n\} \in \mathit{Processes}$ beschreiben:

$$\emptyset \preceq P_1 \preceq P_2 \preceq \ldots \preceq P_n \preceq Q$$

Die Prozesse $P_1, P_2, \ldots, P_n$ stellen dabei Zwischenstadien des Prozesses Q dar und es gilt für zwei Prozesse P_i, P_{i+1} mit $1 \leq i \leq n-1$, dass P_{i+1} alle Prozessschritte von P_i sowie zusätzliche Schritte enthält, die als Fortsetzung von P_i betrachtet werden können.

Definition 2. Teilprozess

Wir bezeichnen einen Prozess $P \in \mathit{Processes}$ als einen Teilprozess von $Q \in \mathit{Processes}$, falls gilt:

$$P \preceq Q$$

□

Ein Teilprozess kann als ein Zwischenstadium eines Prozesses angesehen werden, auf Basis dessen sich der Prozess weiter entwickeln kann. Er fokussiert auf einen bestimmten Ausschnitt des zugrundeliegenden Prozesses. Eine besondere Ausprägung eines Teilprozesses ist das Präfix, das in temporalen Prozessmodellen die Betrachtung auf den Anfang eines Ablaufs beschränkt.

Beispiel. Wir illustrieren das bislang abstrakt diskutierte Konzept des Prozesses am Beispiel einer konkreten Buchungsanfrage bei Holiday Airways. Abbildung 4.2 stellt anhand der Anfrage von Herrn Müller den Ablauf einer Anfragebearbeitung durch Holiday Airways dar. Wir nehmen

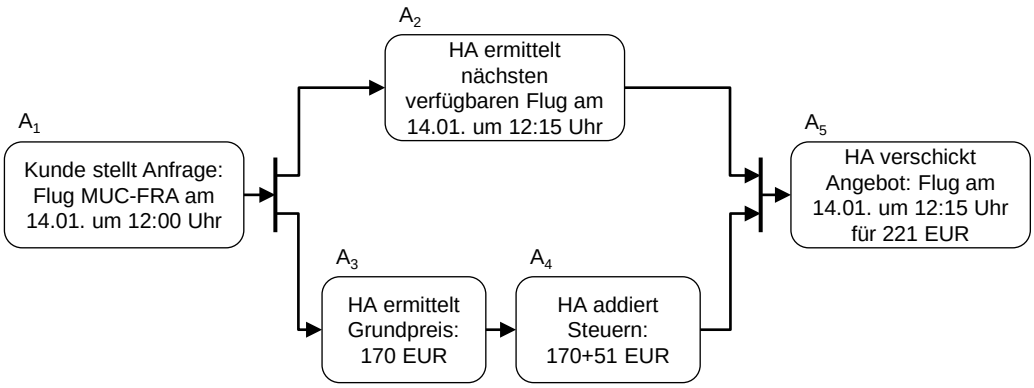

Abbildung 4.2: Buchungsanfrage von Herrn Müller bei Holiday Airways (HA), dargestellt als UML Aktivitätsdiagramm

dabei zunächst eine ergebnisorientierte Perspektive ein. Für die einzelnen Schritte gehen wir dazu nicht auf das Vorgehen, beispielsweise den Algorithmus zur Ermittlung des Grundpreises, ein, sondern erfassen in fünf Aktionen die Ergebnisse der jeweiligen Schritte. Wir werden im weiteren Verlauf unserer Ausführung auch Vorgehensaspekte untersuchen.

Im abgebildeten Beispiel übermittelt Herr Müller eine Anfrage für einen Flug auf der Strecke von München nach Frankfurt (MUC-FRA) am 14. Januar um 12:00 Uhr. Nach Erhalt der Buchungsanfrage ermittelt Holiday Airways einerseits die nächstmögliche Flugverbindung, andererseits berechnet sie den Preis für die gegebene Strecke. Wir gehen davon aus, dass der nächste Flug um 12:15 Uhr verfügbar ist und der Preis für diesen 170 EUR zuzüglich eines pauschalen Aufschlags in Höhe von 30% für Steuern und Gebühren beträgt. Die Ermittlung der Flugverfügbarkeit und die Preisberechnung beeinflussen sich nicht gegenseitig und können daher parallel durchgeführt werden. Zum Schluss stellt Holiday Airways die ermittelte Verbindung und den Preis zu einem Angebot zusammen und schickt dieses zurück an den Kunden.

Wir repräsentieren die fünf Schritte der Buchungsanfrage mit $A_1, \ldots, A_5$, wie in der Abbildung dargestellt. Für einen Prozess, der sich aus diesen fünf Schritten ergibt, verwenden wir zunächst folgende informelle Notation:

$$\langle A_1 A_2 A_3 A_4 A_5 \rangle$$

Dabei gehen wir vorerst von sequentiellen Abläufen aus, in denen die Schritte in der angegebenen Reihenfolge ausgeführt werden. Im nächsten Kapitel werden wir eine formale Repräsentation von Prozessen einführen, die auch die Modellierung paralleler Abläufe zulässt. Aus den einzelnen Schritten können nun verschiedene Prozesse zusammengestellt werden. Jede Kombination dieser Schritte stellt einen Ablauf und damit einen Prozess dar. Wir betrachten exemplarisch folgende Prozesse:

- $P_1 = \langle A_1 \rangle$: Kunde hat Anfrage gestellt
- $P_2 = \langle A_1 A_2 \rangle$: Kunde hat Anfrage gestellt und nächste verfügbare Verbindung wurde ermittelt
- $P_3 = \langle A_1 A_3 \rangle$: Kunde hat Anfrage gestellt und Grundpreis für die Strecke wurde ermittelt
- $P_4 = \langle A_1 A_3 A_4 \rangle$: Kunde hat Anfrage gestellt und es wurden Grund- und Gesamtpreis für die Strecke ermittelt
- $P_5 = \langle A_1 A_3 A_4 A_2 \rangle$: Kunde hat Anfrage gestellt, Grund- und Gesamtpreis sowie die nächste verfügbare Verbindung wurden ermittelt
- $P_6 = \langle A_1 A_2 A_3 A_4 A_5 \rangle$: Anfrage wurde vollständig bearbeitet, wobei die Verbindung vor dem Flugpreis ermittelt wurde
- $P_7 = \langle A_1 A_3 A_4 A_2 A_5 \rangle$: Anfrage wurde vollständig bearbeitet, wobei der Flugpreis vor der Verbindung ermittelt wurde

Zwischen diesen Prozessen bestehen beispielshalber folgende Entwicklungszusammenhänge:

- $P_1 \preceq P_2 \preceq P_6$
- $P_1 \preceq P_3 \preceq P_4 \preceq P_5 \preceq P_7$

Das Beispiel verdeutlicht, wie sich ein Prozess inkrementell durch zusätzliche Handlungen entwickeln kann. Die Zwischenstadien der Anfragebearbeitung P_1, P_2, P_3, P_4, P_5 stellen dabei Teilprozesse der vollständig bearbeiteten Anfrage P_6 und P_7 dar. In der gegebenen sequentiellen Prozessauslegung dieses Beispiels sind sie zugleich Präfixe der vollständigen Anfragebearbeitung. □

4.2.2 Prozessspezifikation

Bisher haben wir nur einzelne Prozesse und Beziehungen zwischen den Prozessen entlang der Entwicklungsordnung betrachtet. Nun fassen wir mehrere Prozesse zu Prozessmengen zusammen. Auf diese Weise können semantische Aussagen getroffen werden, indem wir eine Menge von Prozessen bilden, die wir als äquivalent bezüglich einer semantischen Aussage betrachten.

Wir bezeichnen eine solche Menge von Prozessen mit einer zugrundeliegenden semantischen Aussage auch als Spezifikation. Es gilt an dieser Stelle zu beachten, dass in der Literatur häufig zwischen der syntaktischen und semantischen Auslegung einer Spezifikation differenziert wird. [Wat96, BS99] Dieser Sichtweise folgend beschreibt eine syntaktische Spezifikation einen Prozess auf Basis einer formalen strukturierten Notation. Durch Festlegung einer Semantik für die eingesetzte Syntax kann aus der syntaktischen eine semantische Spezifikation abgeleitet werden, die die Menge aller Prozesse enthält, die der syntaktischen Beschreibung in der gegebenen Semantik folgen.

Wir fokussieren uns im weiteren Verlauf der Diskussion auf die semantische Betrachtung von Spezifikationen. Wenn wir an einzelnen Stellen auf syntaktische Beschreibungsansätze für Prozesse eingehen, werden wir diese explizit als syntaktische Spezifikationen kennzeichnen.

Definition 3. Prozessspezifikation

Eine (semantische) Prozessspezifikation ist eine Teilmenge aller Prozesse $S \subseteq \mathit{Processes}$, auf die eine bestimmte semantische Aussage zutrifft. □

Wir bezeichnen die Menge aller Prozessspezifikationen als:

$$\mathit{Spec} = \mathscr{P}(\mathit{Processes})$$

Eine Prozessspezifikation ist somit eine formale Sprache, die eine Menge von Prozessen enthält, die einer bestimmten Semantik folgen. Die Menge der Prozessspezifikationen bildet zusammen mit der Teilmengenrelation einen vollständigen Mengenverband $(\mathit{Spec}, \subseteq)$, der sich direkt aus der Potenzmengeneigenschaft ableitet. Der Nullelement des Verbandes ist die leere Spezifikation $\{\}$, das Einselement die Menge aller Prozesse *Processes*. Entsprechend stellt $(\mathit{Spec}, \subseteq)$ auch eine vollständige Halbordnung dar. Die Eigenschaften der Reflexivität, Antisymmetrie und Transitivität folgen direkt aus der Teilmengendefinition. Das Supremum jeder aufsteigenden Kette ist die Spezifikation mit den meisten Prozessen in dieser Kette, im maximalen Fall das Einselement des Verbandes.

Beispiel. Unserem Beispiel folgend können wir eine Spezifikation aller zulässigen Abläufe zur Bearbeitung der soeben vorgestellten Buchungsanfrage von Herrn Müller angeben, die wir mit $S_m \in \mathit{Spec}$ bezeichnen:

$S_m = \{\langle A_1 \rangle, \langle A_1 A_2 \rangle, \langle A_1 A_3 \rangle, \langle A_1 A_3 A_4 \rangle, \langle A_1 A_2 A_3 \rangle, \langle A_1 A_3 A_2 \rangle, \langle A_1 A_2 A_3 A_4 \rangle,$

$\langle A_1 A_3 A_2 A_4 \rangle, \langle A_1 A_3 A_4 A_2 \rangle, \langle A_1 A_2 A_3 A_4 A_5 \rangle, \langle A_1 A_3 A_2 A_4 A_5 \rangle, \langle A_1 A_3 A_4 A_2 A_5 \rangle\}$

Wir erkennen unmittelbar, dass die vorgestellten Prozesse $P_1, \ldots, P_7$ Teil der Spezifikation sind. Umgekehrt gibt es Schrittfolgen, die nicht der Spezifikation entsprechen. So kann Holiday Airways beispielsweise keinen Preis berechnen, ohne vorher eine entsprechende Anfrage erhalten zu haben: $\langle A_3 A_4 \rangle \notin S_m$. Die Spezifikation trifft somit eine semantische Aussage zu Prozessen, indem sie geforderte Zusammenhänge zwischen den Schritten darstellt. In diesem trivialen Beispiel ist die Aussagekraft natürlich beschränkt, da nur eine Kundenanfrage behandelt wird. Eine Spezifikation der Fluggesellschaft in voller Allgemeinheit würde Aussagen zu Behandlung aller möglichen Kundenanfragen enthalten. □

Das Beispiel zeigt, dass bereits für einen sehr einfachen Sachverhalt die Spezifikation eine große Menge von Prozessen enthält. Es ist daher nicht praktikabel, für die Beschreibung komplexer Prozesse Spezifikationsmengen direkt anzugeben. Stattdessen kann mittels einer syntaktischen Spezifikation, die einem formalen Beschreibungsmodell folgt, eine Prozessmenge systematisch konstruiert werden. Dazu können im Rahmen der operationellen Semantik Automatenmodelle aufgestellt werden, anhand der Prozesse einer Spezifikation zugeordnet werden können. Zudem können auch mittels einer denotationellen Semantik aus einem formalen Aussagensystem Spezifikationsmengen hergeleitet werden. Für eine ausführliche Diskussion semantischer Beschreibungsansätze von Prozessen verweisen wir auf [Gla90, BO01].

Jedoch ist auch das Aufstellen einer vollständigen Semantik für Prozesse anhand eines formalen Beschreibungsmodells mit signifikanten Aufwänden verbunden. Wir werden uns daher im

Folgenden mit Prozessregeln auseinandersetzen. Diese treffen relative Aussagen zu Prozessen, indem sie Ursache-/Wirkungszusammenhänge beschreiben und somit Kausalitäten erfassen. Auf diese Weise kann die semantische Betrachtung auf einzelne Ausschnitte eines Prozesses fokussiert werden, anstatt eine vollständige Semantik herzuleiten. Die Prozessregeln sind vergleichbar mit der Anwendung von Produktionsregeln auf eine formale Sprache. Wir entwickeln Prozesse und damit verbunden Spezifikationen, indem wir einen gegebenen Prozess, den wir auch als Prozessannahme bezeichnen, anhand von Regeln systematisch fortsetzen. Wir werden zeigen, dass sich solche Regeln feingranular definieren lassen, damit Zusammenhänge in möglichst einfacher und abgegrenzter Weise beschrieben werden können. Mittels Komposition können Regeln dann zu umfassenden Regelwerken zusammengesetzt werden. Dazu werden wir in den folgenden Abschnitten ausführlich diskutieren, wie solche Produktionsregeln formuliert und komponiert werden können.

4.2.3 Prozessregel

Wir haben mit der Teilprozessrelation bereits einen Zusammenhang zwischen Prozessen kennengelernt. Jedoch trifft diese Relation nur eine Aussage, ob ein Ablauf in einem anderen enthalten ist, ohne einen semantischen Bezug zwischen den Schritten eines Ablaufs herzustellen. Für die semantische Beschreibung von Prozessen ist nun von Interesse, welche Prozesse sich in einer gegebenen semantischen Auslegung aus einem beobachteten Prozess heraus entwickeln können. So würden wir unserem Beispiel folgend fordern, dass Preisberechnungen nur auf Basis einer vorliegenden Buchungsanfrage durchgeführt werden.

Mit Prozessregeln führen wir ein Konzept ein, mit dem wir Abhängigkeiten zwischen den Entwicklungsstadien eines Prozesses feingranular beschreiben können. Wir befassen uns zunächst mit der elementaren Definition von Prozessregeln für einzelne Prozesse und diskutieren im Anschluss die Komposition von Regeln sowie die Anwendung auf Spezifikationen als Mengen von Prozessen.

4.2.3.1 Elementare Definition

Prozessregeln beschreiben Zusammenhänge zwischen den Schritten in Prozessen. Eine Regel ergänzt dabei einen Prozess um bestimmte weitere Aktionen unter der Maßgabe einer durch die Regel formulierten Bedingung. Dazu analysiert sie zunächst eine Teilmenge der in einem Prozess enthaltenen Schritte. Die Analysephase lässt sich als Herleitung eines Zustandes auffassen. Durch Ergänzung weiterer Schritte in einem Prozess führt die Regel einen Zustandsübergang durch, da sie einen gegebenen Prozess in einen weiter entwickelten Prozess überführt, der bei erneuter Anwendung der Regel potenziell zu einem anderen Zustand führen kann. Eine einzelne Regel bezieht sich dabei typischerweise nur auf einen sehr kleinen Ausschnitt, also bestimmte Schritte eines Prozesses, um die Beschreibungskomplexität der Regel in Grenzen zu halten.

Definition 4. Prozessregel

Wir definieren eine Prozessregel als eine homogene Relation

$$\longrightarrow \subseteq \preceq$$

Die Anwendung einer Regel $P \longrightarrow Q$ beschreibt die Entwicklung oder auch Fortführung eines Prozesses P in einen Prozess Q. Wir fordern dabei, dass P einen Teilprozess von Q darstellt und definieren daher jede Regel als Teilmenge der Entwicklungsordnung $\preceq$.

Zur Unterscheidung von Prozessregeln führen wir Bezeichner ein, die wir über dem Ableitungspfeil der Regel notieren:

$$P \xrightarrow{r} Q$$

Wir bezeichnen eine Prozessregel mit Bezeichner r, die formal als Relation $\xrightarrow{r} \subseteq \preceq$ definiert ist, auch abkürzend als Prozessregel r. □

Die Menge aller Prozessregeln geben wir an mit:

$$Rules = \mathscr{P}(\preceq)$$

Jede Regel basiert auf einem beobachteten Prozess, der durch Anwendung der Regel weiter entwickelt wird. Dies beruht auf der Grundannahme, dass Regelwerke immer nur einen Ausschnitt der Realität beschreiben können. Sie bauen auf einer Annahme auf, mit der die Existenz eines gewissen Ablaufs vorausgesetzt wird. Für eine Regel r erfassen wir deren Definitionsbereich $Dom(r)$ wie folgt:

$$Dom(r) = \left\{ P \in Processes : \exists Q \in Processes : P \xrightarrow{r} Q \right\}$$

Eine Prozessregel r ist total, wenn $Dom(r) = Processes$ gilt. Eine totale Prozessregel trifft für jeden Prozess eine Entwicklungsaussage. Dabei müssen allerdings nicht in jedem Fall Schritte hinzugefügt werden. Die Regel kann der obigen Definition folgend einen Prozess auch unverändert belassen.

Als Wertebereich einer Regel r bezeichnen wir die Menge aller Prozesse, die von dieser erzeugt werden können. Wir definieren den Wertebereich formal als

$$Range(r) = \left\{ Q \in Processes : \exists P \in Dom(r) : P \xrightarrow{r} Q \right\}$$

Prozessregeln können so definiert sein, dass sie einem beobachteten Prozess mehrere resultierende Prozesse zuordnen. Wir bezeichnen dies als ein nicht deterministisches Verhalten. In diesem Fall stellt jeder dieser Prozesse eine zulässige Ableitung aus dem ursprünglichen Prozess dar. Eine Regel r ist deterministisch, wenn sie als funktionale Relation definiert ist:

$$\forall P, Q, R \in Processes : P \xrightarrow{r} Q \wedge P \xrightarrow{r} R \Rightarrow Q = R$$

Eine besondere Prozessregel ist die Identität, die keine Änderungen an beobachteten Prozessen vornimmt. Wir stellen diese als $\xrightarrow{id} \in Rules$ dar mit $P \xrightarrow{id} Q \Leftrightarrow P = Q$.

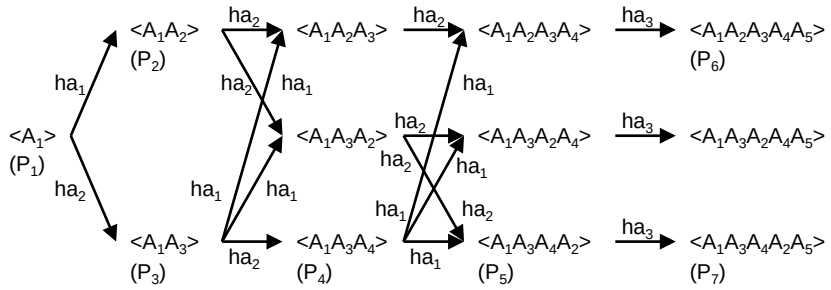

Abbildung 4.3: Ableitungsfolge für die Prozessregeln ha_1, ha_2, ha_3, ha_4

Beispiel. Wir untersuchen die Bearbeitung der Kundenanfrage von Herrn Müller durch Holiday Airways auf Basis von vier Prozessregeln, die wir bereits im letzten Kapitel formuliert haben. Die erste Regel beschreibt den Vorgang zur Ermittlung der nächsten Flugverbindung. Die zweite und dritte Regel beschreiben die Berechnung des Grundpreises beziehungsweise der Steuern und Gebühren für den Flug. Die vierte Regel adressiert schließlich die Zusammenfassung von Preis und Verbindung zu einem Angebot. Wir gehen für den Moment davon aus, dass die Regeln als $ha_1, ha_2, ha_3, ha_4 \in \mathit{Rules}$ definiert sind, und werden uns später mit der Formalisierung dieser Regeln befassen.

Mit den vier Regeln lässt sich die Bearbeitung einer Fluganfrage systematisch entwickeln.

Abbildung 4.3 zeigt alle zulässigen Ableitungen für die vier Prozessregeln ha_1, ha_2, ha_3, ha_4. Die Regeln sind rein kausal aufgebaut, können somit auch zeitlich vorgelagerte Schritte ergänzen, sofern kein kausaler Zusammenhang zu nachgelagerten Schritten existiert. Wir erkennen, dass die Gesamtheit aller Prozesse, die aus der Kundenanfrage P_1 hergeleitet werden können, der Spezifikation für die Bearbeitung der Anfrage S_m entspricht, die wir bereits aufgestellt haben.□

4.2.3.2 Komposition

Bei komplexen Handlungsabläufen führt die elementare Beschreibung einer Regel auf Grund der Fülle von Abhängigkeiten nahezu unvermeidbar zu einem komplizierten, stark verschachtelten Formalismus, der nur schwer handhabbar ist. Es ist daher wünschenswert, eine solch komplexe Regel aus einfachen Regeln sukzessive zusammenzusetzen. Wir betrachten in diesem Abschnitt die Komposition zweier Regeln und unterscheiden dazu drei Formen, die wir anhand von folgenden Operatoren definieren:

- *Alternative Komposition (|)*
  Wenn zwei Regeln alternativ komponiert werden, stellt für eine gegebene Prozessbeobachtung jeder Prozess, der aus einer der beiden Regeln abgeleitet werden kann, ein gültiges Resultat für die komponierte Regel dar.

- *Parallele Komposition ($\|$)*
  Wenn zwei Regeln parallel komponiert werden, generieren sie für eine gegebene Prozessbeobachtung das Supremum der beiden Prozesse, in denen die beiden Regeln für diese Beobachtung resultieren. Wenn die Einzelregeln nicht deterministisch definiert sind, lässt die komponierte Regel jedes Supremum der beiden einzelnen Prozesse als Ergebnis zu.
- *Sequentielle Komposition (;)*
  Wenn zwei Regeln sequentiell komponiert werden, wird eine Prozessbeobachtung zunächst durch die erste Regel erfasst. Das Ergebnis der Anwendung der ersten Regel wird durch die zweite Regel beobachtet und durch diese weiterentwickelt zum Ergebnis der sequentiellen Komposition.

Mit der Komposition können wir eine ganzheitliche Perspektive auf einen komplexen Ablauf gewinnen, indem wir die Gesamtheit aller Abhängigkeiten sukzessive in Form einzelner Regeln formulieren.

Definition 5. Komposition von Prozessregeln

Wir bezeichnen als Komposition von zwei Prozessregeln eine Abbildung

$$|, \|, ; \in \mathit{Rules} \times \mathit{Rules} \to \mathit{Rules}$$

Diese sind für zwei gegebene Regeln r_1, r_2 und drei Prozesse $P, Q, R \in \mathit{Processes}$ wie folgt definiert:

- Alternative Komposition:
  $\frac{P \xrightarrow{r_1} Q}{P \xrightarrow{r_1|r_2} Q}, \frac{P \xrightarrow{r_2} R}{P \xrightarrow{r_1|r_2} R}$
- Parallele Komposition:
  $\frac{P \xrightarrow{r_1} Q, P \xrightarrow{r_2} R}{P \xrightarrow{r_1\|r_2} Q \veebar R}$
  falls $Q \veebar R$ als Supremum der beiden Prozesse existiert
- Sequentielle Komposition:
  $\frac{P \xrightarrow{r_1} Q, Q \xrightarrow{r_2} R}{P \xrightarrow{r_1;r_2} R}$

□

Die Komposition zweier Regeln mit den drei Operatoren stellt wiederum eine Prozessregel nach Definition 4 dar. Alle drei Kompositionsformen erhalten die Teilprozesseigenschaft einer Prozessregel. Bei der alternative Komposition gilt dies, da eine der beiden Regeln zur Anwendung kommt und für diese die Teilprozesseigenschaft gilt. Bei der parallelen Komposition folgt dies aus der Definition des Supremums. Bei der sequentiellen Komposition kann dies gezeigt werden, da die Teilprozessrelation transitiv ist. Die Menge aller Prozessregeln ist somit abgeschlossen bezüglich der Komposition und weist algebraische Eigenschaften auf. Alle drei Operatoren sind

assoziativ. Die n-fache sequentielle Komposition einer Prozessregel r schreiben wir auch als r^n mit folgender rekursiver Definition:

$$r^0 = id$$
$$r^n = r; r^{n-1}$$

Die alternative und parallele Komposition sind zudem kommutativ. Für die alternative beziehungsweise parallele Komposition einer Menge von Prozessregeln $R = \{r_1, r_2, \ldots, r_n\}$ verwenden wir daher auch die folgende abkürzende Notation:

$$|\, R = r_1 \mid r_2 \mid \ldots \mid r_n$$

$$\|\, R = r_1 \parallel r_2 \parallel \ldots \parallel r_n$$

Umgekehrt bezeichnen wir die Menge der Prozessregeln R als eine Dekomposition einer Prozessregel r, wenn $r =|\, R$ beziehungsweise $r =\|\, R$ gilt.

Die eingeführten Basisoperatoren können problemlos um weitere Operatoren wie Schleifen, Rekursionen und Fallunterscheidungen ergänzt werden. Wir fokussieren uns allerdings auf die vorgestellten Operatoren, die für unsere Diskussion ausreichend sind.

Beispiel. Wir komponieren nun die Prozessregeln für die Anfragebearbeitung bei Holiday Airways zu einem Regelsystem. Dazu betrachten wir zwei Varianten der Regelkomposition:

$$ha = (ha_1 \parallel (ha_2; ha_3)); ha_4$$

$$ha' =|\, \{ha_1, ha_2, ha_3, ha_4\}$$

In der ersten Komposition *ha* antizipieren wir die Bearbeitungsreihenfolge, die wir aus den bestehenden Abhängigkeiten ableiten. Die Fluggesellschaft führt in diesem Fall parallel die Ermittlung der nächsten Verbindung und die Preisberechnung durch. Die Preisberechnung setzt sich wiederum aus zwei sequentiellen Schritten, der Grundpreisberechnung und der Berechnung der Steuern und Gebühren zusammen. Die Ergebnisse dieser beiden Teilprozesse werden parallel komponiert, so dass das Ergebnis dieser Komposition den Prozess darstellt, der sowohl die Preis- als auch die Verbindungsermittlung enthält. Dieser wird nun noch sequentiell um die Angebotserstellung ergänzt. Auf diese Weise kann anhand der Komposition direkt aus der Anfrage der Prozess der Buchungsanfrage vollständig abgeleitet werden:

$$P_1 \xrightarrow{ha} P_7$$

In der zweiten Komposition ha' wenden wir die alternative Komposition an. Dies bedeutet, dass in jedem Entwicklungsschritt eine der Regeln zum Einsatz kommt. Durch mehrfache Anwendung der alternativ komponierten Regeln lässt sich auch hier aus der Kundenanfrage die vollständige Bearbeitung systematisch entwickeln:

$$P_1 \xrightarrow{ha'} P_3 \xrightarrow{ha'} P_4 \xrightarrow{ha'} P_5 \xrightarrow{ha'} P_7$$

Auf Grund der zugelassenen Parallelität zwischen Preisberechnung und Verbindungsermittlung lässt die Regel auch weitere Ableitungsfolgen zu. □

Das Beispiel zeigt, dass eine explizit sequentielle und parallel Regelkomposition gegenüber der generischen alternativen Variante dahingehend vorteilhaft ist, dass Abhängigkeiten transitiv erfasst werden können. Jedoch setzt eine solche Komposition voraus, dass die Ablaufreihenfolge zum Definitionszeitpunkt feststeht. In föderalen Systemen, in denen mehrere Akteure miteinander interagieren, ist dies jedoch nicht immer gegeben. Dies gilt insbesondere für Abläufe, die sich gegenseitig beeinflussen. In der alternativen Komposition können Wechselwirkungen durch iterative Anwendung der komponierten Regel systematisch entwickelt werden. Durch die wiederholte Anwendung werden neu entstandene Abhängigkeiten durch die Regeln erkannt und entsprechend aufgelöst. Dieser Vorgang kann fortgesetzt werden, bis ein Prozess vollständig entwickelt ist und die erneute Anwendung der Regel keine erweiterten Prozesse mehr generiert. Die Menge aller Prozesse, die aus einer Prozessbeobachtung durch wiederholte Regelanwendung abgeleitet werden können, stellt einen transitiven Abschluss einer Prozessregel dar.

Definition 6. Transitiver Abschluss einer Prozessregel

Für eine Prozessregel r definieren wir deren transitiven Abschluss als Prozessregel r^* wie folgt:

$$r^* =| \{r^n : n \in \mathbb{N}\}$$

Wir bezeichnen den transitiven Abschluss einer Prozessregel auch als transitive Hülle. □

Die Wertemenge eines transitiven Abschlusses enthält alle Entwicklungsstufen von Prozessen, die in der iterativen Regelanwendung entstehen können. Die Menge ist konvergent, da jede Ableitungskette in einem Fixpunkt bezüglich der Regel resultiert. Wir verstehen dabei unter einem Fixpunkt einen Prozess, der durch Anwendung der Regel nicht mehr weiter entwickelt werden kann.

Die Konvergenz in einem Fixpunkt kann anhand von zwei Eigenschaften begründet werden. Zum einen haben wir Prozessregeln so definiert, dass der beobachtete Prozess grundsätzlich ein Teilprozess des resultierenden Prozesses ist. Zum anderen stellt $(Processes, \preceq)$ eine vollständige Halbordnung dar. Dadurch bildet die wiederholte Anwendung einer Regel eine aufsteigende Kette gemäß der Ordnung. Wir wissen, dass jede Kette ein Supremum besitzt auf Grund der Vollständigkeit der Halbordnung. Dieses Supremum ist für eine iterative Regelanwendung der Fixpunkt. Wir beschreiben die Menge aller Fixpunkte einer Prozessregel r als

$$Fix(r) = \left\{ P \in Dom(r) : P \xrightarrow{r} P \right\}$$

Die Elemente dieser Menge bezeichnen wir auch als vollständig entwickelte Prozesse bezüglich der Regel r. Es gilt $Fix(r) \subseteq Range(r^*)$, da jeder Fixpunkt per Definition wieder im Wertebereich der Regelanwendung liegt. Umgekehrt kann der Wertebereich von r^* aber auch Prozesse enthalten, die noch keine Fixpunkte im Sinne der Regel darstellen und durch erneute Anwendung der Regel r weiter entwickelt werden können.

4.2.3.3 Regelanwendung bei Spezifikationen

Wir haben bis jetzt Regeln nur auf Prozesse angewendet. Allerdings können in analoger Weise auch Spezifikationen auf Basis von Regeln weiterentwickelt werden. In diesem Fall beschreibt

eine Prozessregel eine Menge aller Prozesse, die aus einer Ursprungsmenge von Prozessen abgeleitet werden können. Wir werden Prozessregeln für Spezifikationen in kanonischer Weise aus korrespondierenden Regeln für Prozesse herleiten. Dazu legen wir eine Regel für eine Spezifikation so aus, dass sie die einzelnen Prozesse entlang einer gegebenen Regellogik entwickelt. Wenn sich aus einem Prozess der Ursprungsspezifikation in nicht deterministischer Weise mehrere Prozesse entwickeln lassen, enthält die Ergebnismenge der Regel all diese Prozesse. Betrachten wir einen Prozess als einen Zustand, so ist der Übergang von Prozessregeln für einzelne Prozesse zu den korrespondierenden Regeln für Spezifikationen vergleichbar mit der aus der Automatentheorie bekannten Potenzmengenkonstruktion.

Wir werden zudem auch Kompositionsoperatoren für Spezifikationsregeln einführen. Wir legen die Operatoren dahingehend aus, dass die Regelkomposition für Spezifikationen isomorph zur Komposition der korrespondierenden Regeln für Prozesse durchgeführt werden kann.

Zunächst leiten wir für eine gegebenen Prozessregel eine korrespondierende Regel für eine Spezifikation ab.

Definition 7. Prozessregel (für eine Spezifikation)

Wir definieren für eine gegebene Prozessregel $\xrightarrow{r} \in \mathit{Rules}$ eine korrespondierende Regel für eine Prozessspezifikation als eine Relation

$$\overset{r}{\Longrightarrow} \subseteq \mathit{Spec} \times \mathit{Spec}$$

Diese ist für zwei beliebige Spezifikationen $S, T \in \mathit{Spec}$ wie folgt definiert:

$$S \overset{r}{\Longrightarrow} T \Leftrightarrow T = \left\{ Q : \exists P \in S : P \xrightarrow{r} Q \right\}$$

Die Regel für eine Spezifikation greift somit auf die korrespondierende Regel für Prozesse zurück und entwickelt alle Prozesse in der Ausgangsspezifikation S gemäß der Prozessregel. Ist die Prozessregel nicht deterministisch definiert, leitet die Regel für die Spezifikation die Menge aller Prozesse T ab, die aus einer Menge beobachteter Prozesse S resultieren können. □

Die Menge aller Prozessregeln für Spezifikationen, die auf Basis der vorgestellten Definition hergeleitet werden können, bezeichnen wir im Folgenden auch abkürzend mit *SRules*. Es gilt $\mathit{SRules} \subset \mathscr{P}(\mathit{Spec} \times \mathit{Spec})$, da die Definition 7 nur bestimmte Entwicklungen einer Spezifikation zulässt. Die hier geltenden Einschränkungen werden deutlich, wenn wir folgende drei Eigenschaften einer Spezifikationsregel r nach Definition 7 für vier beliebige Spezifikationen $S_1, S_2, T_1, T_2 \in \mathit{Spec}$ betrachten:

(1) $S_1 \overset{r}{\Longrightarrow} T_1 \wedge S_2 \overset{r}{\Longrightarrow} T_2 \Rightarrow S_1 \cup S_2 \overset{r}{\Longrightarrow} T_1 \cup T_2$ (Additivität)

(2) $S_1 \overset{r}{\Longrightarrow} T_1 \wedge S_2 \overset{r}{\Longrightarrow} T_2 \wedge S_1 \subseteq S_2 \Rightarrow T_1 \subseteq T_2$ (Monotonie)

(3) $S_1 \overset{r}{\Longrightarrow} T_1 \wedge S_1 \overset{r}{\Longrightarrow} T_2 \Rightarrow T_1 = T_2$ (funktionale Relation)

Die Eigenschaften (1) und (2) folgen direkt aus der Definition der Spezifikationsregel. Eigenschaft (3) ist erfüllt, da der Nicht-Determinismus einer Prozessregel durch Aufnahme aller ableitbaren Prozesse in die resultierende Spezifikation im Sinne einer Potenzmengenkonstruktion aufgelöst wird.

In voller Allgemeinheit gibt es Prozessregeln $\overset{r}{\Longrightarrow} \subseteq Spec \times Spec$, die diese Eigenschaften nicht erfüllen und folglich nicht nach Definition 7 formuliert werden können. Dies trifft beispielshalber immer dann zu, wenn eine Spezifikationsregel nicht nur einzelne Prozesse in der Ausgangsspezifikation entwickelt, sondern auch zusätzliche Aktionen aus Prozesskombinationen ableitet. In diesem Fall kann die Additivität nicht zusammen mit einer funktionalen Relation gewährleistet werden. Wir beschränken uns im Folgenden jedoch auf die Betrachtung von Spezifikationsregeln, die sich nach der zuvor vorgestellten Definition formulieren lassen.

Mit dem Theorem von Knaster-Tarski [Tar55] kann man zeigen, dass eine Prozessregel für eine Spezifikation einen kleinsten und größten Fixpunkt hat. Das Theorem ist anwendbar, da die Relation monoton ist und mit $(Spec, \subseteq)$ als vollständigem Verband auf einer Struktur operiert, die eine vollständige Halbordnung bezüglich der Teilmengenrelation beinhaltet und Suprema respektiert. Der kleinste beziehungsweise größte Fixpunkt ist die Spezifikation mit den wenigsten beziehungsweise meisten Prozessen, die die Regel unverändert auf sich selbst abbildet. Der kleinste Fixpunkt kann nach dem Fixpunkttheorem von Kleene [Kle52] iterativ aus der leeren Spezifikation als Nullelement des Verbandes entwickelt werden.

Auch im Kontext von Prozessspezifikationen lassen sich Operatoren für die Regelkomposition definieren. Auf diese Weise können wir analog zu den Prozessen komplexe Entwicklungsschritte auf Basis einfacherer Regeln beschreiben.

Definition 8. Komposition von Prozessregeln (für Spezifikationen)

Wir bezeichnen als Komposition von zwei Prozessregeln für eine Spezifikation eine Abbildung

$$\overline{|}, \overline{\|}, \overline{;} \in SRules \times SRules \to SRules$$

Diese sind für zwei Regeln r_1, r_2, drei Prozesse $P, Q, R \in Processes$ und drei Spezifikationen $S, T, U \in Spec$ wie folgt definiert:

- Alternative Komposition:
  $$\frac{S \overset{r_1}{\Longrightarrow} T, S \overset{r_2}{\Longrightarrow} U}{S \overset{r_1 \overline{|} r_2}{\Longrightarrow} T \cup U}$$
- Parallele Komposition:
  $$\{\} \overset{r_1 \overline{\|} r_2}{\Longrightarrow} \{\}, \quad \frac{\{P\} \overset{r_1}{\Longrightarrow} X, \{P\} \overset{r_2}{\Longrightarrow} Y, S \overset{r_1 \overline{\|} r_2}{\Longrightarrow} T}{\{P\} \cup S \overset{r_1 \overline{\|} r_2}{\Longrightarrow} \{Q \veebar R : Q \in X \wedge R \in Y\} \cup T}$$
  falls $Q \veebar R$ als Supremum der beiden Prozesse existiert
- Sequentielle Komposition:
  $$\frac{S \overset{r_1}{\Longrightarrow} T, T \overset{r_2}{\Longrightarrow} U}{S \overset{c_1 \overline{;} c_2}{\Longrightarrow} U}$$

□

Es besteht ein Isomorphismus zwischen den Regeln für Prozesse und denen für Spezifikationen:

$$(Rules, \{|, \|, ;\}) \cong \left(SRules, \left\{\bar{|}, \bar{\|}, \bar{;}\right\}\right)$$

Die Abbildung von Regeln für Prozesse auf Regeln für Spezifikationen $Rules \to SRules$ ergibt sich unmittelbar aus der Definition 7. Die Abbildung ist bijektiv. Die eine Richtung der Bijektion folgt aus der Definition einer Spezifikationsregel. Die Rückrichtung lässt sich zeigen, da für eine Regel $\overset{r}{\Longrightarrow} \in SRules$ ein beliebiges Element der Relation $S \overset{r}{\Longrightarrow} T$ auf Grund der Additivität und der funktionalen Auslegung der Relation eindeutig aufgespalten werden kann wie folgt:

$$\{P\} \cup \{Q\} \cup \ldots \overset{r}{\Longrightarrow} \{P_1, \ldots, P_m\} \cup \{Q_1, \ldots, Q_n\} \cup \ldots$$

mit $\{P\} \overset{r}{\Longrightarrow} \{P_1, \ldots, P_m\}, \{Q\} \overset{r}{\Longrightarrow} \{Q_1, \ldots, Q_n\}, \ldots$. Daraus lässt sich eine Regel $\overset{r}{\longrightarrow} \in Rules$ ableiten, deren Relation für jedes Element der Regel $\overset{r}{\Longrightarrow}$, das wir in der soeben vorgestellten aufgespaltenen Darstellung betrachten, folgende Elemente enthält:

$$P \overset{r}{\longrightarrow} P_1, \ldots, P \overset{r}{\longrightarrow} P_m, Q \overset{r}{\longrightarrow} Q_1, \ldots, Q \overset{r}{\longrightarrow} Q_n, \ldots$$

Somit können wir jede Regel für Prozesse eineindeutig in eine korrespondierende Regel für Spezifikationen übersetzen. Es bleibt noch zu zeigen, dass die Operatoren die Abbildung respektieren:

(1) $S \overset{r_1 \bar{|} r_2}{\Longrightarrow} T \Leftrightarrow S \overset{r_1 | r_2}{\Longrightarrow} T$

(2) $S \overset{r_1 \bar{\|} r_2}{\Longrightarrow} T \Leftrightarrow S \overset{r_1 \| r_2}{\Longrightarrow} T$

(3) $S \overset{r_1 \bar{;} r_2}{\Longrightarrow} T \Leftrightarrow S \overset{r_1 ; r_2}{\Longrightarrow} T$

Wir zeigen die Äquivalenz exemplarisch für die sequentielle Komposition. Aus $S \overset{r_1 \bar{;} r_2}{\Longrightarrow} U$ lässt sich per Definition der Komposition für Spezifikationen ableiten, dass $S \overset{r_1}{\Longrightarrow} T$ und $T \overset{r_2}{\Longrightarrow} U$ gilt. Wir übersetzen diese beiden Aussagen nun in die Prozesssicht. Damit gilt

$$T = \left\{Q : \exists P \in S : P \overset{r_1}{\longrightarrow} Q\right\}$$

$$U = \left\{R : \exists Q \in T : Q \overset{r_2}{\longrightarrow} R\right\}$$

Daraus lässt sich für die Menge U folgende Definition ableiten:

$$U = \left\{R : \exists P \in S, Q \in T : P \overset{r_1}{\longrightarrow} Q \wedge Q \overset{r_2}{\longrightarrow} R\right\}$$

Per Definition der sequentiellen Komposition für Prozesse entspricht dies der folgenden Menge:

$$U = \left\{R : \exists P \in S : P \overset{r_1 ; r_2}{\longrightarrow} R\right\}$$

Damit haben wir gezeigt, dass $S \overset{r_1 ; r_2}{\Longrightarrow} U$ gilt.

Wir sprechen daher im Folgenden nur noch von Prozessregeln ohne den Zusatz „für Prozesse" beziehungsweise „für Spezifikationen" und wenden diese analog auf Prozesse und Spezifikationen an. Wir gehen dabei davon aus, dass eine Regel $\xrightarrow{r}$ mit der Regel $\overset{r}{\Longrightarrow}$ korrespondiert, die den gleichen Bezeichner aufweist. Ebenso verzichten wir im Folgenden bei den Operatoren auf den Überstrich und wenden $|, \|, ;$ in kongruenter Weise bei Prozessen und Spezifikationen an.

Aus dem Isomorphismus folgt, dass auch für eine Spezifikation $S \in Spec$ und eine Regel r ein transitiver Abschluss $T \in Spec$ gebildet werden kann:

$$S \overset{r^*}{\Longrightarrow} T$$

Die Spezifikation T ist ein Fixpunkt der systematischen Entwicklung einer Spezifikation S entlang der Regel r. Die Spezifikation T enthält alle Prozesse, die auf Basis der Regellogik r aus der Prozessmenge S heraus entwickelt werden können. Betrachten wir S als einen Startzustand und die Regel r als Produktionsregel einer formalen Grammatik, so stellt der Fixpunkt die formale Sprache dar, die aus der Grammatik gebildet werden kann. Wir werden im Folgenden das zulässige Verhalten von Akteuren auf Basis eines Startzustandes, ausgedrückt in einer Menge möglicher Startprozesse, und einer Regellogik als Fixpunkt der Regelentwicklung beschreiben.

Beispiel. Wir kommen zurück zur Preisberechnung von Holiday Airways und betrachten nun den transitiven Abschluss der alternativ komponierten Prozessregel ha', die das Verhalten von Holiday Airways in der gegebenen Buchungsanfrage beschreibt. Wenn wir davon ausgehen, dass die Spezifikation für Kundenanfragen nur P_1 zulässt, können wir die transitive Hülle der Regelbearbeitung herleiten:

$$\{P_1\} \overset{(ha')^*}{\Longrightarrow} S_m$$

Sie entspricht der Spezifikation S_m, die wir vorhin diskutiert haben. Wir können nun die Spezifikation der Kundenanfragen auf die Menge aller zulässigen Anfragen ausweiten. In diesem Fall würde $(ha')^*$ alle zulässigen Ablaufentwicklungen dieser Anfragen entwickeln und es ließe sich so eine vollständige Spezifikation für die Bearbeitung von Buchungsanfragen bei Holiday Airways auf Basis von gegebenen Kundenanfragen definieren.

Das Beispiel zeigt, wie auf Grundlage einer Prozessannahme eine Regellogik systematisch Abläufe, die sich aus der Annahme ergeben, entwickelt. □

4.2.4 Regelbeziehungen

Wir haben bereits kennengelernt, dass sich komplexe Regelsysteme aus einer Vielzahl einzelner Regeln zusammensetzen können. Zur Strukturierung solcher Regelsysteme kann es wünschenswert sein, die Regeln zueinander in Bezug zu stellen.

Definition 9. Regelbeziehung

Wir bezeichnen eine Relation, die eine Aussagen über den Zusammenhang zwischen zwei Prozessregeln trifft, als eine Regelbeziehung:

$$rel \subseteq Rules \times Rules$$

□

Wir werden im Folgenden einige Ausprägungen solcher Regelbeziehungen diskutieren.

4.2.4.1 Verfeinerung

Mit Prozessregeln sowie deren Komposition lassen sich umfassende Abläufe beschreiben. Da das Aufstellen eines entsprechend komplexen Regelwerks allerdings typischerweise mit größeren Aufwänden verbunden ist, kann es von Vorteil sein, zunächst ein einfacheres Regelwerk zu entwickeln und dieses im Anschluss sukzessive zu präzisieren. Wir bezeichnen dieses Vorgehen als Verfeinerung.

Definition 10. Verfeinerung einer Prozessregel

Wir bezeichnen eine Prozessregel als eine Verfeinerung einer anderen Regel, wenn die verfeinerte Regel nur noch eine Teilmenge der Prozessentwicklungen der ursprünglichen Regel enthält. Die Verfeinerung stellt eine Regelbeziehung dar:

$$\rightsquigarrow \subseteq \mathit{Rules} \times \mathit{Rules}$$

Sie ist für zwei Prozessregeln r_1, r_2 wie folgt definiert:

$$\xrightarrow{r_1} \rightsquigarrow \xrightarrow{r_2} \Leftrightarrow \xrightarrow{r_2} \subseteq \xrightarrow{r_1}$$

□

Wir schreiben für zwei Prozessregeln r_1, r_2, die in einer Verfeinerungsbeziehung stehen, auch abkürzend $r_1 \rightsquigarrow r_2$. Die Verfeinerung stellt eine Präzisierung einer Regel dar, da sie die Menge zulässiger Ableitungen einschränkt.

Eine präzise formale Beschreibung einer Prozessregel ist auf Grund des hohen Beschreibungsaufwands nicht immer praktikabel. Regeln können jedoch auch informell formuliert werden. Wann immer Regeln informell beschrieben werden, ist die Menge der anhand der Regel entwickelbaren Prozesse für eine gegebene Prozessbeobachtung weiter auslegbar. Eine informell definierte Prozessregel lässt also in nicht deterministischer Weise für eine bestimmte Prozessbeobachtung eine Vielzahl daraus ableitbarer Prozesse zu. Mit der Verfeinerung einer Regel kann diese Menge zunehmend eingeschränkt werden bis hin zu einer formalen Betrachtung.

Die informelle Regeldefinition kann in unterschiedlicher Präzision erfolgen. Diese reicht von einer textuell deskriptiven Darstellung des Zusammenhangs zwischen beobachteten und resultierenden Handlungen bis hin zu einer detaillierten Vorgangsbeschreibung unter Einbeziehung formaler Vor- und Nachbedingungen. In der Praxis werden Regeldefinitionen häufig auf einer einfachen deskriptiven Ebene aufgesetzt und dann sukzessive präzisiert.

4.2.4.2 Sichten

In der Beschreibung komplexer Zusammenhänge auf Basis von Prozessregeln kann es von Vorteil oder sogar erforderlich sein, Regeln aus verschiedenen Sichten zu betrachten. Ersteres ermöglicht beispielsweise in einer arbeitsteilige Konstellation die Erfassung der Beiträge der einzelnen Akteure zu einem Gesamtablauf. Letzteres ist zum Beispiel dann der Fall, wenn die an den Abläufen beteiligten Akteure unterschiedliche Auslegungen des Prozessbegriffs anwenden.

Definition 11. Sicht auf einen Prozess

Wir verstehen unter einer Sicht eine Übersetzung eines Prozesses in eine andere Betrachtungsweise:

$$\xi \in Processes \rightarrow Processes$$

□

Wir bezeichnen die Menge aller Sichten als $Views = Processes \rightarrow Processes$. Eine Sicht $\xi \in Views$ kann in analoger Weise auch auf Spezifikationen angewendet werden $\xi \in Spec \rightarrow Spec$, wobei für eine Spezifikation $S \in Spec$ dann gilt:

$$\xi(S) = \{\xi(P) : P \in S\}$$

Betrachten wir Prozessregeln im Kontext von Sichten, so ist es wünschenswert, auch Regeln dahingehend zu modifizieren, dass sie in einer gegebenen Sicht anwendbar sind. Dazu führen wir eine Regeltransformation durch. Eine Regeltransformation ist eine besondere Form der Regelbeziehung $rel \in Rules \rightarrow Rules$, die als funktionale Relation einer gegebenen Regel eindeutig eine andere Regel zuweist. Die Menge aller Regeltransformationen bezeichnen wir als $Trans = Rules \rightarrow Rules$ Regeltransformationen stellen eine Übersetzung einer Prozessregel dar. Im Zusammenhang mit Sichten können Transformationen homomorph durchgeführt werden.

Definition 12. Homomorphe Regeltransformation

Wir bezeichnen eine Regeltransformation $t_\xi \in Trans$ als homomorph zu einer Sicht $\xi \in Views$, falls für eine beliebige Prozessregel r gilt:

$$P \xrightarrow{r} Q \Leftrightarrow \xi(P) \xrightarrow{t_\xi(r)} \xi(Q)$$

Für den Definitions- und Wertebereich der Transformation gilt dabei $Dom\left(t_\xi(r)\right) = \xi(Dom(r))$ und $Range\left(t_\xi(r)\right) = \xi(Range(r))$. Eine homomorphe Regeltransformation realisiert folglich einen Homomorphismus zwischen $(Processes, r)$ und $\left(Processes, t_\xi(r)\right)$ anhand der Sicht ξ. □

Homomorphe Regeltransformationen gewährleisten, dass Prozessregeln im Kontext einer Sicht die gleichen Zusammenhänge zwischen Prozessen herstellen wie im Urbild. Durch die Umsetzung von homomorph abgebildeten Regellogiken lassen sich somit auch Verhaltensweisen konsistent in Umfeldern beschreiben, die durch unterschiedliche Sichtweisen auf Abläufe geprägt sind.

Beispiel. In den Buchungsanfragen von Frau Müller und Herrn Huber bei Holiday Airways haben wir bislang die dreistelligen Flughafenkürzel „MUC" für München und „FRA" für Frankfurt verwendet. Den Kunden sind diese Abkürzungen aber nicht unbedingt geläufig, wir definieren daher eine Kundensicht auf den Prozess der Flugbuchung, in der die ausgeschriebenen Flughafennamen verwendet werden.

Die Definition der Sicht ist in diesem Fall einfach die Übersetzung eines Prozesses, in dem Kürzel für die Referenzierung von Flughäfen eingesetzt werden, in einen Prozess, der unverändert ist, bis auf dass er die ausgeschriebenen Flughafenbezeichnungen enthält. Entsprechend kann beispielsweise die Regel für das System der Grundpreisberechnung übersetzt werden. In der übersetzten Variante erwartet die Regel eine Anfrage mit den ausgeschriebenen Flughafenbezeichnungen anstatt der Kürzel. Die Transformation der Regel ist homomorph, wenn die Preisberechnung für die gleiche Strecke, in der lediglich ausgeschriebene Namen statt Kürzel verwendet werden, den gleichen Preis ermittelt. □

4.2.5 Zusammenfassung des abstrakten Prozessmodells

Wir haben in den letzten Abschnitten ein abstraktes Grundmodell eingeführt und anhand dessen Eigenschaften der Prozessbeschreibung unabhängig von einer bestimmten Prozessrepräsentation diskutiert. Ein Prozess stellt ein bestimmtes Entwicklungsstadium eines Ablaufs dar, das auch als Zustand des Ablaufs angesehen werden kann. Mit Prozessregeln treffen wir Aussagen über die Entwicklung von Abläufen, indem wir einen Bezug zwischen einem Prozess und einem fortgeführten Prozess mit zusätzlichen Handlungen herstellen. Auf diese Weise beschreiben wir Ursache-/Wirkungszusammenhänge von bestimmten Handlungen in Abläufen. Komplexe Regeln können durch Komposition aus einzelnen Regeln zusammengesetzt werden.

Unter einer Spezifikation verstehen wir eine Menge von Prozessen. Spezifikationen können in analoger Weise anhand von Prozessregeln weiterentwickelt werden, indem die Regeln auf die einzelnen Prozesse in der Spezifikation angewendet werden. Wir haben gezeigt, dass das Kompositionsverhalten solcher Regeln für Prozesse und Spezifikationen isomorph ist, und wir folglich Regelzusammenhänge auf Prozess- und Spezifikationsebene in gleicher Weise untersuchen können. Wir betrachten eine Spezifikation als eine Menge von Prozessen, die semantisch äquivalent sind bezüglich einer Aussage. In Prozessregeln kann eine solche Aussage in strukturierter Form formuliert und damit eine Prozessmenge systematisch entwickelt werden.

Zum Schluss haben wir uns mit Beziehungen und Sichten befasst. So können wir Verfeinerungsbeziehungen zwischen Prozessregeln herstellen und somit schrittweise Verhaltensbeschreibungen präzisieren. Ferner lassen sich mit Sichten unterschiedliche Perspektiven auf Prozesse, beispielsweise von unterschiedlichen Akteuren, abbilden. Durch eine homomorphe Übersetzung der korrespondierenden Regeln gewährleisten wir, dass die Verhaltensbeschreibung auch in der Abbildung der Logik der ursprünglichen Sicht entspricht.

4.3 Konkretes Prozessmodell

Wir führen nun eine formale Repräsentation für die bisher abstrakt diskutierten Prozesse ein, damit wir konkrete Abläufe beschreiben können. Dabei fokussieren wir weiterhin zunächst ausschließlich auf Prozesse und Spezifikationen sowie deren regelbasierte Entwicklung. Weitere Sichten auf einen Prozess, die in der geschäftlichen Auslegung im Sinne eines Geschäftsprozesses von Interesse sind, werden wir im nächsten Abschnitt untersuchen.

4.3.1 Ereignisse

Es gibt eine Reihe unterschiedlicher Repräsentationsmöglichkeiten von Prozessen. Wir wenden in unserem Modell eine ereignisbasierte Beschreibung von Abläufen an. Damit fassen wir einen Prozess als eine Menge von beobachteten Ereignissen auf. Die ereignisorientierte Betrachtung erweist sich im Kontext dieser Arbeit als vorteilhaft, da Ereignisse auf Grund ihrer feingranularen Natur eine hohe Flexibilität in der Beschreibung zulassen und damit im weiteren Verlauf die intuitive Einführung unterschiedlicher Sichten auf einen Prozess ermöglichen.

Gleichzeitig lässt sich mit Ereignissen der enge Bezug zur theoretischen Informatik aufrecht erhalten, indem wir Ereignisse als einer Art Alphabet einer formalen Sprachen ansehen, die wir regelbasiert entwickeln. Ein Prozess kann dabei als Repräsentation eines Zustandes angesehen werden, der sich aus der Beobachtung der in ihm enthaltenen Ereignisse ergibt. Die Regeln sind grundsätzlich so aufgebaut, dass sie Prozesse fortsetzen durch Ergänzung zusätzlicher Ereignisse. Die zu ergänzenden Ereignisse werden definiert in Abhängigkeit der bereits enthaltenen Ereignisse. Auf diese Weise kann mit Prozessregeln das Verhalten eines Akteurs beschrieben werden, der auf beobachtete Ereignisse mit der Erzeugung weiterer Ereignisse reagiert. Wir unterscheiden dabei nicht zwischen terminalen und nicht terminalen Symbolen.

Wir führen das Konzept des Ereignisses zusammen mit dem Konzept der Aktion ein. Wenn wir mit Ereignissen Abläufe beschreiben, kann es vorkommen, dass die gleiche Handlung öfters durchgeführt wird. Wir differenzieren daher zwischen der Beschreibung des Schemas einer Handlung (Aktion) und der Instanz einer Handlung (Ereignis).

Definition 13. Aktion

Wir definieren eine Aktion als eine Handlung, die mehrfach durchgeführt werden kann. □

Der Begriff Aktion kann dabei prinzipiell im weitesten Sinne ausgelegt werden als jede Form einer Tätigkeit. Aktionen können sowohl von Personen als auch Maschinen ausgeführt werden. Im letzten Fall stellen Aktionen eine Form der Datenverarbeitung oder -übermittlung dar. Wir gehen davon aus, dass Holiday Airways Buchungsanfragen rechnergestützt bearbeitet und fokussieren uns daher auf die Betrachtung von datenbezogenen Aktionen.

Wir bezeichnen die Menge aller Aktionen als *Actions*. Die strukturierte Beschreibung der in dieser Menge enthaltenen Aktionen kann entlang einer formalen Grammatik, beispielsweise in Bakus-Naur-Form, erfolgen. Da wir aber im Folgenden mit einem verhältnismäßig einfachen

Aktionenmodell argumentieren werden, verzichten wir auf die Definition einer vollständigen Grammatik und stellen Aktionen in folgender Syntax dar:

$$Identifier(arg_1, arg_2, \ldots) \in Actions$$

Wir referenzieren Aktionen mit einem Bezeichner *Identifier* und fügen optional Parameter $arg_1, arg_2, \ldots$ hinzu. Ein Ereignis repräsentiert eine Instanz einer Aktion.

Definition 14. Ereignis

Mit einem Ereignis erfassen wir eine bestimmte Instanz einer Aktion. Wir ordnen einem Ereignis e dabei die Aktion $a \in Actions$, die in dem Ereignis ausgeführt wird, als Attribut zu:

$$e.a$$

□

Wir bezeichnen die Menge aller Ereignisse als *Events*. Auf Basis von Ereignissen können wir somit Abläufe, die sich aus Aktionen zusammensetzen, formal darstellen. Ein Ablauf kann dabei mehrere Ereignisse enthalten, in denen die gleiche Aktion ausgeführt wird. Das Ereignis repräsentiert die einmalige konkrete Ausführung einer Handlung, während die Aktion die Handlung losgelöst von einer bestimmten Durchführung beschreibt.

Beispiel. Wir formalisieren nun die Ablaufbeschreibung von Buchungsanfragen bei Holiday Airways anhand von Ereignissen. Die Abbildung 4.4 zeigt die Anfragebearbeitung auf Basis einer gegebenen Kundenanfrage seitens der Fluggesellschaft in vier teils parallel ausführbaren Schritten, die wir bereits zuvor diskutiert haben. Wir gehen davon aus, dass zum Abschluss eines jeden Bearbeitungsschrittes das produzierte Ergebnis zur weiteren Bearbeitung übergeben wird. Im Fall der Verbindungsermittlung sowie der Preisberechnung handelt es sich dabei um eine interne Übergabe, im Fall der Angebotserstellung verstehen wir darunter die Übermittlung des Angebots an den Kunden.

Wir erfassen die vier Schritte zunächst formal anhand der Übergabe der jeweiligen Teilergebnisse und definieren dazu vier mit den Bearbeitungsschritten korrespondierende Aktionen:

(1) $Flight(\langle Date \rangle)$

(2) $Fare(\langle Value \rangle)$

(3) $AmtDue(\langle Value \rangle)$

(4) $Offer(\langle Date \rangle, \langle AmtDue \rangle)$

Damit abstrahieren wir vorerst von den Vorgängen zur Ermittlung dieser Ergebnisse. Zudem definieren wir eine weitere Aktion, die die Übermittlung einer Kundenanfrage repräsentiert:

(0) $Inq(\langle Orig \rangle, \langle Dest \rangle, \langle Date \rangle)$

Die Aktion (0) steht für eine Kundenanfrage für den ersten Flug ab dem Zeitpunkt $\langle Date \rangle$ auf der Strecke von $\langle Orig \rangle$ nach $\langle Dest \rangle$. Aktion (1) repräsentiert den Zeitpunkt der nächsten verfügbaren Flugverbindung $\langle Date \rangle$. Aktion (2) steht für den ermittelten Grundpreis für die angefragte

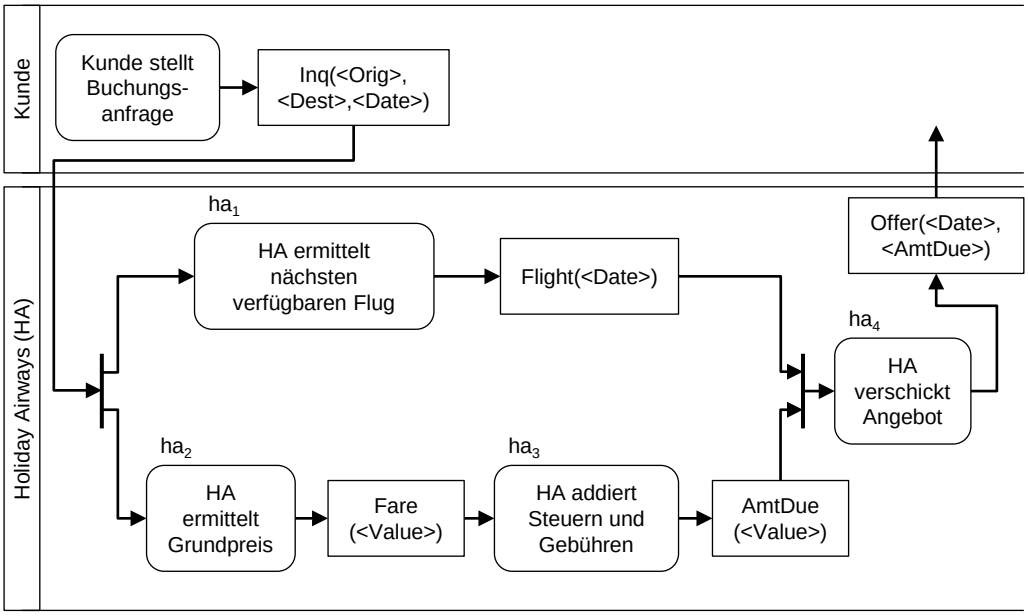

Abbildung 4.4: Ablaufbeschreibung einer Buchungsanfrage auf Basis von fünf Aktionen

Ereignis	$e.a$
e_{11}	$Inq\,(\text{MUC}, \text{FRA}, 14.01.\#12{:}00)$
e_{12}	$Flight\,(14.01.\#12{:}15)$
e_{13}	$Fare\,(170)$
e_{14}	$AmtDue\,(221)$
e_{15}	$Offer\,(14.01.\#12{:}15, 221)$

Tabelle 4.1: Ereignisse für eine exemplarische Buchungsanfrage bei Holiday Airways

Strecke und Aktion (3) entsprechend für den Gesamtpreis einschließlich Steuern und Gebühren. Mit Aktion (4) wird schließlich das Angebot, bestehend aus Datum $\langle Date \rangle$ und Gesamtpreis $\langle AmtDue \rangle$, an den Kunden versendet.

Jede dieser Aktionen kann nun in beliebigen Ereignissen ausgeführt werden. Wir untersuchen vorerst einen exemplarischen Ablauf der Buchungsanfrage von Herrn Müller, in dem die fünf Aktionen jeweils einmal auftreten. Dies resultiert in den fünf in Tabelle 4.1 aufgeführten Ereignissen. □

4.3.2 Ereignisbasierte Prozessrepräsentation

Wir beschreiben nun Prozesse auf Basis von Ereignissen. Bei der Zusammenstellung von Ereignissen zu einem Prozess ist von Interesse, in welchem zeitlichen Zusammenhang die Ereignisse

stehen. Wir gehen davon aus, dass jedes Ereignis zu einem diskreten Zeitpunkt stattfindet. Dies stellt eine Vereinfachung des Modells dar. Der allgemeine Fall, dass sich Ereignisse auch über Zeiträume erstrecken können, kann allerdings problemlos durch eine Erweiterung des Modells integriert werden. Dazu müssen lediglich zwei Zeitpunkte für den Start und das Ende eines Ereignisses erfasst werden.

Es liegt nun in der Natur der Dinge, dass Ereignisse sowohl zeitgleich, als auch versetzt auftreten können. Wir sprechen von zeitgleich, wenn sie zum gleichen diskreten Zeitpunkt eintreten. Im allgemeinen Fall würden dies gelten, wenn sich die Zeiträume zwischen Start- und Endzeitpunkt der beiden Ereignisse überschneiden. Wenn wir Aussagen zu den Zeitpunkten treffen, zu denen Ereignisse auftreten, sprechen wir von einer temporalen Ordnung der Ereignisse.

Wir betrachten drei verschiedene ereignisbasierte Repräsentationen von Prozessen, denen unterschiedliche temporale Ordnungsvorstellungen zugrundeliegen:

- *Ungeordnete Prozesse*
  Im einfachsten Fall treffen wir keine Aussage zu den Zeitpunkten, zu den Ereignisse auftreten können. Ein Prozess stellt damit eine Menge von Ereignissen dar, die in beliebiger Reihenfolge vorkommen dürfen.
- *Kausale Prozesse*
  Den kausalen Prozessen liegt ein relatives Zeitmodell zugrunde. In kausalen Prozessen können auf Basis einer Halbordnung Vorgaben zur Reihenfolge gemacht werden, in der Ereignisse auftreten dürfen. Ereignisse, die nicht durch die Ordnungsrelation erfasst werden, können in beliebiger Reihenfolge auftreten. Das Zeitmodell in kausalen Prozessen trifft lediglich relative Aussagen, es gibt keine konkreten Zeitpunkte für die Ereignisse vor.
- *Temporale Prozesse*
  Ein temporaler Prozess ist eine Menge von Ereignissen, bei der jedem Ereignis ein konkreter Ausführungszeitpunkt zugeordnet ist und somit ein quantitatives Zeitmodell angewendet wird. Daraus ergibt sich eine temporale Ordnung der Ereignisse.

Die drei Repräsentationen eines Prozesses stellen unterschiedliche Abstraktionsstufen dar. Die Beschreibung eines temporalen Prozesses folgt der natürlich Intuition, dass sich ein Beobachter an einen bestimmten Ort stellt und Ereignisse, die für ihn sichtbar sind, mit ihren Auftrittszeitpunkten erfasst.

Ein kausaler Prozess stellt dahingehend eine Abstraktion dar, dass nur noch relative Aussagen zu den Eintrittszeitpunkten der Ereignisse getroffen werden. Ein kausaler Prozess repräsentiert folglich eine Menge temporaler Prozesse, in denen die relative zeitliche Ordnung des kausalen Prozesses zur Anwendung kommt. Kausale Prozesse eignen sich gut zur Darstellung von Ursache-/Wirkungszusammenhängen, also Kausalitäten, in Prozessen. Wenn eine Kausalität zwischen zwei Ereignissen besteht, setzen wir grundsätzlich voraus, dass die Ursache zeitlich vor der Wirkung eintritt. Über die Ordnungsrelation eines kausalen Prozesses kann eine solche zeitliche Abhängigkeit adäquat abgebildet werden.

Grundsätzlich sind auch Mischungen aus temporalen und kausalen Prozessen denkbar. So kann die relative Ordnung eines kausalen Prozesses dahingehend weiter konkretisiert werden, dass

Anforderungen bezüglich des zeitlichen Abstandes zwischen Ereignissen gestellt werden, ohne dass aber konkrete Zeitpunkte für die Ereignisse vorgegeben werden. So ließe sich beispielsweise für unser Beispiel einer Buchungsanfrage fordern, dass Holiday Airways jede Anfrage innerhalb von einer Minute zu bearbeiten hat. Wir werden die weitere Diskussion jedoch auf rein temporale beziehungsweise rein kausale Prozesse fokussieren.

Ein ungeordneter Prozess stellt schließlich eine weitere Abstraktion eines kausalen Prozesses dar, indem nun jede Ordnung zugelassen ist. Ein ungeordneter Prozess entspricht damit der Menge aller kausalen Prozesse, die die gleichen Ereignisse ungeachtet ihrer Ordnung enthalten.

4.3.2.1 Ungeordnete Prozesse und Spezifikationen

Zunächst untersuchen wir zeitlich ungeordnete Prozesse als allgemeinste und damit einfachste Form der Prozessrepräsentation.

Definition 15. Ungeordneter Prozess

Wir definieren einen (zeitlich) ungeordneten Prozess als einen Ablauf, der auf Basis von Ereignissen beschrieben wird und keine temporale Aussage zu den Ereignissen trifft. Formal beschreiben wir einen ungeordneten Prozess als eine Ereignismenge:

$$P \subseteq Events$$

□

Die Menge der ungeordneten Prozesse bezeichnen wir als $Processes_N = \mathscr{P}(Events)$. Sie stellt eine Repräsentation unseres abstrakten Prozessbegriffs dar:

$$(Processes_N, \subseteq)$$

Die auf dieser Menge definierte Ordnung ist die Teilmengenrelation bezüglich der im Prozess enthaltenen Ereignisse. Sie ist damit nicht zu verwechseln mit einer temporalen Ordnung zwischen den Ereignissen, die im Fall der ungeordneten Prozesse nicht existiert. Wir bezeichnen einen Prozess $P' \in Processes_N$ als Teilprozess eines anderen Prozesses $P \in Processes_N$, wenn dieser eine Teilmenge der Ereignisse des anderen Prozesses enthält:

$$P' \subseteq P$$

Die Menge der ungeordneten Prozesse $(Processes_N, \subseteq)$ bildet einen vollständigen Verband, der sich direkt aus der Potenzmengeneigenschaft ableiten lässt. Folglich ist $Processes_N$ eine Halbordnung. Die Eigenschaften der Reflexivität, Antisymmetrie und Transitivität folgen aus der Teilmengendefinition. Das Nullelement des Verbandes ist der Prozess, der keine Ereignisse enthält $\emptyset$, das Einselement die Menge aller Ereignisse $Events$. Im Verband lässt sich für jede Teilmenge von Prozessen ein Supremum konstruieren, das durch Vereinigung der in den Prozessen enthaltenen Ereignisse gebildet wird.

Ein Prozess kann sowohl eine endliche als auch eine unendliche Menge von Ereignissen enthalten. Wir sprechen von endlichen beziehungsweise unendlichen Prozessen.

Das Konzept der Spezifikation als Menge von Prozessen mit einer zugrundeliegenden semantischen Aussage lässt sich direkt bei ungeordneten Prozessen anwenden. Eine Spezifikation stellt hier Mengen von Ereignismengen dar. Die Prozessmengen sind dabei vergleichbar mit den Wortmengen einer formalen Sprache, wobei ungeordnete Prozesse nicht konkateniert, sondern vereinigt werden.

Beispiel. Wir formalisieren die Spezifikation für den Teilprozess der Anfragebearbeitung bei Holiday Airways, in dem für eine Kundenanfrage die nächstmögliche Verbindung ermittelt wird, zunächst in einer vereinfachten Version:

$$S = \mathscr{P}(\{\{e_1, e_2\} : e_1.a = Inq(\langle Orig\rangle, \langle Dest\rangle, \langle rDate\rangle) \wedge e_2.a = Flight(\langle Date\rangle) \wedge next\})$$

Wir gehen dabei davon aus, dass *next* ein Prädikat ist, das auf Basis der gegebenen Buchungslage für eine Fluganfrage zur Strecke von $\langle Orig\rangle$ nach $\langle Dest\rangle$ am Tag $\langle rDate\rangle$ die nächstmögliche Verbindung $\langle Date\rangle$ ermittelt. Die Spezifikation verknüpft also Kundenanfragen mit dazu passenden Flugverbindungen durch Holiday Airways. Zudem ist auch jede Teilmenge als Entwicklungsstadium in der Spezifikation enthalten. In Ermangelung eines Zeitbegriffs können in einer Spezifikation von ungeordneten Prozessen keine Kausalitätsaussagen zwischen den Ereignispaaren getroffen werden. Somit kann der Spezifikation folgend auch die Kundenanfrage aus der Reaktion der Fluggesellschaft heraus entwickelt werden. □

Wir sehen an der Diskussion des Beispiels, dass sich Spezifikationen auf Grundlage ungeordneter Prozesse zwar prinzipiell eignen, um Zusammenhänge zwischen Ereignissen zu erfassen, in ihrer semantischen Aussagekraft jedoch begrenzt sind, da keine gerichteten Abhängigkeiten zwischen den Ereignissen untersucht werden können. Ungeordnete Prozesse dienen somit insbesondere für die Darstellung einfacher Zusammenhänge, sind aber nicht für die systematische Herleitung komplexer Ablauffolgen mit nicht trivialen Abhängigkeiten anwendbar.

4.3.2.2 Kausale Prozesse und Spezifikationen

Um zeitliche Abhängigkeiten zwischen Ereignissen in Prozessen adäquat in unserem Modell abbilden zu können, führen wir als zweite Form der Prozessrepräsentation kausale Prozesse ein. In unserer Definition und Auslegung eines kausalen Prozesses orientieren wir uns eng an [Bro98].

Definition 16. Kausaler Prozess

Wir definieren einen kausalen Prozess als einen Ablauf, der auf Basis von Ereignissen beschrieben wird und in dem Aussagen zur Auftrittsreihenfolge von Ereignissen getroffen werden. Formal beschreiben wir einen kausalen Prozess als $P = (E, \leq)$ mit

$E \subseteq Events$	Eine Menge von Ereignissen, die im Prozess vorkommen
$\leq \subseteq E \times E$	Eine Halbordnung über den Ereignissen, die die zeitliche Reihenfolge der Ereignisse definiert

□

Wir bezeichnen $\leq$ auch als temporale Ordnung eines Prozesses. Stehen zwei Ereignisse $a, b \in Events$ in temporaler Ordnung $a \leq b$, so darf Ereignis b nicht vor Ereignis a auftreten. Wenn keine Ordnungsaussage zu zwei Ereignissen getroffen wird, dürfen diese in beliebiger Reihenfolge eintreten. Die Menge der kausalen Prozesse bezeichnen wir als $Processes_K$. Sie stellt eine Repräsentation unseres abstrakten Prozessbegriffs dar:

$$(Processes_K, \preceq)$$

Die Entwicklungsordnung $\preceq$ der kausalen Prozesse ist dabei als Teilprozessrelation für $P_1, P_2 \in Processes_K$ mit $P_1 = (E_1, \leq_1), P_2 = (E_2, \leq_2)$ wie folgt definiert:

$$P_1 \preceq P_2 \Leftrightarrow E_1 \subseteq E_2 \wedge \leq_1 = (\leq_2 \cap (E_1 \times E_1))$$

Ein kausaler Prozess $P_1 \in Processes_K$ ist somit ein Teilprozess eines anderen Prozesses $P_2 \in Processes_K$ mit $P_1 \preceq P_2$, wenn P' eine Teilmenge der Ereignisse des Prozesses P enthält und die im Teilprozess P' vorkommenden Ereignisse die gleichen Ordnungsbeziehungen aufweisen wie im vollständigen Prozess P. Die Teilprozessrelation kann weiter verfeinert werden zu einer Präfixrelation $\sqsubseteq$ mit folgender Definition:

$$P_1 \sqsubseteq P_2 \Leftrightarrow P_1 \preceq P_2 \wedge \leq_2 \cap (E_2 \backslash E_1 \times E_1) = \emptyset$$

Ein Präfix eines Prozesses beschränkt die Betrachtung auf den Anfang des Prozesses. Wir fordern für ein Präfix, dass der Prozess, aus dem es abgeleitet wird, keine nach der temporalen Ordnung zeitlich vorgelagerten Ereignisse enthält. Präfixe sind für die Untersuchung eines Prozessablaufs von hervorgehobenem Interesse. Da Kausalitäten zwischen Ereignissen stets voraussetzen, dass die Ursache der Wirkung zeitlich vorgelagert ist, lassen sich durch die Betrachtung der Prozesspräfixe die einzelnen Entwicklungsschritte eines kausalen Prozesses systematisch erfassen. Wir werden im weiteren Verlauf der Diskussion auf diesen Aspekt zurückkommen.

Die Menge der kausalen Prozesse $(Processes_K, \preceq)$ bildet eine vollständige Halbordnung. Die Eigenschaften der Reflexivität, Antisymmetrie und Transitivität folgen direkt aus der Definition der Präfixordnung. Die Ordnung besitzt mit dem Prozess $(\emptyset, \emptyset)$ ein Nullelement, aber kein Einselement, da auf Basis des relativen Zeitmodells in einem kausalen Prozess unterschiedliche Abhängigkeiten zwischen den gleichen Ereignissen festgelegt werden können. Die Ordnung ist vollständig, da jede aufsteigende Kette bezüglich der Ereignismenge konvergiert, im maximalen Fall gegen $Events$.

Für zwei kausale Prozesse $P_1, P_2 \in Processes_K$ mit $P_1 = (E_1, \leq_1), P_2 = (E_2, \leq_2)$ kennzeichnen wir deren Infimum bezüglich der Entwicklungsordnung für kausale Prozesse $\preceq$ als $P_1 \barwedge P_2$. Die beiden Prozesse besitzen ein Supremum bezüglich dieser Ordnung, wenn sie über ein Infimum $P = (E, \leq)$ verfügen, so dass gilt $E_1 \cap E_2 = E$. Das Supremum ist dann definiert als:

$$P_1 \veebar P_2 = (E_1 \cup E_2, \leq_1 \cup \leq_2)$$

Mit der Forderung nach Disjunktheit der Ereignisse jenseits des Infimums gewährleisten wir, dass in der Vereinigung keine gegensätzlichen temporalen Ordnungsbeziehungen aufeinander treffen, die zu einem Widerspruch im Zeitmodell führen würden.

Definition 17. Sequentieller Prozess

Wir bezeichnen einen kausalen Prozess $(E, \leq) \in \mathit{Processes}_K$ als einen sequentiellen Prozess, wenn $\leq$ eine totale Ordnung ist. □

Die Menge aller sequentiellen Prozesse geben wir mit $\mathit{Processes}_{KS}$ an. Es handelt sich dabei um eine echte Teilmenge von $\mathit{Processes}_K$. Ein kausaler Prozess kann sequentialisiert werden, indem seine Ordnung zu einer totalen Ordnung ergänzt wird. Die Ergänzung ist nicht eindeutig, insofern können jedem kausalen Prozess eine Menge sequentialisierter Prozesse zugeordnet werden. Wir verwenden dazu eine Funktion $\mathit{Seq} \in \mathit{Processes}_K \to \mathscr{P}(\mathit{Processes}_{KS})$ mit folgender Definition für einen Prozess $(E, \leq) \in \mathit{Processes}_K$:

$$\mathit{Seq}((E, \leq)) = \{(E, \leq_s) \in \mathit{Processes}_{KS} : \leq \subseteq \leq_s\}$$

Für sequentielle Prozesse können Spuren angegeben werden. Spuren stellen die einfachste Form der semantischen Prozessbeschreibung (Trace Semantics) dar. [Gla90] Eine Spur ist die Konkatenation der Aktionen aller Ereignisse eines Prozesses in der Reihenfolge, die die totale Ordnung vorgibt. Nach [Bro98] kann eine Funktion $\mathit{Trace} \in \mathit{Processes}_{KS} \to \mathit{Actions}^{\omega}$ definiert werden, die einen sequentiellen Prozess in eine Spur umwandelt:

$$\mathit{Trace}((E, \leq)) = e_1.a \circ e_2.a \circ \ldots : e_1, e_2, \ldots \in E \wedge e_1 \leq e_2 \leq \ldots$$

Der leere Prozess resultiert dabei in einer leeren Spur: $\mathit{Trace}((\emptyset, \emptyset)) = \varepsilon$. Wir verwenden im Folgenden zudem eine Funktion $\mathit{Traces} \in \mathit{Processes}_K \to \mathscr{P}(\mathit{Actions}^{\omega})$, die für einen beliebigen kausalen Prozess eine Menge von Spuren angibt. Es handelt sich dabei um die Spuren aller sequentialisierten Prozesse, die aus dem kausalen Prozess abgeleitet werden können.

Beispiel. Wir modellieren die vorhin diskutierten Ereignisse für die Buchungsanfrage eines Kunden bei Holiday Airways als kausalen Prozess. Es gilt:

$$P_k = (\{e_{11}, e_{12}, e_{13}, e_{14}, e_{15}\}, \{(e_{11}, e_{12}), (e_{11}, e_{13}), (e_{13}, e_{14}), (e_{12}, e_{15}), (e_{14}, e_{15})\})$$

Kausale Prozesse können anschaulich in Hasse Diagrammen visualisiert werden (vgl. [DP02]). Die Abbildung 4.5 zeigt in Form eines Graphen die bestehenden temporalen Abhängigkeiten zwischen den Ereignissen. Wir erkennen unmittelbar, dass die Verfügbarkeitsüberprüfung und die Preisberechnung jeweils von der Kundenanfrage abhängen und parallel durchgeführt werden. Aus diesen beiden Teilabläufen leitet sich wiederum die Erstellung des Kundenangebots ab.

Der Prozess hat drei verschiedene Sequentialisierungen und damit verbunden drei Spuren, die als mögliche sequentielle Bearbeitungsfolgen des Prozesses in Frage kommen:

- $\mathit{Inq}(\text{MUC}, \text{FRA}, 14.01.\#12{:}00) \circ \mathit{Flight}(14.01.\#12{:}15) \circ \mathit{Fare}(170)$ $\circ \mathit{AmtDue}(221) \circ \mathit{Offer}(14.01.\#12{:}15, 221)$
- $\mathit{Inq}(\text{MUC}, \text{FRA}, 14.01.\#12{:}00) \circ \mathit{Fare}(170) \circ \mathit{Flight}(14.01.\#12{:}15)$ $\circ \mathit{AmtDue}(221) \circ \mathit{Offer}(14.01.\#12{:}15, 221)$

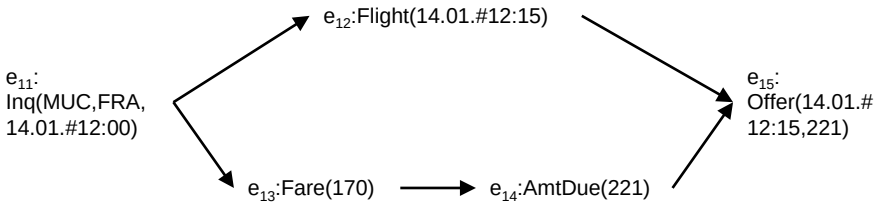

Abbildung 4.5: Buchungsanfrage, dargestellt als kausaler Prozess

- $Inq\,(\text{MUC}, \text{FRA}, 14.01.\#12{:}00) \circ Fare\,(170) \circ AmtDue\,(221)$
  $\circ Flight\,(14.01.\#12{:}15) \circ Offer\,(14.01.\#12{:}15, 221)$ □

Auch für kausale Prozesse können Spezifikationen als Prozessmengen aufgestellt werden, um semantische Aussagen zu den Prozessen zu treffen. Ein komplexer kausaler Prozess kann allerdings eine Vielzahl von Ereignissen und Abhängigkeiten zwischen diesen enthalten. Es ist daher nur bedingt praktikabel, eine Spezifikation direkt anzugeben. Die systematische Entwicklung einer Spezifikation kann stattdessen unter anderem auf Basis eines formalen Automaten erfolgen. Die Spezifikation lässt sich in diesem Fall als operationelle Semantik verstehen. Wir definieren einen Automaten als Tripel

$$(St, s_0, \rho)$$

Dabei steht St für die Zustandsmenge des Automaten, $s_0 \in St$ für den Startzustand und $\rho \subseteq St \times Actions \times St$ für die Zustandsübergangsrelation. Wir bezeichnen die Menge aller Automaten als $Machines$. Auch für Automaten kann eine Menge von Spuren angegeben werden. Die entsprechende Funktion $Traces \in Machines \to \mathscr{P}\,(Actions^{\omega})$ ist für einen Automaten $M = (St, s_0, \rho)$ definiert als

$$Traces\,(M) = \{\varepsilon\} \cup \{a_1 \circ a_2 \circ \ldots : \exists s_1, s_2, \ldots \in St : (s_0, a_1, s_1), (s_1, a_2, s_2), \ldots \in \rho\}$$

Die operationelle Semantik beschreiben wir mit einer Funktion $OpS \in Machines \to Spec$, die wie folgt definiert ist:

$$\begin{aligned} OpS(M) = \{(E, \leq) \in Processes_K \quad : \quad & Traces\,((E, \leq)) \subseteq Traces\,(M) \wedge \\ & \forall \tilde{\leq} \subset \leq : Traces\left(\left(E, \tilde{\leq}\right)\right) \not\subseteq Traces\,(M)\} \end{aligned}$$

Eine so hergeleitete operationelle Semantik umfasst alle durch den Automaten ausführbaren Prozesse, die maximal parallel sind und somit eine minimal mögliche Anzahl temporaler Ordnungsbeziehungen zwischen den Ereignissen aufweisen. Zudem fordern wir für Spezifikationen, dass die in den Prozessen vorkommenden Ereignisse disjunkt sind bis auf gemeinsame Teilprozesse.

Für eine Spezifikation kausaler Prozesse $S \in Spec$ gilt somit für beliebige Prozesse $P_1, P_2 \in S$ mit $P_1 = (E_1, \leq_1)$ und $P_2 = (E_2, \leq_2)$:

$$P_1 \barwedge P_2 = (E_1 \cap E_2, \leq_1 \cap \leq_2)$$

In dieser Auslegung lassen sich Ereignisse als Markierungen der Ausführung von Aktionen betrachten. Wird die gleiche Aktion an zwei verschiedenen Stellen innerhalb eines Prozesses durchgeführt, so erwarten wir, dass diese mit unterschiedlichen Ereignissen markiert wird.

Sofern Prozesse in einer Spezifikation entlang der Präfixordnung entwickelt werden, stellt die temporale Ordnung eines kausalen Prozesses auch eine kausale Ordnung für die im Prozess enthaltenen Ereignisse dar. Eine Prozessentwicklung gemäß der Präfixordnung entspricht dem intuitiven Verständnis eines Prozessablaufs in zeitlich fortschreitender Reihenfolge. Stehen zwei Ereignisse in einem so spezifizierten Prozess in temporaler Ordnungsbeziehung, so können wir gleichzeitig daraus folgern, dass das zeitlich nachgelagerte Ereignis nur auftritt, wenn das zeitlich vorausgehende Ereignis eingetreten ist. Dies folgt aus der Disjunktheitsforderung, die wir gerade aufgestellt haben. Es besteht demnach eine Ursache-/Wirkungsbeziehung zwischen den Ereignissen, also eine Kausalität. Diese Eigenschaft motiviert die Bezeichnung von Prozessen mit einer Temporalordnung als kausale Prozesse.

Beispiel. Wir kommen zurück zu unserem Beispiel einer Flugbuchungsanfrage und betrachten nun den Teilprozess der Grundpreisberechnung detaillierter.

Abbildung 4.6 zeigt die Spezifikation der Preisberechnung im Überblick. Nach erfolgter Kundenanfrage wird zunächst der Entfernungspreis ermittelt. Dieser wird auf Basis einer Entfernungstabelle und eines festen Tarifs pro Meile berechnet. Im Anschluss erfolgt die Kalkulation der Zuschläge. Dazu wird parallel überprüft, ob eine Verbindung an einem Wochentag angefragt wird. Ist dies der Fall, wird ein Zuschlag von 20% berechnet. Gleichzeitig wird der Zuschlag für die Destination nach einer vorhandenen Tabelle ermittelt. Wir gehen davon aus, dass die Fluggesellschaft für beliebte Ziele einen pauschalen Zuschlag von 50 EUR berechnet. Nach Berechnung der Zuschläge werden diese zum Entfernungspreis addiert und anschließend als Grundpreis ausgegeben.

Wir formalisieren die Preisberechnung anhand eines Automaten $HA_2 = (St, s_0, \rho)$, dessen Zustandsmenge wie folgt definiert ist:

$$St \subseteq String \times String \times Date \times Val \times Val \times Val \times Val$$

Der Startzustand des Automaten ist $s_0 = (\bot, \bot, \bot, \bot, \bot, \bot, \bot)$. Das Siebentupel eines Zustands steht dabei für interne Variablen, die der Automat benutzt. Das Symbol $\bot$ bedeutet, dass eine Variable nicht initialisiert ist. Die Zustandsübergangsfunktion des Automaten ist in Tabelle 4.2 abgebildet. Ein Grundstrich im Anfangszustand steht für eine beliebige Variable. Ein Grundstrich im Endzustand bedeutet, dass die Variable nicht verändert wird. Die Funktion $f \in String \times String \to Val$ ermittelt für eine gegebene Strecke auf Basis der Entfernungstabelle den entfernungsabhängigen Grundpreis. Die Funktion $g \in String \to Val$ berechnet den Zuschlag für eine gegebene Destination. $\langle monfri? \rangle$ steht für alle Daten, die auf einen Wochentag fallen, $\langle satsun? \rangle$ analog für Wochenendtage. Das Symbol $\diamondsuit$ repräsentiert den Zustand einer abgeschlossenen Preisberechnung.

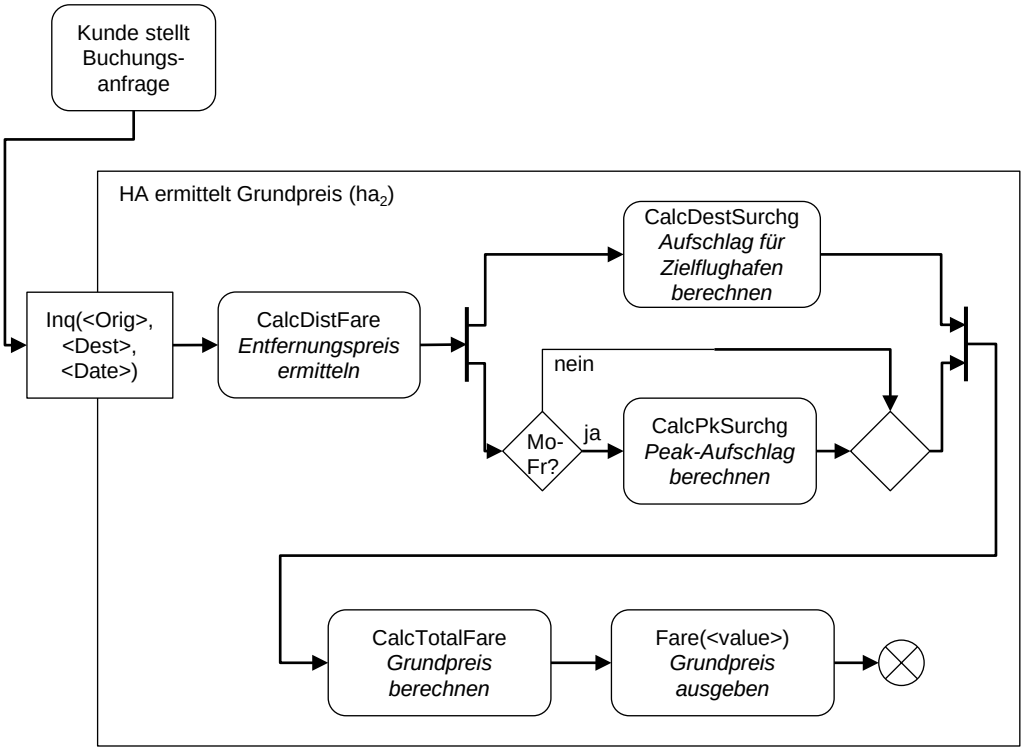

Abbildung 4.6: Spezifikation der Preisberechnung

$s \in St$	$a \in Actions$	$s \in St$
$(\bot, \bot, \bot, \_, \_, \_, \_)$	$Inq(\langle or \rangle, \langle dst \rangle, \langle dat \rangle)$	$(\langle or \rangle, \langle dst \rangle, \langle dat \rangle, \_, \_, \_, \_)$
$(\langle or \rangle, \langle dst \rangle, \_, \bot, \_, \_, \_)$	$CalcDistFare$	$(\_, \_, \_, f(\langle or \rangle, \langle dst \rangle), \_, \_, \_)$
$(\_, \_, \langle monfri? \rangle, \langle df \rangle, \bot, \_, \_)$	$CalcPkSurchg$	$(\_, \_, \_, \_, \langle df \rangle * 0.2, \_, \_)$
$(\_, \_, \langle satsun? \rangle, \langle df \rangle, \bot, \_, \_)$	ε	$(\_, \_, \_, \_, 0, \_, \_)$
$(\_, \langle dst \rangle, \_, \_, \_, \bot, \_)$	$CalcDestSurchg$	$(\_, \_, \_, \_, \_, g(\langle dst \rangle), \_)$
$(\_, \_, \_, \langle df \rangle, \langle ps \rangle, \langle ds \rangle, \bot)$	$CalcTotalFare$	$(\_, \_, \_, \_, \_, \_, \langle df \rangle + \langle ps \rangle + \langle ds \rangle)$
$(\_, \_, \_, \_, \_, \_, \langle fare \rangle)$	$Fare(\langle fare \rangle)$	$\diamondsuit$

Tabelle 4.2: Definition der Zustandsübergangsrelation ρ für die Preisberechnung

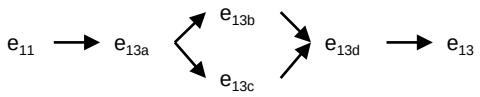

Abbildung 4.7: Exemplarische Preisberechnung

Ereignis	$e.a$
e_{11}	$Inq(\text{MUC}, \text{FRA}, 14.01.\#12{:}00)$
e_{13a}	$CalcDistFare$
e_{13b}	$CalcPkSurchg$
e_{13c}	$CalcDestSurchg$
e_{13d}	$CalcTotalFare$
e_{13}	$Fare(170)$

Tabelle 4.3: Ereignisse für die Preisberechnung in der exemplarischen Buchungsanfrage

Abbildung 4.7 zeigt einen exemplarischen Prozess für die Preisberechnung mit Ereignissen wie in Tabelle 4.3 definiert. Dieser Prozess ist in der operationellen Semantik $OpS(HA_2)$ enthalten. Wir gehen davon aus, dass die Strecke München-Frankfurt in einem Entfernungspreis von 100 resultiert. Der abgebildete Prozess realisiert die Preisberechnung für die Kundenanfrage von Herrn Müller, die wir in den vorangegangenen Beispielen diskutiert haben. Die für die Preisberechnung relevanten Ereignisse e_{11}, e_{13} der Buchungsanfrage P_k kommen im Prozess vor, die Ereignisse $e_{13a}, e_{13b}, e_{13c}, e_{13d}$ stellen dabei Zwischenschritte der Preisberechnung dar. □

Auch ungeordnete Prozesse können auf Basis von einfachen Automaten spezifiziert werden. Dies ist der Fall, wenn die operationelle Semantik für eine Ereignismenge jede beliebige partielle Ordnung über den Ereignissen zulässt.

4.3.2.3 Temporale Prozesse und Spezifikationen

Mit kausalen Prozessen lassen sich Aussagen über die Eintrittsreihenfolge von Ereignissen treffen. Beobachten wir jedoch in der Praxis einen Einzelablauf, so können wir in voller Allgemeinheit nicht die relative temporale Ordnung der Ereignisse aus der Beobachtung ableiten. Für ein gegebenes Ereignispaar des Ablaufs lässt sich ohne weitere Kenntnis der zugrundeliegenden Ablauflogik nicht beurteilen, ob diese zwingend in der gegebenen Reihenfolge auftreten müssen. Wir können die Beobachtung eines Einzelablaufs somit als sequentialisierten Prozess auffassen, da wir die Ereignisse stets in einer bestimmten Reihenfolge erfassen. Das dem Prozess zugrundeliegende zeitliche Ordnungsmodell kann grundsätzlich auch andere Reihenfolgen zulassen, jedoch können diese aus der Betrachtung eines einzelnen Ablaufs nicht rekonstruiert werden. [Bro98]

Wir führen daher eine dritte Repräsentation des Prozessbegriffs ein, die Prozesse temporal darstellt, so wie sie von einem Beobachter als Einzelablauf erfasst werden können. Dazu erweitern wir zunächst unseren Ereignisbegriff um eine diskrete Zeitrechnung.

Definition 18. Temporales Ereignis

Wir definieren ein temporales Ereignis als ein Ereignis, dem ein bestimmter Auftrittszeitpunkt zugeordnet ist. Dazu annotieren wir ein Ereignis e mit zwei Attributen wie folgt:

$e.a \in Action$	Die Aktion, die in dem Ereignis ausgeführt wird
$e.t \in \mathbb{N}$	Ein diskreter Zeitpunkt, zu dem das Ereignis auftritt

□

In der Definition eines temporalen Ereignisses folgen wir dabei unserer simplifizierenden Annahme, dass Ereignisse zeitpunktbezogen erfasst werden können. Wir bezeichnen die Menge aller temporalen Ereignisse mit $Events_T$. Wir stellen den Bezug zwischen der Ereignismenge $Events$ und der Menge aller temporalen Ereignisse $Events_T$ durch eine Zuordnungsfunktion her:

$$ta \in Events \to Events_T$$

Die Funktion ta weist jedem Ereignis einen Auftrittszeitpunkt zu. Wir bezeichnen ta daher auch als Zeitpunktzuordnung und fordern folgende zwei Eigenschaften für die Funktion:

(1) $\forall e \in Events : e.a = ta(e).a$ (Aktionskonsistenz)

(2) $\forall e_1, e_2 \in Events : ta(e_1) = ta(e_2) \Rightarrow e_1 = e_2$ (Injektivität)

Mit der ersten Eigenschaft gewährleisten wir, dass die Aktionseigenschaft in der Zuordnung erhalten bleibt. Die Forderung nach Injektivität reflektiert die Tatsache, dass ein Ereignis eine Markierung einer einmaligen Durchführung einer Aktion darstellt und folglich eine eindeutige Zuordnung vorgenommen werden kann. Wir bezeichnen mit $TA \subset Events \to Events_T$ die Menge aller Zeitpunktzuordnungen, die diese beiden Eigenschaften erfüllen.

Definition 19. Temporaler Prozess

Wir definieren einen temporalen Prozess als einen Ablauf, der auf Basis von Ereignissen beschrieben wird und in dem jedem Ereignis ein eindeutiger Eintrittszeitpunkt zugeordnet ist. Formal beschreiben wir einen temporalen Prozess als eine Ereignismenge:

$$P \subseteq Events_T$$

□

Die Menge der temporalen Prozesse bezeichnen wir als $Processes_T = \mathscr{P}(Events_T)$. Sie stellt eine Repräsentation unseres abstrakten Prozessbegriffs dar:

$$(Processes_T, \subseteq)$$

Die auf dieser Menge definierte Ordnung ist die Teilmengenrelation bezüglich der im Prozess enthaltenen Ereignisse. Sie ist nicht zu verwechseln mit einer temporalen Ordnung der Ereignissen, die sich bei jedem temporalen Prozess aus dem Zeitpunktattribut ergibt. Wir bezeichnen einen Prozess $P' \in Processes_T$ als Teilprozess eines anderen Prozesses $P \in Processes_T$, wenn dieser eine Teilmenge der Ereignisse des anderen Prozesses enthält:

$$P' \subseteq P$$

Auch für temporale Prozesse können Präfixe betrachtet werden. Wir verstehen unter einem Präfix eines temporalen Prozesses die Teilmenge aller Ereignisse, die bis zu einem bestimmten Zeitpunkt eintreten. Formal erfassen wir für einen Prozess $P \in Processes_T$ und einen beliebigen Zeitpunkt $t \in \mathbb{N}$ ein entsprechendes Präfix als:

$$P \downarrow t = \{e \in P : e.t \leq t\}$$

Die Menge der temporalen Prozesse $(Processes_T, \subseteq)$ bildet analog zu den ungeordneten Prozessen einen vollständigen Verband, der sich direkt aus der Potenzmengeneigenschaft ableiten lässt. Folglich ist $Processes_T$ eine Halbordnung. Die Eigenschaften der Reflexivität, Antisymmetrie und Transitivität folgen aus der Teilmengendefinition. Das Nullelement des Verbandes ist der Prozess $\emptyset$, der keine Ereignisse enthält, das Einselement die Menge aller Ereignisse $Events_T$. Im Verband lässt sich für jede Teilmenge an Prozessen ein Supremum konstruieren, das durch Vereinigung der in den Prozessen enthaltenen Ereignisse gebildet wird.

Jeder ungeordnete Prozess repräsentiert eine Menge von temporalen Prozessen, in denen den Ereignissen des ungeordneten Prozesses beliebige Zeitpunkte zugeordnet werden. Wir definieren eine Abbildungsfunktion $Tmp \in Processes_N \rightarrow \mathscr{P}(Processes_T)$ wie folgt:

$$Tmp(P) = \{\{ta(e) : e \in P\} : ta \in TA\}$$

Analog repräsentiert auch jeder kausale Prozess eine Menge von temporalen Prozessen, die korrespondierende Ereignisse in der durch den kausalen Prozess vorgegebenen Reihenfolge enthalten. Auch dazu definieren wir eine Abbildungsfunktion $Tmp \in Processes_K \rightarrow \mathscr{P}(Processes_T)$:

$$Tmp((E, \leq)) = \{\{ta(e) : e \in E\} : ta \in TA : \forall e_1, e_2 \in E : e_1 \leq e_2 \Rightarrow ta(e_1).t \leq ta(e_2).t\}$$

Wir bezeichnen einen Prozess $P \in Processes_T$ als streng sequentiell, falls gilt:

$$\forall e_1, e_2 \in P : e_1.t = e_2.t \Rightarrow e_1 = e_2$$

Es gilt zu beachten, dass die temporale Ordnung in kausalen Prozessen keine strenge Sequentialität fordert, sondern auch die gleichzeitige Ausführung von Ereignissen zulässt.

Beispiel. Wir kommen zurück auf unsere exemplarische Kundenanfrage, die wir bereits als kausalen Prozess $P_k = (E, \leq)$ definiert haben. Wir untersuchen nun exemplarisch drei korrespondierende temporale Prozesse, die auf Basis dreier Zeitpunktzuordnungen $ta_1, ta_2, ta_3 \in TA$ wie in Tabelle 4.4 dargestellt definiert sind.

- $P_{t1} = \{ta_1(e) : e \in E\}$
- $P_{t2} = \{ta_2(e) : e \in E\}$
- $P_{t3} = \{ta_3(e) : e \in E\}$

Die drei Prozesse sind in Abbildung 4.8 entlang eines Zeitstrahls dargestellt. Wir erkennen, dass die temporalen Prozesse die im kausalen Prozess P_k festgelegten zeitlichen Abhängigkeiten zwischen den Ereignissen respektieren. Es gilt $P_{t1}, P_{t2}, P_{t3} \in Tmp(P_k)$. Die Abbildung auf temporale Prozesse lässt für eine gegebene zeitliche Ordnung eine Vielzahl von Verteilungen über die Zeit hinweg zu. □

	e_{11}	e_{12}	e_{13}	e_{14}	e_{15}
$ta_1(\ldots).t$	1	2	3	4	5
$ta_2(\ldots).t$	1	5	3	6	8
$ta_3(\ldots).t$	3	11	7	11	14

Tabelle 4.4: Exemplarische Zeitpunktzuordnungen für die Buchungsanfrage

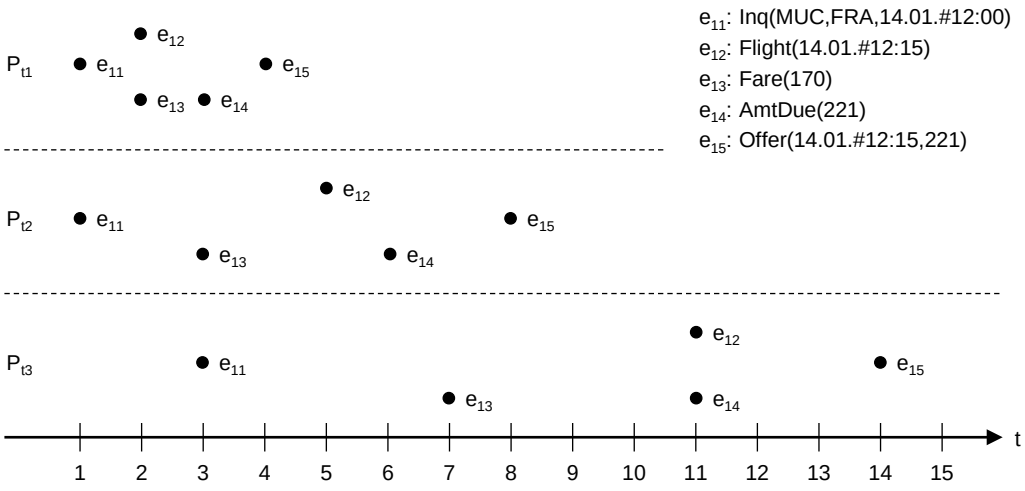

Abbildung 4.8: Buchungsanfrage, dargestellt in temporalen Prozessen

Auch für temporale Prozesse können semantische Aussagen anhand von Spezifikationen getroffen werden. Eine Spezifikation eines temporalen Prozesses kann aus der operationellen Semantik eines Automaten hergeleitet werden, indem zuerst die kausalen Prozesse aus der Automatendefinition abgeleitet und diese anschließend in temporale Prozesse umgewandelt werden. Ein formaler Automat $M \in Machines$ resultiert somit in der folgenden Spezifikation temporaler Prozesse:

$$\{Tmp(P) : P \in OpS(M)\}$$

4.3.2.4 Anwendungsbereich von Prozessen und Spezifikationen

Wir haben nun drei Formen der konkreten Prozessrepräsentation auf Basis von Ereignissen kennengelernt, die alle drei eine Ordnung bezüglich der Prozessentwicklung respektieren und damit als Instanzierungen des abstrakten Prozessmodells angesehen werden können. Wir werden die weiteren Eigenschaften von Prozessen daher wieder anhand der abstrakten Menge *Processes* diskutieren und gehen dabei davon aus, dass sie auf alle drei Repräsentationen anwendbar sind. Wenn eine Eigenschaft nur für eine Repräsentation gilt ist, werden wir dies explizit anmerken.

In allen drei Repräsentationsformen können in Prozessen theoretisch beliebig große Mengen von aufgetretenen Aktionen mit entsprechenden zeitlichen Abhängigkeiten erfasst werden. Damit sind dem Beschreibungsumfang von Prozessen grundsätzlich keine Grenzen gesetzt. In der Praxis werden wir aber selbstverständlich in Prozessen immer nur Ausschnitte der Realität untersuchen können. In diesem Zusammenhang ist von Interesse, welche Aktionen in einem Prozess vorkommen.

Definition 20. Anwendungsbereich

Wir definieren den Anwendungsbereich eines Prozesses als die Menge von Aktionen, die im Prozess zur Ausführung kommen. Formal erfassen wir den Anwendungsbereich mit einer Funktion $Scope \in Processes \to \mathscr{P}(Actions)$, die für einen Prozess $P \in Processes$ wie folgt definiert ist:

$$Scope(P) = \{a \in Actions : \exists e \in P : e.a = a\}$$

Im Fall von kausalen Prozessen bezieht sich die *Scope* Funktion auf die Ereignismenge des Prozesses. □

Wir verwenden im Folgenden auch für die Menge der Prozesse, deren Anwendungsbereich $A \subseteq Actions$ oder eine Teilmenge davon ist, die abkürzende Schreibweise:

$$Processes(A) = \{P \in Processes : Scope(P) \subseteq A\}$$

Für die detaillierte Untersuchung eines Prozesses kann es hilfreich sein, einen Prozess mit großem Anwendungsbereich durch Projektion auf einen Ausschnitt mit kleinerem Anwendungsbereich zu beschränken.

Definition 21. Projektion

Wir definieren die Projektion als eine Sicht $\Pi_A \in \mathit{Views}$ auf einen Prozess. Diese ist für $P \in \mathit{Processes}$ und eine Aktionsmenge $A \subseteq \mathit{Actions}$ wie folgt definiert:

$$\Pi_A(P) = \{e \in P : e.a \in A\}$$

Bei kausalen Prozessen bezieht sich die Projektion auf die Ereignismenge und in der temporalen Ordnung des Prozesses werden alle Tupel entfernt, die sich auf mindestens ein entferntes Ereignis beziehen. □

Mit der Projektion können für einen gegebenen Prozess Ausschnitte betrachtet werden, indem alle Ereignisse gefiltert werden, die nicht im durch die Projektion vorgesehenen Anwendungsbereich liegen. Die Projektion und die Betrachtung von Anwendungsbereichen lässt sich analog auch auf Spezifikationsmengen anwenden.

4.3.3 Ereignisbasierte Prozessregeln

Da alle drei Prozessrepräsentationen die Entwicklungsordnung respektieren, kann die Entwicklung von Prozessen in den Repräsentationen auch anhand von Prozessregeln erfolgen. Prozessregeln treffen semantische Aussagen durch Festlegung von Ursache-/Wirkungszusammenhängen zwischen Ereignissen in Prozessen. Eine Regel leitet aus bereits aufgetretenen Ereignissen zusätzliche Ereignisse ab, die aus den existierenden Ereignissen semantisch resultieren.

Dabei existiert naturgemäß ein enger Zusammenhang zwischen Regeln und Spezifikationen. Wir werden im Folgenden Prozessregeln auch auf Basis einer gegebenen Spezifikation formulieren. In diesem Fall stellt eine Regel einen Zusammenhang zwischen zwei Prozessen in einer Spezifikation her, so dass der eine Prozess ein Teilprozess des anderen ist. Auf diese Weise beschreibt eine Regel, wie Prozesse innerhalb der Spezifikation entwickelt werden können. Wir können damit leicht die Analogie zu formalen Automaten erkennen. Eine Regel leitet auf Basis eines gegebenen Prozesses einen Zustand ab und überführt diesen durch Ergänzung weiterer Ereignisse in einen neuen Zustand. Prozessregeln sind somit vergleichbar mit den Produktionsregeln eines Automaten oder einer formalen Grammatik.

Regeln müssen jedoch nicht zwingend auf Basis einer existierenden Spezifikation formuliert werden. Der Vorteil von Prozessregeln liegt darin, dass relative Aussagen zu Ereigniszusammenhängen in Prozessen getroffen werden können, ohne den vollständigen Aufbau eines Prozess angeben zu müssen. Dies beruht auf der Überlegung, dass wir grundsätzlich nur Ausschnitte der Realität erfassen können und dabei eine Annahme bezüglich der Umgebung treffen. So beschreibt eine Prozessregel auf Basis existierender Ereignisse, aus denen die Umgebungsannahme abgeleitet wird, ein bestimmtes Verhalten.

Wir werden in den folgenden Beispielen Regeln jedoch vorwiegend auf Basis von Spezifikationen entwickeln, um die Formalisierung der Regeln anschaulicher zu gestalten.

4.3.3.1 Regelanwendung

Prinzipiell kann der Regelbegriff, den wir im abstrakten Prozessmodell eingeführt haben, direkt auf die drei Repräsentationen angewendet werden. Eine Regel r entwickelt damit einen Prozess entlang der Entwicklungsordnung der entsprechenden Repräsentation:

$$\xrightarrow{r} \subseteq \mathit{Processes} \times \mathit{Processes}$$

Für $P \xrightarrow{r} Q$ gilt $P \preceq Q$. Dabei steht die Menge *Processes* für eine der drei Prozessrepräsentationen $\mathit{Processes}_N, \mathit{Processes}_K, \mathit{Processes}_T$ und $\preceq$ für die entsprechende Entwicklungsordnung, die in der jeweiligen Repräsentation gilt. Wir bezeichnen für eine Regelanwendung $P \xrightarrow{r} Q$ den Prozess P auch als den beobachteten Prozess, den die Regel zu einem resultierenden Prozess Q entwickelt.

Die Menge der kausalen Prozesse ist nicht disjunkt. So besteht beispielshalber zwischen den drei Prozessen $P_1 = (\{e_1, e_2\}, \{(e_1, e_2)\})$, $P_2 = (\{e_1, e_2\}, \{(e_2, e_1)\})$ und $P_3 = (\{e_1, e_2\}, \emptyset)$ ein Zusammenhang. Für Spezifikationen haben wir bereits gefordert, dass die Ereignisse der Prozesse disjunkt sind bis auf gemeinsame Teilprozesse. Auf diese Weise gewährleisten wir eine eindeutige temporale Ordnung zwischen den Ereignissen. Wir fordern in gleicher Weise für kausale Prozessregeln, dass diese nur Aussagen treffen zu Prozessen, die disjunkt sind hinsichtlich ihrer Ereignisse. Wir nehmen wiederum Teilprozesse aus, um die Entwicklung eines Prozesses regelbasiert beschreiben zu können.

Eine Prozessregel leitet aus der Beobachtung von Ereignissen weitere Ereignisse ab, die die Fortsetzung eines Prozesses darstellen. Daraus folgen Abhängigkeiten zwischen den Ereignissen. Wir fordern, dass eine Regel solche Abhängigkeiten in kausaler Weise entwickelt. Wie bereits zuvor diskutiert unterscheiden wir zwei Formen der Kausalität, die notwendige und die hinreichende Voraussetzung. Für beide Formen der Kausalität fordern wir, dass bei der Anwendung einer Regel das ursächliche Ereignis im beobachteten Prozess zeitlich vor dem Ereignis liegt, das als Wirkung durch die Regel im resultierenden Prozess ergänzt wird.

Für kausale Prozesse bedeutet dies, dass sich Kausalitäten zwischen zwei Ereignissen nur entlang der temporalen Ordnung des Prozesses entwickeln können. Dies motiviert auch die Bezeichnung als kausalen Prozess. Wir bezeichnen eine Regel r als kausal, wenn diese Prozesse entlang der Präfixordnung entwickelt und für drei beliebige Prozesse $P, P_1, P_2 \in \mathit{Processes}_K$ gilt:

$$\begin{aligned} P \sqsubseteq P_1 \wedge P \sqsubseteq P_2 \quad \Rightarrow \quad & \left\{ Q : \exists (P_1, Q_1) \in \xrightarrow{r} : Q \sqsubseteq Q_1 \wedge (Q \barwedge P_1 = Q \barwedge P_2 = P) \right\} \\ & = \left\{ Q : \exists (P_2, Q_2) \in \xrightarrow{r} : Q \sqsubseteq Q_2 \wedge (Q \barwedge P_1 = Q \barwedge P_2 = P) \right\} \end{aligned}$$

Bei der Anwendung einer kausalen Regel ist für zwei Prozesse, die ein gleiches Präfix aufweisen, die Menge alle Teilprozesse, die ausschließlich aus diesem Präfix heraus entwickelt werden können, identisch. Dies folgt aus der Forderung, dass Kausalitäten nur zwischen Ereignissen mit temporalen Zusammenhang bestehen dürfen und somit das Präfix ursächlich für die weitere Prozessentwicklung ist. In der vorgestellten Eigenschaft werden dabei alle Teilprozesse jenseits des Präfixes aus der Betrachtung ausgenommen, die bereits in den ursprünglichen Prozessen enthalten waren und damit nicht durch die Regelanwendung entwickelt wurden.

In analoger Weise fordern wir auch für temporale Prozesse, dass in deren regelbasierter Entwicklung Ursachen stets den Wirkungen zeitlich vorangestellt sind. Dies bedeutet, dass eine Regel für zwei Prozesse mit dem gleichen Präfix für den Zeitraum, der durch das Präfix erfasst ist, die gleichen Wirkungen erzeugen muss (vgl. [BS01, Bro10]). Wir bezeichnen eine Prozessregel r für zwei beliebige temporale Prozesse $P_1, P_2 \in Processes_T$ und einen beliebigen Zeitpunkt $t \in \mathbb{N}$ als kausal, wenn folgendes gilt:

$$P_1 \downarrow t = P_2 \downarrow t \Rightarrow \left\{ Q_1 \downarrow t : P_1 \xrightarrow{r} Q_1 \right\} = \left\{ Q_2 \downarrow t : P_2 \xrightarrow{r} Q_2 \right\}$$

Wir sprechen in diesem Fall auch von schwach kausalen Prozessregeln. In Abgrenzung dazu ist eine Regel stark kausal, wenn die Wirkungen erst einen Zeitschritt später eintreten und damit auf der rechten Seite der Implikation für die Präfixe $t+1$ gilt.

In ungeordneten Prozessen kann eine solche Kausalitätsforderung nicht gestellt werden, da in Ermangelung eines Zeitbegriffs nicht zwischen Ursache und Wirkung differenziert werden kann.

Beispiel. Wir stellen eine Prozessregel für die detaillierte Preisberechnung von Holiday Airways auf, die wir bereits auf Basis des Automaten HA_2 spezifiziert haben. Die Regel ha_{2wb} für kausale Prozesse ist wie folgt definiert:

$$P \xrightarrow{ha_{2wb}} Q \Leftrightarrow P, Q \in OpS(HA_2) : P \preceq Q \wedge Inq(\langle orig \rangle, \langle dest \rangle, \langle date \rangle) \notin Scope(Q) \setminus Scope(P)$$

Der Zusatz *wb* steht für White-Box. Die Regel ha_{2wb} erfasst im Kontrast zur Regel ha_2 nicht nur das Ergebnis der Preisberechnung, sondern beschreibt auch den Vorgang zur Ermittlung des Preises.

Die Regel entwickelt Prozesse entlang der Spezifikation der detaillierten Preisberechnung, die durch HA_2 festgelegt wird. Wir lassen jedoch nicht zu, dass Kundenanfragen durch die Regel generiert werden. Wir formulieren die Regel aus der Perspektive von Holiday Airways und die Fluggesellschaft hat keinen Einfluss auf die Erstellung von Buchungsanfragen, sondern kann auf diese nur reagieren.

Die Abbildung 4.9 zeigt drei exemplarische Ableitungen der Prozessregel. Die erste Entwicklung ist eine vollständige Durchführung der Preisberechnung auf Basis der Kundenanfrage (Ereignis e_{11}). Die Regel lässt aber auch Prozessentwicklungen in mehreren Schritten zu. Auf diese Weise kann anhand der Regel der Prozessablauf zur Preiskalkulation nachvollzogen werden. Die zweite und dritte Zeile der Abbildung zeigen eine solche Entwicklung des Prozesses in zwei Schritten.

Das Beispiel zeigt auch einen wichtigen Unterschied zwischen Prozessregeln und den Produktionsregeln eines Automaten. In den Prozessregeln erfolgt immer eine vollständige Ableitung des Zustands aus dem beobachteten Prozess, eine Prozessregel ist per se zustandsfrei. □

Betrachten wir den Zusammenhang zwischen Prozessregeln für ungeordnete, kausale und temporale Prozesse, so stellt sich dieser wie in Abbildung 4.10 gezeigt dar. Jede Prozessregel für ungeordnete Prozesse kann in eine entsprechende Prozessregel für kausale Prozesse überführt

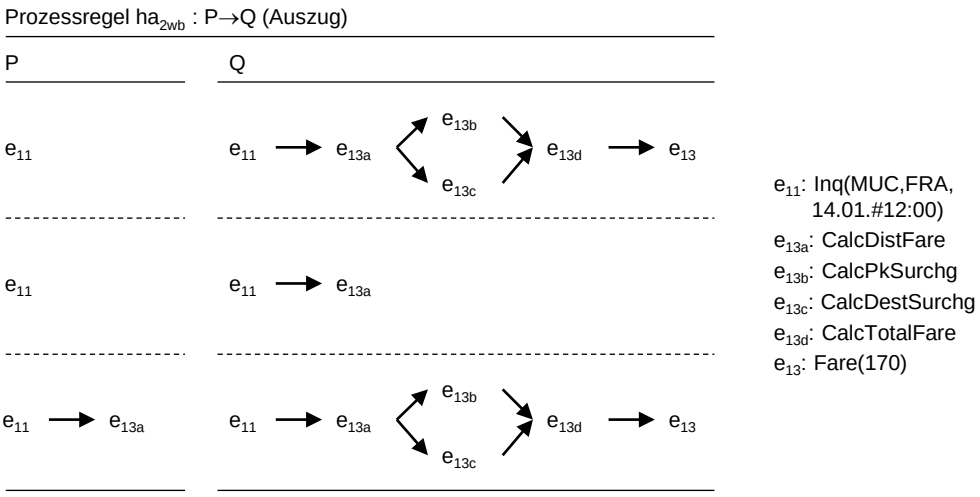

Abbildung 4.9: Definition der Prozessregel zur Preisberechnung ha_{2d} (Auszug)

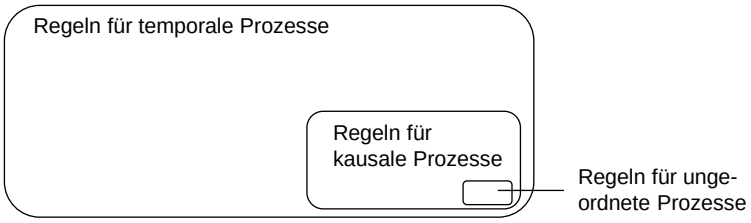

Abbildung 4.10: Zusammenhang zwischen Regeln für temporale, kausale und ungeordnete Prozesse

werden. Die Übersetzung ist trivial, wir übernehmen die Ereignismengen unverändert und lassen in den kausalen Regeln beliebige kausale Beziehungen zwischen den Ereignissen zu, solange die Entwicklungsordnung eingehalten wird. Damit ist auch die Konsistenzbedingung erfüllt. Die Umkehrung gilt nicht, es kann nicht jede kausale Prozessentwicklung in eine ungeordnete Prozessentwicklung überführt werden.

Weiterhin kann auch jede kausale Prozessregel $\xrightarrow{k} \subseteq Processes_K \times Processes_K$ in eine korrespondierende temporale Prozessregel $\xrightarrow{t} \subseteq Processes_T \times Processes_T$ übersetzt werden. Für zwei kausale Prozesse $P_k, Q_k \in Processes_K$ gilt in diesem Fall:

$$P_k \xrightarrow{k} Q_k \Leftrightarrow \bigwedge P_t \xrightarrow{t} Q_t : (P_t, Q_t) \in Tmp(P_k) \times Tmp(Q_k) \wedge P_t \subseteq Q_t$$

Die Umkehrung gilt wiederum nicht. So können wir Regeln für temporale Prozesse aufstellen, die nicht in kausalen Regeln ausdrückbar sind. Dies trifft insbesondere auf Prozesse zu, an deren Ausführung wir zeitliche Anforderungen stellen. Die Aussage „Holiday Airways muss eine Kundenanfrage innerhalb von 30 Sekunden beantworten" lässt sich nicht durch eine kausale Prozessregel, sehr wohl aber durch eine temporale Prozessregel ausdrücken. Wie bereits zuvor angedeutet können hier auch Mischformen aus kausalen und temporalen Regeln zum Einsatz kommen, auf die wir aber nicht weiter eingehen werden.

Die drei Repräsentationen eines Prozesses verhalten sich homomorph bezüglich einer Prozessregel, wenn die Regel gemäß der gerade vorgestellten Logik transformiert wird. Dies ermöglicht, dass sich die Entwicklung temporale Prozesse auch auf Basis der leichter zu definierenden Prozessregeln für kausale oder ungeordnete Prozesse beschreiben lässt, solange eine entsprechende Abstraktion von zeitlichen beziehungsweise kausalen Aspekten möglich ist.

4.3.3.2 Anwendungsbereich von Regeln

Unsere Überlegungen zum Anwendungsbereich von Prozessen gelten in gleicher Weise auch für Prozessregeln. Eine Regel trifft typischerweise nur eine semantische Aussage zu einem kleinen Ausschnitt der Realität. Entsprechend beziehen sich Regeln zumeist nur auf eine eingeschränkte Menge von Aktionen. Wir erweitern daher den Begriff des Anwendungsbereichs auch auf eine Prozessregel. Der Anwendungsbereich einer Regel entspricht dabei der Vereinigung des Anwendungsbereichs aller Prozesse in der Wertemenge der Regel. Für die Funktion $Scope \in Rules \rightarrow \mathscr{P}(Actions)$ gilt damit für eine Prozessregel r:

$$Scope(r) = \bigcup_{P \in Range(r)} Scope(P)$$

Durch die Beschränkung des Anwendungsbereichs einer Regel kann die Regeldefinition erheblich vereinfacht werden. Mit der Komposition können aus einfachen Regeln mit beschränktem Anwendungsbereich sukzessive komplexe, übergreifende Regelwerke hergeleitet werden. Um eine Komposition von Prozessregeln mit unterschiedlichen Anwendungsbereichen durchführen zu können, müssen diese jedoch einen gemeinsamen Definitionsbereich aufweisen. Wir gehen

daher davon aus, dass Prozessregeln für Prozesse außerhalb ihres Anwendungsbereichs definiert sind und diese nicht weiter entwickeln. Für Regeln mit eingeschränktem Anwendungsbereich erweitern wir dazu deren Domäne für Prozesse, die nicht im Anwendungsbereich liegen.

Definition 22. Prozessregel mit erweitertem Anwendungsbereich

Für eine Regel r mit Anwendungsbereich $A = Scope(r)$ definieren wir eine korrespondierende Regel $\hat{r}$ mit erweitertem Anwendungsbereich, so dass für beliebige Prozesse $P, Q \in Processes(A)$ und $R \in Processes(\bar{A})$ gilt:

$$R \veebar P \xrightarrow{\hat{r}} R \veebar Q \Leftrightarrow P \xrightarrow{r} Q$$

$\bar{A}$ steht hierbei für das Komplement von A. Das Supremum existiert in der gegebenen Definition in jedem Fall, da die Ereignismengen auf Grund der komplementär definierten Anwendungsbereiche disjunkt sind und damit keine konträren Ordnungsbeziehungen in den Prozessen auftreten können. □

Wir bezeichnen im Folgenden eine Prozessregel mit erweitertem Anwendungsbereich abkürzend auch als vollständige Prozessregel. Eine vollständige Regel ist nicht notwendigerweise total, da sie innerhalb ihres ursprünglichen Anwendungsbereichs nicht zu allen Prozessen eine Entwicklungsaussage treffen muss. Es gilt jedoch Totalität außerhalb des Anwendungsbereichs der ursprünglichen Regel.

Durch die Erweiterung des Anwendungsbereichs können somit auch Aussagen zur Entwicklung von Prozessen getroffen werden, die Aktionen enthalten, die nicht im Anwendungsbereich der ursprünglichen Regel liegen. Allerdings haben alle Aktionen außerhalb dieses Anwendungsbereichs keinen Einfluss auf die Entwicklung. Wenn wir die Beobachtung eines Prozesses als Zustandsbildung ansehen, so verändern alle Aktionen außerhalb des Anwendungsbereichs der Regel während der Prozessbeobachtung nicht den Zustand.

Betrachten wir den Zusammenhang zwischen einer Regel mit eingeschränktem Anwendungsbereich r und einer vollständigen Regel $\hat{r}$, so lässt sich eine Äquivalenzrelation definieren mit $P_1 \sim P_2 \Leftrightarrow \Pi_A(P_1) = \Pi_A(P_2)$. Die Projektionen auf den Anwendungsbereich A repräsentieren dabei die Äquivalenzklassen der Relation und die vollständige Regel $\hat{r}$ entwickelt Prozesse entlang dieser Äquivalenzklassen. Folglich kann die Prozessentwicklung anhand einer vollständigen Regel $\hat{r}$ auch mit der ursprünglichen Regel r beschrieben werden, die in homomorpher Weise auf den Repräsentationen der Äquivalenzklassen operiert. Um die Äquivalenzklassen auch in der Komposition zu erhalten, erweitern wir die drei Kompositionsoperatoren $\hat{|}, \hat{\|}, \hat{;} \in Rules \times Rules \to Rules$ wie folgt:

$$r_1 \hat{|} r_2 = \Pi_{Scope(r_1) \cup Scope(r_2)} (\hat{r_1} \mid \hat{r_2})$$

$$r_1 \hat{\|} r_2 = \Pi_{Scope(r_1) \cup Scope(r_2)} (\hat{r_1} \parallel \hat{r_2})$$

$$r_1 \hat{;} r_2 = \Pi_{Scope(r_1) \cup Scope(r_2)} (\hat{r_1}; \hat{r_2})$$

Beispiel. Wir betrachten erneut die detaillierte Definition der Preisberechnung von Holiday Airways und die entsprechende Prozessregel ha_{2wb}, die wir bereits im letzten Abschnitt aufgestellt

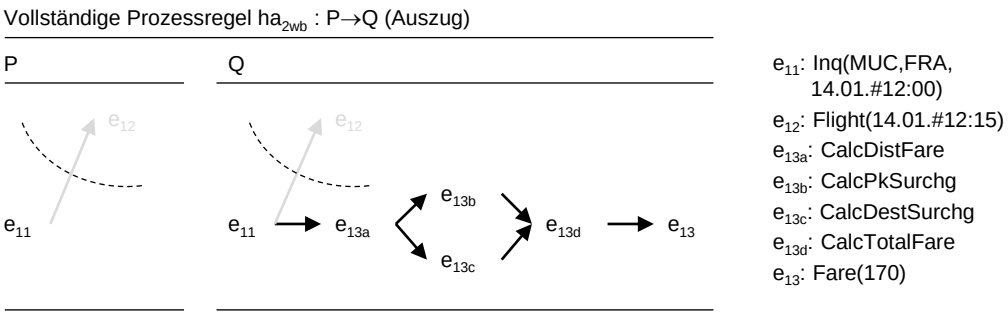

Abbildung 4.11: Vollständige Prozessregel $\hat{ha_{2wb}}$ (Auszug)

haben. Die Prozessregel ist momentan so definiert, dass sie nur auf Prozesse anwendbar ist, die Aktionen bezüglich der Preisberechnung enthalten. Nun stellt sich die Frage, wie die Regel mit einem Prozess umgeht, in dem beispielsweise die Ermittlung des nächsten verfügbaren Verbindung im Ereignis e_{12} schon erfolgt ist: $(\{e_{11}, e_{12}\}, \{(e_{11}, e_{12})\})$. Wir erwarten in diesem Fall, dass die Preisberechnung parallel zur Ermittlung der nächsten Verbindung durchgeführt werden kann.

Durch die Erweiterung des Anwendungsbereichs der Regel zu $\hat{ha_{2wb}}$ lässt sich dieses Verhalten erzielen. Die Regel ist, wie in Abbildung 4.11 dargestellt, nun auf $(\{e_{11}, e_{12}\}, \{(e_{11}, e_{12})\})$ anwendbar, führt die Preisberechnung aber weiterhin in Abhängigkeit der Kundenanfrage e_{11} durch und lässt dabei das Ergebnis der Flugverfügbarkeitsanfrage e_{12} unberührt. □

Mit Hilfe von Projektionen kann der Anwendungsbereich einer Prozessregel auf einen bestimmten Ausschnitt konzentriert werden. Auf diese Weise können wir einzelne Teilaspekte eines komplexen Prozessverhaltens, das in einer übergreifenden Prozessregel ausgedrückt ist, untersuchen. In der Projektion bilden wir die Entwicklung des Gesamtprozesses auf einen eingeschränkten Anwendungsbereich ab und abstrahieren dabei von allen Entwicklungen, die außerhalb des Bereichs stattfinden. Wir führen dazu eine abkürzende Notation ein. Für eine Regel r stellt $[r]_A$ die Beschränkung dieser Regel auf den Anwendungsbereich $A \subseteq Actions$ nach folgender Maßgabe dar:

$$P \xrightarrow{r} Q \Rightarrow \Pi_A(P) \xrightarrow{[r]_A} \Pi_A(Q)$$

Die Projektion kann zu einer nicht deterministischen Verhaltensbeschreibung führen, da die Prozessentwicklung nun potenziell von Ereignissen abhängt, die in der Regel nicht mehr erfasst sind. Die Regel beschreibt im projizierten Anwendungsbereich die Menge aller Prozessentwicklungen, die in voller Allgemeinheit möglich wären.

Die Projektion ermöglicht auch das Ausblenden von Zwischenschritten in einem detailliert beschriebenen Ablauf. Dies setzt voraus, dass die Regel transitiv abgeschlossen ist, so dass das Endergebnis des Prozesses auch ohne Betrachtung der Zwischenschritte entwickelt werden kann. Eine Regel, die durch Projektion von Zwischenschritten abstrahiert, wird auch als Black-Box-Perspektive der ursprünglichen Regel bezeichnet.

Neben der Fokussierung auf bestimmte Aktionen in einem Prozess ist auch von Interesse, welche Aktionen durch eine Regel in der Prozessentwicklung hinzugefügt werden. Wir definieren dazu eine Funktion $Delta \in Rules \to \mathscr{P}(Actions)$, die für eine Regel r die Menge aller Aktionen angibt, die durch Anwendung der Regel im Prozess durchgeführt werden. Sie beschreibt das Delta an Aktionen zwischen beobachteten und resultierenden Prozessen:

$$Delta(r) = \bigcup_{(P,Q) \in \xrightarrow{r}} Scope(Q) \setminus Scope(P)$$

Wir können nun auch die Menge von Aktionen, mit denen ein Prozess entwickelt wird, beschränken. Dazu erweitern wir die eingeführte abkürzende Notation. Für eine gegebene Regel $[r]_A$ stellt $[r]_{A,B}$ die Projektion der Prozessentwicklung auf die Aktionen $B \subseteq A$ dar mit folgender Definition:

$$P \xrightarrow{[r]_A} Q \Rightarrow P \xrightarrow{[r]_{A,B}} P \veebar \Pi_B(Q)$$

Es gilt offensichtlich $Delta\left([r]_{A,B}\right) \subseteq B$. Mit der Anwendung des Supremums $P \veebar \Pi_B(Q)$ gewährleisten wir, dass bereits im Prozess enthaltene Aktionen jenseits von B in der Prozessentwicklung nicht entfernt werden, was einen Verstoß gegen die Entwicklungsordnung darstellen würde.

Beispiel. Wir übersetzen per Projektion die White-Box-Variante der Preisberechnung von Holiday Airways ha_{2wb} in eine zusammengefasste Black-Box-Version ha_2, die wir bereits in der Begriffsbildung vorgestellt haben. Dazu beschränken wir den Anwendungsbereich der Regel auf die Ergebnisse der Preisberechnung, die in Abhängigkeit einer gegebenen Kundenanfrage ermittelt werden:

$$A = \{Inq(\langle Origin \rangle, \langle Destination \rangle, \langle Date \rangle), Fare(\langle Value \rangle)\}$$

Wir projizieren die detaillierte Regel ha_{2wb} auf den Anwendungsbereich A:

$$ha_2 = [ha_{2wb}]_A$$

Die Projektion entfernt damit die Aktionen *CalcDistFare*, *CalcPkSurchg*,*CalcDestSurchg* und *CalcTotalFare* aus den in der Regel betrachteten und generierten Prozessen und eliminiert auf diese Weise die Zwischenschritte der Flugpreisberechnung. Es gilt beispielsweise:

$$(\{e_{11}\}, \emptyset) \xrightarrow{ha_2} (\{e_{11}, e_{13}\}, \{(e_{11}, e_{13})\})$$

Die aggregierte Regel leitet direkt aus einer Kundenanfrage e_{11} den Grundpreis e_{13} ab, ohne dabei den Vorgang der Berechnung herzuleiten. □

4.3.3.3 Parametrierte Prozessregeln

Bei der Komposition von Prozessregeln kann es hilfreich sein, die gleiche Regel in bestimmten Abwandlungen mehrfach einzusetzen. Dies ist der Fall, wenn in Prozessen ein analog beschreibbares Verhalten zu beobachten ist, das nur in Details variiert werden muss.

Wir führen dazu die Parametrierung von Prozessregeln ein. Auf diese Weise können Regeln unabhängig von einem bestimmten Kontext definiert und dann mehrfach eingesetzt werden. Wir werden im weiteren Verlauf dieses Kapitels ein Beispiel für eine solche Mehrfachverwendung diskutieren.

Definition 23. Parametrierte Prozessregel

Wir bezeichnen eine parametrierte Prozessregel als eine Prozessregel, deren Bedingung r von einem oder mehreren Parametern $arg_1, arg_2, \ldots$ abhängt, und notieren diese folgendermaßen:

$$\xrightarrow{r(arg_1,arg_2,\ldots)} \in Rules$$

□

Zur Anwendung einer parametrierten Regel ist eine Belegung der Parameter erforderlich. Wir bezeichnen eine solche Parameterbelegung auch als Instanzierung der Regel.

Beispiel. In der Preisberechnung von Holiday Airways sind wir bisher davon ausgegangen, dass die entsprechende Prozessregel auf einen Entfernungstabelle zur Kalkulation des Grundpreises zurückgreifen kann. Wir können die Regel modifizieren, so dass die Entfernungstabelle als Parameter übergeben wird und somit die Abhängigkeit von dieser explizit dargestellt werden kann. □

4.3.4 Akteurbezug in Prozessen und Prozessregeln

In unserer bisherigen Diskussion sind wir nicht darauf eingegangen, von wem Aktionen durchgeführt werden und für wen die Durchführung dieser Aktionen sichtbar ist. Für eine differenzierte Betrachtung geschäftlicher Abläufe ist es jedoch essentiell, aktiv und passiv an den Handlungen beteiligte Akteure identifizieren zu können. Wir befassen unser daher im folgenden Abschnitt mit der expliziten Modellierung von Akteuren in geschäftlichen Abläufen.

4.3.4.1 Akteure

Wir richten unsere Auslegung des Akteurbegriffs an handelnden Entitäten in betrieblichen Abläufen ab.

Definition 24. Akteur

Wir betrachten einen Akteur als eine Entität, die an der Ausführung einer Aktion aktiv oder passiv beteiligt sein kann. Wir unterscheiden dabei zwischen Personen, die wir als manuelle Akteure bezeichnen, und Rechnersystemen, die wir als automatisierte Akteure ansehen. □

Wir bezeichnen die Menge aller manuellen und automatisierten Akteure als $Actors$. Unter manuellen Akteuren verstehen wir einzelne Personen, beispielsweise Mitarbeiter oder Kunden eines Unternehmens, sowie Personengruppen, wie zum Beispiel Organisationseinheiten eines Unternehmens. Unter automatisierten Akteuren verstehen wir Rechnersysteme, die ein Unternehmen betreibt. Wir führen die Modellierung von geschäftlichen Abläufen uniform für manuelle und automatisierte Akteure durch, so dass die Entscheidung, welche Abläufe durch IT-Systeme unterstützt werden sollen, nicht in der fachlichen Architektur antizipiert werden muss.

Beispiel. Für die Bearbeitung einer Buchungsanfrage bei Holiday Airways gehen wir von folgenden Akteuren aus:

- $HACustomers \subset Actors$ als die Menge der (manuellen) Akteure, die Kunden von Holiday Airways sind und Buchungsanfragen stellen
  - $cm \in HACustomers$ als den Kunden Herrn Müller
  - $ch \in HACustomers$ als die Kundin Frau Huber
- $HASystems \subset Actors$ als die Menge aller Rechnersysteme (automatisierte Akteure), die Holiday Airways zur Bearbeitung von Buchungsanfragen betreibt
  - $sb \in HASystems$ als das Buchungssystem, das auf Basis einer gegebenen Buchungsanfrage verfügbare Flugverbindungen ermittelt
  - $sb \in HASystems$ als das System, das Preisberechnungen durchführt
  - $so \in HASystems$ als das System, das Kundenangebote zusammenstellt □

Akteure können zueinander in Interaktion treten, indem sie Aktionen durchführen, die durch andere Akteure beobachtbar sind. Wir gehen davon aus, dass jede Aktion von einem Akteur als handelnde Entität ausgeht und von dem gleichen sowie anderen Akteuren beobachtet werden kann. Bei manuellen Akteuren kann jede Alltagstätigkeit als eine Aktion betrachtet werden – das Öffnen einer Tür, das Einsteigen in ein Auto oder auch das Kennenlernen einer Person. Im geschäftlichen Umfeld sind darunter insbesondere Arbeitsvorgänge zu verstehen, zum Beispiel die Ablage von Dokumenten oder die Prüfung einer Unterlage, sowie Interaktionen, beispielsweise die Übermittlung einer Anfrage oder die Beantwortung einer Frage. In unserem Praxisbeispiel untersuchen wir ausschließlich Handlungen, die von Rechnern erfasst und somit als Daten repräsentiert werden können.

Definition 25. Aktion (akteurbezogene Definition)

Wir betrachten eine Aktion als eine Handlung, die von einem Akteur einfach oder mehrfach durchgeführt und dabei von dem gleichen und/oder anderen Akteuren beobachtet werden kann. Formal stellen wir eine solche Aktion dar als $(id \star c, T) \in Actions$ mit:

$id \in Identifiers$	Ein eindeutiger Bezeichner zur Beschreibung der Aktion
$c \in Actors$	Ein ausführender Akteur, der die beschriebene Aktion durchführt
$T \subseteq Actors$	Eine Menge von beobachtenden Akteuren, für die die Aktion des ausführenden Akteurs sichtbar sind

□

Wir können nun die Betrachtung von Aktionen, die in Ereignissen, Prozessen und Prozessregeln auftreten, um Akteure erweitern. Wir verwenden im Folgenden als abkürzende Notation drei Funktionen $ActionsBy, ActionsFor, ActionsWith \in Actors \to \mathscr{P}(Actions)$, die die Menge der Aktionen angeben, die von einem bestimmten Akteur $act \in Actors$ durchgeführt werden, für diesen sichtbar sind beziehungsweise die Vereinigung aus beiden:

$$ActionsBy(act) = \{(id \star c, T) \in Actions : c = act\}$$

$$ActionsFor(act) = \{(id \star c, T) \in Actions : act \in T\}$$

$$ActionsWith(act) = ActionsBy(act) \cup ActionsFor(act)$$

Wir gehen davon aus, dass die drei Funktionen auch auf Mengen von Akteuren angewendet werden können und in diesem Fall die Vereinigung aller Aktionen, die durch diese Akteure ausgeführt beziehungsweise beobachtet werden, enthält.

Prozessregeln werden häufig aus der Perspektive eines Akteurs aufgestellt. Dies bedeutet, dass durch Anwendung einer Regel r nur Aktionen hinzugefügt werden, die durch den Akteur $act \in Actors$ initiiert werden: $Delta(r) \subseteq ActionsBy(act)$. Zudem enthält der Anwendungsbereich einer solchen Regel r typischerweise nur Aktionen, die der Akteur beobachten kann und/oder selbst durchführt: $Scope(r) \subseteq ActionsWith(act)$. Wir bezeichnen solche Regeln auch als akteurorientierte Regeln und werden diese in Abschnitt 4.4.3 ausführlicher diskutieren. Eine akteurorientierte Regel beschreibt, wie ein gegebener Akteur in Abhängigkeit der für ihn sichtbaren Aktionen neue Aktionen durchführt. Dies folgt der Intuition, dass sich Akteure nach bestimmten Regeln verhalten.

Die Beobachtung von Aktionen in Ereignissen versetzt einen Akteur in einen Zustand, da er auf Basis des Wissens über die Existenz bestimmter Ereignisse sein eigenes Verhalten, welches sich anhand der von ihm generierten Ereignissen erfassen lässt, anpassen kann. Wir verwenden somit Prozesse auch als Mittel zur Zustandsbeschreibung von Akteuren.

In der Formulierung der Prozessregeln gehen wir davon aus, dass sich der Zustand eines Akteurs rein aus den beobachteten Ereignissen ergibt und ein Akteur jenseits der Ereignisbeobachtung über keinen Speicher verfügt. Nichtsdestotrotz lassen sich damit zustandsbehaftete Akteure modellieren, indem wir davon ausgehen, dass ein solcher Akteur auf Basis eines definierten Startzustandes einen Prozess in mehreren Abschnitten beobachtet und wir für jeden Abschnitt einen Zustandsübergang erfassen. Wir haben ein solches Verhalten bereits im Beispiel der detaillierten Preisberechnung ansatzweise diskutiert, indem wir den Zustandsbegriff auf Basis von sechs Variablen aufgebaut haben. In ähnlicher Form lassen sich auch Datenbanken in unser Modell integrieren, deren Daten durch Ein- und Ausgabeaktionen modifiziert werden. Die explizite Erfassung von Zustandsübergängen auf Basis von beobachteten Aktionen vereinfacht die konsistente Definition des Modells, da Zustände explizit dargestellt und somit Abhängigkeiten leichter nachvollzogen werden können.

Beispiel. Wir können nun den Aktionen, die wir bis jetzt im Kontext einer Buchungsanfrage diskutiert haben, konkrete Akteure zuweisen. Wir nehmen zunächst eine Zuweisung exemplarisch für die Bearbeitung einer Anfrage durch den Kunden Herrn Müller *cm* vor:

(0) $Inq(\langle Orig\rangle, \langle Dest\rangle, \langle Date\rangle) \star cm, \{sb, sp\}$

(1) $Flight(\langle Date\rangle) \star sb, \{so\}$

(2) $Fare(\langle Value\rangle) \star sp, \{sp\}$

(3) $AmtDue(\langle Value\rangle) \star sp, \{so\}$

(4) $Offer(\langle Date\rangle, \langle AmtDue\rangle) \star so, \{cm\}$

Anhand der Zuweisung wird nun auch die Motivation für die White-Box-Definition der detaillierte Preisberechnung ha_{2wb} ersichtlich. Der Anwendungsbereich der Regel ist auf das Preisberechnungssystem von Holiday Airways sp beschränkt. Wir haben die Regel dahingehend definiert, dass sie die Aktion $Inq(\langle Orig\rangle, \langle Dest\rangle, \langle Date\rangle)$, die durch einen Kunden durchgeführt wird, nur beobachten, aber nicht selbst ausführen kann. Die Regel beschreibt, wie das Preissystem der Fluggesellschaft durch den Empfang einer Kundenanfrage in einen Zustand versetzt wird und welche Aktionen auf eine gegebene Anfrage seitens des Systems zu erfolgen haben.□

4.3.4.2 Akteurbezogene Instanzierung von Regeln

In unserer bisherigen Diskussion der Preisberechnung bei Holiday Airways haben wir lediglich die Bearbeitung einer einzelnen Kundenanfrage betrachtet. In der Praxis ist zu erwarten, dass eine Vielzahl solcher Anfragen auftreten. Wir werden uns daher im Folgenden mit der Instanzierung von Abläufen beschäftigen. Im Sinne der Anschaulichkeit führen wir die Instanzierung direkt am Praxisbeispiel ein. Die Überlegungen in diesem Abschnitt sind aber für Regeln jeder Art anwendbar.

Die aktuelle Definition der Prozessregel ha_2 sieht nur die Berechnung einer einzelnen Kundenanfrage vor und unterscheidet dabei nicht, von welchem Kunden aus der Menge *HACustomers* die Anfrage gestellt wird. Wir erweitern unsere Regeldefinition nun dahingehend, dass die Preisberechnung mehrfach durchgeführt werden kann und dabei gewährleistet ist, dass die Einzelberechnungen dem jeweils richtigen Kunden zugeordnet werden. Dazu übersetzen wir die Black-Box-Variante der Preisberechnung ha_2 in eine Regel $ha_2(c)$ mit $c \in HACustomers$. Die transformierte Regel $ha_2(c)$ bezieht sich nur noch auf die Anfrage eines bestimmten im Parameter spezifizierten Kunden c und stellt damit ausschließlich einen Zusammenhang zwischen Aktionen her, die diesen Kunden betreffen:

(0) $Inq(\langle Orig\rangle, \langle Dest\rangle, \langle Date\rangle) \star c, \{sb(c), sp(c)\}$

(2) $Fare(\langle Value\rangle) \star sp(c), \{sp(c)\}$

Dabei steht $sp(c)$ für eine Instanz des Buchungssystems sp, die ausschließlich mit der Bearbeitung der Kundenanfrage von c befasst ist. Die Regel $ha_2(c)$ leitet für eine Kundenanfrage, die vom Kunden c empfangen und an die entsprechende Instanz des Systems gerichtet ist, einen Flugpreis innerhalb der Instanz her. Auf diese Weise gewährleisten wir, dass die einzelnen Abläufe der unterschiedlichen Buchungsanfragen den Kunden richtig zugeordnet werden können. Wir unterstellen dabei, dass ein Dispatcher die Kundenanfragen an die entsprechenden Systeminstanzen vermittelt.

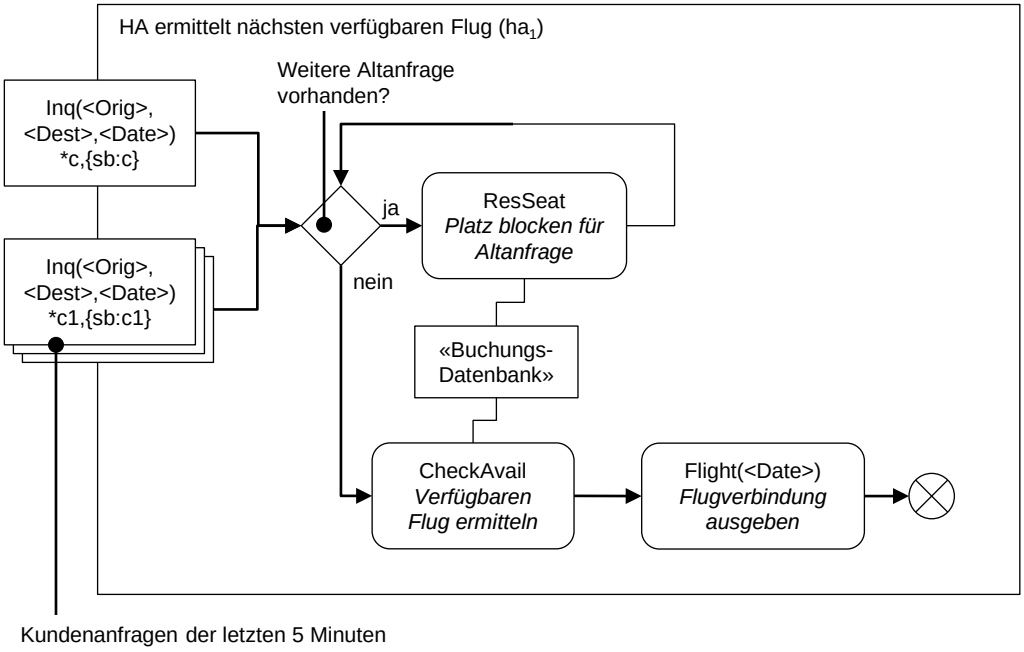

Abbildung 4.12: Spezifikation für die Ermittlung der nächsten verfügbaren Flugverbindung

Für den Anwendungsbereich der Regel gilt:

$$Scope(ha_2(c)) \subseteq ActionsWith(\{c, sp(c)\})$$

Aus der Menge der Regeln für die einzelnen Instanzen $ha_2(c)$ lässt sich durch Komposition eine Regel erzeugen, die die Preisberechnung für alle Kunden unabhängig voneinander durchführt:

$$\hat{\bigcup}_{c \in Customers} ha_2(c)$$

Dabei haben wir das Verhalten der Fluggesellschaft bis jetzt so beschrieben, dass die einzelnen Instanzen unabhängig voneinander operieren können und sich nicht gegenseitig beeinflussen. Ein solcher Ansatz vereinfacht das Aufstellen von Prozessregeln, da diese nur die Abhängigkeiten zwischen Aktionen im Einzelablauf beschreiben müssen. Allerdings können auf diese Weise keine Wechselwirkungen zwischen den Instanzen erfasst werden.

Wir untersuchen nun in informeller Weise die Prozessregel ha_1, mit der die nächstmögliche Verbindung für eine gegebene Buchungsanfrage ermittelt wird. Die Spezifikation, auf deren Basis wir die Prozessregel ha_1 definieren, ist in Abbildung 4.12 dargestellt. Wir definieren die Regel ebenfalls in Abhängigkeit eines Kunden als $ha_1(c)$. Wenn von diesem Kunden c eine Buchungsanfrage eintrifft, werden zunächst alle vorausgegangenen Anfragen anderer Kunden $c_1, \ldots, c_n$

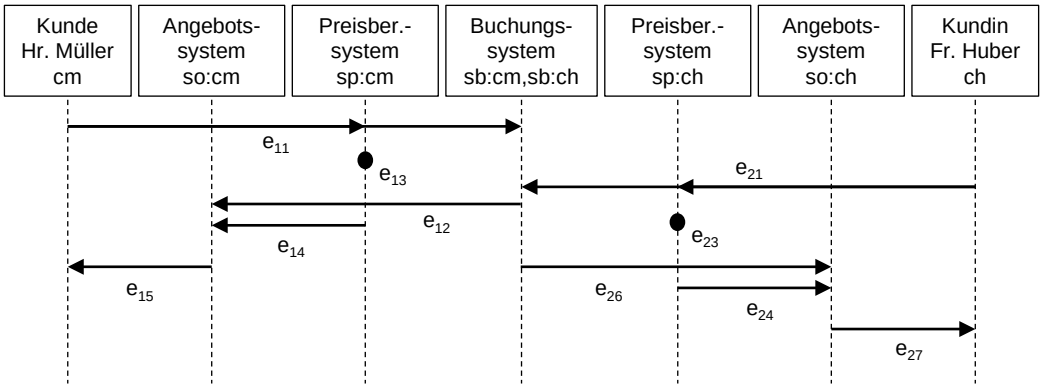

Abbildung 4.13: Ereignisfolge für zwei exemplarische Kundenanfragen

in den letzten 5 Minuten überprüft. Wir gehen davon aus, dass Zugriff auf eine Buchungsdatenbank besteht, in der alle Flüge mit den jeweils noch verfügbaren Plätzen gespeichert sind. Mit der Aktion *ResSeat* wird für jede vorausgegangene Anfrage, die sich auf die gleiche Flugverbindung wie die aktuelle Anfrage bezieht, ein Sitzplatz vorgemerkt. Nach Verarbeitung aller Vormerkungen wird die nächste Flugverbindung, auf der noch ein Platz verfügbar ist, mit der Aktion *CheckAvail* ermittelt. Diese wird zuletzt in der Aktion $Flight(\langle Date \rangle)$ ausgegeben.

Erneut können wir durch alternative Komposition eine Prozessregel angeben, die die Bearbeitung von Verfügbarkeitsanfragen für alle Kunden formuliert:

$$\hat{\bigg\uparrow}_{c \in Customers} ha_1(c)$$

Die Instanzen dieser Regel sind im Gegensatz zur oben betrachteten Preisberechnung nicht disjunkt, da für eine gegebene Kundenanfrage alle anderen Kundenanfragen der letzten 5 Minuten mit in Betracht gezogen werden. Das Ergebnis einer Verfügbarkeitsanfrage hängt damit auch von vorangegangenen Kundenanfragen ab.

Wir stellen nun eine komponierte Prozessregel für die gesamte Bearbeitung einer Buchungsanfrage auf:

$$ha = \hat{\bigg\uparrow}_{c \in Customers} (ha_1(c) \parallel (ha_2(c); ha_3(c))); ha_4(c)$$

Wir gehen davon aus, dass die Regeln ha_3 und ha_4 entsprechend der informellen Beschreibung im Kapitel 3 formalisiert sind. Die Regel ha_3 baut auf dem Ergebnis einer Grundpreisberechnung auf, addiert pauschal 30% für Steuern und Gebühren und gibt den Gesamtpreis als Aktion $AmtDue(\langle Date \rangle)$ aus. Die Regel ha_4 fasst eine ermittelte Flugverbindung und einen ermittelten Gesamtpreis zu einem Angebot zusammen und schickt dieses zurück an den Kunden als $Offer(\langle Date \rangle, \langle AmtDue \rangle)$.

Wir betrachten exemplarisch die zwei Buchungsanfragen der Kunden Müller und Huber anhand der Ereignisse, die in Tabelle 4.5 und Abbildung 4.13 dargestellt sind. Der Prozess M stellt den

	Ereignis	$e.t$	$e.a$		
Prozess M	e_{11}	1	$Inq(\text{MUC},\text{FRA},14.01.\#12{:}00)$	$\star cm,$	$\{sb(cm),sp(cm)\}$
	e_{12}	5	$Flight(14.01.\#12{:}15)$	$\star sb(cm),$	$\{so(cm)\}$
	e_{13}	3	$Fare(170)$	$\star sp(cm),$	$\{sp(cm)\}$
	e_{14}	6	$AmtDue(221)$	$\star sp(cm),$	$\{so(cm)\}$
	e_{15}	8	$Offer(14.01.\#12{:}15,221)$	$\star so(cm),$	$\{cm\}$
Prozess H_1	e_{21}	4	$Inq(\text{MUC},\text{FRA},14.01.\#12{:}00)$	$\star ch,$	$\{sb(ch),sp(ch)\}$
	e_{22}	8	$Flight(14.01.\#12{:}15)$	$\star sb(ch),$	$\{so(ch)\}$
	e_{23}	6	$Fare(170)$	$\star sp(ch),$	$\{sp(ch)\}$
	e_{24}	9	$AmtDue(221)$	$\star sp(ch),$	$\{so(ch)\}$
	e_{25}	11	$Offer(14.01.\#12{:}15,221)$	$\star so(ch),$	$\{ch\}$
Prozess H_2	e_{21}	4	$Inq(\text{MUC},\text{FRA},14.01.\#12{:}00)$	$\star ch,$	$\{sb(ch),sp(ch)\}$
	e_{26}	8	$Flight(14.01.\#13{:}15)$	$\star sb(ch),$	$\{so(ch)\}$
	e_{23}	6	$Fare(170)$	$\star sp(ch),$	$\{sp(ch)\}$
	e_{24}	9	$AmtDue(221)$	$\star sp(ch),$	$\{so(ch)\}$
	e_{27}	11	$Offer(14.01.\#13{:}15,221)$	$\star so(ch),$	$\{ch\}$

Tabelle 4.5: Ereignistabelle für drei exemplarische Anfragebearbeitungen

Ablauf der Buchungsanfrage von Herrn Müller dar. Die Prozesse H_1 und H_2 repräsentieren die Anfrage von Frau Huber in zwei Varianten. Wir gehen davon aus, dass vor den Anfragen dieser beiden Kunden nur noch ein Platz auf dem Flug München-Frankfurt am 14. Januar um 12:15 Uhr verfügbar ist. Der nächste verfügbare Flug ist eine Stunde später angesetzt. Wir stellen die Anfragen der Kunden mit $M_{inq} = \{e_{11}\}$ und $H_{inq} = \{e_{21}\}$ dar. Es gilt:

(1) $M_{inq} \xrightarrow{ha} M$

(2) $H_{inq} \xrightarrow{ha} H_1$

(3) $M_{inq} \cup H_{inq} \xrightarrow{ha^*} M \cup H_2$

Die Anwendung der Regel *ha* beschreibt die Bearbeitung der Kundenanfrage seitens Holiday Airways. In (1) und (2) stellt nur einer der beiden Kunden eine Anfrage und erhält jeweils den letzten Platz auf der Verbindung um 12:15 Uhr angeboten. In (3) stellen beide Kunden eine Anfrage für den gleichen Flug. In diesem Fall ist die temporale Ordnung der Ereignisse von Bedeutung. Da die Anfrage von Frau Huber nach der Anfrage von Herrn Müller erfolgt, ist der letzte Platz der 12:15 Uhr Maschine für Herrn Müller geblockt und Frau Huber wird die Verbindung um 13:15 Uhr angeboten.

Die Bearbeitung von Buchungsanfragen gemäß der Regel *ha* lässt sich nun auch in voller Allgemeinheit betrachten. Dazu definieren wir eine Spezifikation aller möglichen Kundenanfragen wie folgt:

$$Inq_{ha} = \mathscr{P}(\{e \in Events : \exists c \in HACustomers : e.a = (Inq(\_,\_,\_) \star c, \{sb(c), sp(c)\})\})$$

Durch die transitiv abgeschlossene Anwendung der Regel auf die Spezifikation der Kundenanfragen kann systematisch die Spezifikation der Anfragebearbeitung durch Holiday Airways hergeleitet werden:

$$Inq_{ha} \stackrel{ha^*}{\Longrightarrow} S_{ha}$$

Es gilt $M, H_1, M \cup H_2 \in S_{ha}$. Die Spezifikation Inq_{ha} bezeichnen wir auch als Prozessannahme. Sie stellt eine Annahme bezüglich des Umfeldes von Akteuren dar, das nicht regelbasiert hergeleitet werden kann. Entsprechend sind Prozessannahmen in der Regel generisch formuliert und enthalten eine Vielzahl möglicher Prozesse. In unserem Beispiel umfasst die Prozessannahme alle denkbaren Anfragekonstellationen der Kunden von Holiday Airways.

4.3.5 Zusammenfassung des konkreten Prozessmodells

In den letzten Abschnitten haben wir ein konkretes Prozessmodell zur Beschreibung von Handlungsabläufen hergeleitet. Die Darstellung von Abläufen basiert dabei auf Aktionen und Ereignissen. Unter einer Aktion verstehen wir eine Handlung, die mehrfach durchgeführt werden kann. Ein Ereignis repräsentiert die einmalige Ausführung einer solchen Aktion. Entsprechend kann jedem Ereignis eindeutig eine Aktion zugeordnet werden, die Umkehrung gilt nicht.

Wir fassen einen Prozess als eine Menge von Ereignissen auf. Dazu haben wir drei Repräsentationen eines Prozesses kennengelernt, denen unterschiedliche temporale Ordnungen zugrundeliegen. Ein ungeordneter Prozess erfasst Ereignisse unabhängig von ihren Auftrittszeitpunkten. In kausalen Prozessen treffen wir relative Aussagen zu den Auftrittszeitpunkten von Ereignissen in einer partiellen Ordnungsrelation. In temporalen Prozessen erfassen wir die Ereignisse, die im Prozess auftreten, explizit mit ihrem Eintrittszeitpunkt. Für alle drei Repräsentationen lässt sich der Begriff der Spezifikation als eine Menge von Prozessen mit einer zugrundeliegenden semantischen Aussage anwenden.

Mit Prozessregeln beschreiben wir die Entwicklung von Prozessen, indem wir Zusammenhänge zwischen beobachteten und daraus resultierenden Ereignissen festhalten. Prozessregeln fokussieren dabei typischerweise auf eine abgegrenzte Menge von Aktionen, zu denen sie Aussagen treffen. Wir bezeichnen diese Menge auch als Anwendungsbereich einer Regel. Durch Komposition kann aus mehreren Regeln mit einem begrenzten Anwendungsbereich sukzessive ein umfassendes Regelwerk konstruiert werden. Wir sprechen von einer Prozessregel mit erweitertem Anwendungsbereich oder kurz einer vollständigen Prozessregel, wenn diese in ihrem Anwendungsbereich Prozesse durch Generierung zusätzlicher Ereignisse entwickelt und sich außerhalb des Anwendungsbereichs indifferent bezüglich auftretender Ereignisse verhält. Durch Projektion kann der Anwendungsbereich einer Regel entsprechend angepasst werden.

Schließlich haben wir unsere Auslegung von Aktionen um Akteure ergänzt, so dass aktiv und passiv an den Abläufen beteiligte Personen und Systeme erfasst werden können. Dies ermöglicht auch die Modellierung von Instanzen, so dass ein Ablauf durch unterschiedliche Akteure zeitgleich oder zeitversetzt nach einer einheitlichen Spezifikation durchgeführt und entsprechend im Modell erfasst werden kann.

4.4 Weiterführende Konzepte

Neben dem Prozess haben wir bereits eine Vielzahl weiterer Konzepte, wie den Dienst, die Transaktion und den Anwendungsfall, kennengelernt, mit denen ebenfalls geschäftliche Abläufe beschrieben werden können. Wir werden in diesem Abschnitt nun untersuchen, welcher Zusammenhang zwischen den Konzepten besteht. Dabei betrachten wir den Prozess als Grundkonzept und werden zeigen, dass weitere Konzepte zur Erfassung geschäftlicher Abläufe als Sichten auf einen Prozess aufgefasst werden können.

Die unterschiedlichen Sichten ermöglichen dabei eine differenziertere und flexiblere Beschreibung geschäftlicher Abläufe. Durch die Aufrechterhaltung eines konsequenten Bezugs zum Konzept Prozess ermöglichen wir die Darstellung von komplexen geschäftlichen Handlungsabläufen in einer konsistenten Art und Weise.

4.4.1 Aktivitäten

Der Begriff einer Aktivität ist unterschiedlich auslegbar. Aktivitäten werden einerseits als elementare Einheiten oder auch Schritte betrachtet, aus denen sich ein Prozess zusammensetzt und die nicht weiter unterteilt werden können (vgl. [EHH+08]). In unserem Modell entspräche dies prinzipiell der Definition eines Ereignisses beziehungsweise der damit verbundenen Aktion.

Hierbei stellt sich allerdings die Frage, wann eine Handlung als elementar anzusehen ist. Jedes Ereignis und damit auch jede Aktion kann durch eine Abbildung $\xi \in \mathit{Views}$ in einer andere Sicht übersetzt werden, in der die Aktion durch eine Menge korrespondierender Aktionen repräsentiert wird, die eine weitere Zerlegung der Aktion in Teilaktionen realisieren. Umgekehrt kann jeder Prozess als Menge von durchgeführten Aktionen auch durch eine entsprechende Abbildung in einen anderen Prozess transformiert werden, in dem diese Aktionenmenge als eine einzelne Aktion auf einem höheren Abstraktionsniveau dargestellt wird. Diese Interpretation setzt den Gedanken der gegenseitigen Überführbarkeit der Begriffe Prozess und Aktivität um, den wir bereits in Abschnitt 2.1 diskutiert haben.

Wir betrachten eine Aktivität daher als Synonym zu einem Prozess. Die Frage, wann eine Handlung als elementar anzusehen ist, hängt dabei rein von der angewendeten Sichtweise auf einen Prozess beziehungsweise eine Aktivität ab. Es lässt sich jede elementare Handlung auch als Schrittfolge auffassen und umgekehrt kann jede Schrittfolge zu einer elementaren Handlung zusammengefasst werden.

4.4.2 Transaktionen

Wir haben uns bereits mit der akteurbezogenen Instanzierung von Prozessregeln befasst, die es in unserem Beispiel der Buchungsanfrage ermöglichte, die Bearbeitung von mehreren Buchungsanfragen in einem Prozess zu betrachten. Eine solche Instanzierung von Prozessregeln ist in der Praxis häufig von Relevanz, da Unternehmen typischerweise Interaktionen mit einer Vielzahl von Kunden und Geschäftspartnern unterhalten, denen einheitliche Interaktionsmuster

zugrundeliegen. In der Gesamtbetrachtung enthalten die Prozesse solcher Unternehmen mehrere Abläufe, die sowohl parallel als auch sequentiell stattfinden. Die einzelnen Abläufe, die sich aus den Instanzen der Prozessregel ergeben, basieren auf Aktionen, die als zusammengehörig betrachtet werden können, da sie im Rahmen der gleichen Instanz durchgeführt wurden. Wir bezeichnen eine Menge zusammengehöriger Aktionen auch als Aktionsfamilie. Zur präzisen Untersuchung solcher Abläufe führen wir das Konzept der Transaktion ein (vgl. [Bro98]).

Definition 26. Transaktion

Wir bezeichnen einen Prozess $P \in Processes$ als eine Transaktion im Hinblick auf eine Aktionsfamilie $A \subseteq Actions$, wenn der Anwendungsbereich des Prozesses vollständig in der Aktionsfamilie enthalten ist:

$$Scope(P) \subseteq A$$

□

Entsprechend stellt $Processes(A)$ die Menge aller Transaktionen für die Aktionsfamilie A dar. Prinzipiell kann jede Teilmenge von $Actions$ als Aktionsfamilie aufgefasst werden und die Transaktionseigenschaft im Hinblick auf diese Teilmenge überprüft werden. Wir fokussieren uns im Folgenden auf Transaktionen, die sich aus der akteurbezogenen Instanzierung von Prozessregeln ergeben. Unsere Überlegungen sind jedoch analog auch auf jede andere Transaktionsauslegung anwendbar.

Wenn wir einen Prozess mit Aktionen, die unterschiedlichen Instanzen einer Prozessregel zugeordnet werden können, in Transaktionen unterteilen möchten, so ist es erforderlich, dass die Aktionen entsprechend voneinander abgegrenzt werden können. Wir ordnen die Aktionen dazu einem Akteur zu, den wir auch als Referenzakteur bezeichnen. Im geschäftlichen Umfeld handelt es sich dabei typischerweise um den Auftraggeber, der eine Transaktion initiiert. Im Beispiel der Flugbuchung ist dies der Kunde. Alle Aktionen, an denen der Referenzakteur unmittelbar ausführend oder beobachtend beteiligt ist, können problemlos zugeordnet werden. Wir haben in der Diskussion des Beispiels allerdings auch Konstellationen kennengelernt, in denen die Fluggesellschaft intern Aktionen durchführt, die sich eindeutig auf die Anfrage eines Kunden beziehen, an denen der Kunde aber nicht direkt beteiligt ist, beispielsweise die Berechnung des Grundpreises. Wir haben dazu für einen Akteur $s \in Actors$ (in unserem Beispiel ein System von Holiday Airways) eine Notation $s(act)$ eingeführt, um kenntlich zu machen, dass der Akteur s in einer Instanz aktiv ist, die sich auf den Akteur $act \in Actors$ bezieht. Damit lässt sich die Menge aller Aktionen, die einen Bezug zu einem bestimmten Akteur $act \in Actors$ aufweisen, formal fassen anhand einer Funktion $Ref \in Actors \to \mathscr{P}(Actions)$ mit folgender Definition:

$$Ref(act) = ActionsWith(\{act\} \cup \{s(act) : s \in Actors \to Actors\})$$

Wir bezeichnen einen Prozess $T_{act} \in Processes$, in dem sich alle Aktionen auf den Akteur $act \in Actors$ beziehen mit $Scope(T_{act}) \subseteq Ref(act)$, als Transaktion des Akteurs act.

Definition 27. Transaktionsfluss

Wir bezeichnen einen Prozess, der Transaktionen mehrerer Akteure enthält, als Transaktionsfluss. □

Mit Transaktionsflüssen erfassen wir Bündel von Abläufen, die jeweils durch Transaktionen darstellbar sind und gleichzeitig oder zeitversetzt stattfinden können. Wir verwenden den Begriff Transaktionsfluss insbesondere in Abgrenzung zum Begriff Transaktion, um Prozesse zu bezeichnen, die mehrere Transaktionen enthalten. Die Transaktion eines Akteurs $act \in Actors$ kann aus einem Transaktionsfluss $P \in Processes$ durch Projektion extrahiert werden:

$$\Pi_{Ref(act)}(P)$$

Wir gehen in unserer Diskussion vereinfachend davon aus, dass sich Transaktionen auf einen einzelnen Akteur beziehen. Die Definition kann aber ohne Weiteres auf eine Menge von Akteuren erweitert werden. So können beispielsweise in Auktionsprozessen Transaktionen untersucht werden, die sich auf einen Anbieter und einen Bieter beziehen.

Beispiel. Wir betrachten den Prozess mit den zwei Buchungsanfragen von Herrn Müller und Frau Huber bei Holiday Airways, die wir in Abschnitt 4.3.4.2 formalisiert haben:

$$P = M \cup H_2$$

In Abbildung 4.14 ist der Prozess P in kausaler Weise dargestellt. Die Abbildung zeigt, dass jede Aktion im Prozess M und H_2 eindeutig einem der beiden Kunden zugeordnet werden kann und es sich somit um Transaktionen der Kunden handelt. Es gilt $Scope(M) \subseteq Ref(cm)$ und $Scope(H_2) \subseteq Ref(ch)$. Der Prozess P als Vereinigung der beiden Transaktionen stellt einen Transaktionsfluss dar.

Wir gehen davon aus, dass die Anfrage von Frau Huber binnen fünf Minuten nach der Anfrage von Herrn Müller eintrifft. Folglich ergibt sich eine Abhängigkeit zwischen der Anfrage von Herrn Müller e_{11} und der vorgeschlagenen Flugverbindung für Frau Huber e_{26}. Dies zeigt, dass sich Transaktionen, auch wenn sie eindeutig einem Akteur zugeordnet werden können, durchaus ein- oder auch gegenseitig beeinflussen können. Hätte die Transaktion von Herrn Müller nicht stattgefunden, so könnte Holiday Airways Frau Huber die Verbindung um 12:15 Uhr anbieten. □

Auch für Transaktionen können Spezifikationen betrachtet werden. Diese lassen sich auf Basis von Prozessregeln entwickeln, deren Anwendungsbereich auf den der Transaktion beschränkt ist.

Definition 28. Transaktionsregel

Wir bezeichnen eine Prozessregel r als eine Transaktionsregel mit Bezug auf einen Akteur $act \in Actors$, falls gilt:

$$Scope(r) \subseteq Ref(act)$$

□

Mit Transaktionsregeln können so Verhaltensaspekte innerhalb einer Transaktion beschrieben werden. Aus einer Prozessregel r kann durch Projektion eine Transaktionsregel für den Akteur $act \in Actors$ abgeleitet werden:

$$[r]_{Ref(act)}$$

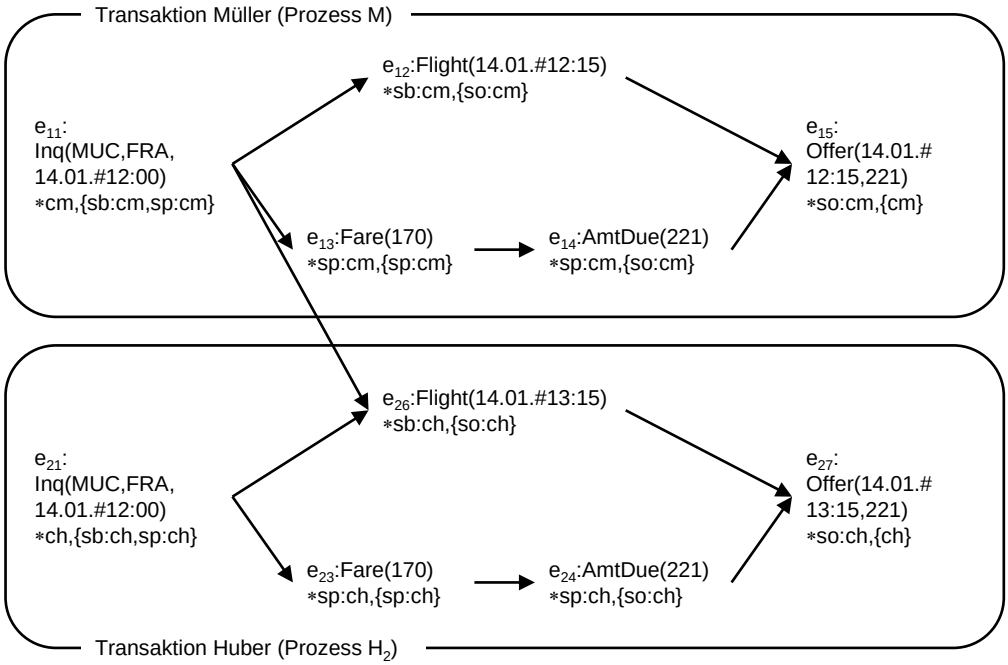

Abbildung 4.14: Transaktionsfluss, bestehend aus zwei Transaktionen

Eine so hergeleitete Transaktionsregel fokussiert die Entwicklungsaussage, die eine Prozessregel trifft, auf den Anwendungsbereich der Transaktion. Sie abstrahiert von sämtlichen Aktionen außerhalb der Transaktion, die wir auch als transaktionsfremde Aktionen bezeichnen. Durch die Projektion können Kausalitäten zu transaktionsfremden Aktionen nicht mehr explizit erfasst werden. Es werden daher im Zuge der Projektion in nicht deterministischer Weise sowohl die Existenz als auch die Nicht-Existenz von transaktionsfremden Aktionen angenommen. Falls Kausalitäten zwischen den Aktionen der Transaktion und transaktionsfremden Aktionen existieren, so bestehen Abhängigkeiten zwischen den Transaktionen.

Definition 29. Unabhängige Transaktionen

Wir bezeichnen eine Transaktion in einem Transaktionsfluss als unabhängig, wenn der Transaktionsfluss anhand einer Regel r beschrieben werden kann, so dass gilt:

$$r = \hat{\bigcap_{act \in Actors}} [r]_{Ref(act)}$$

□

Entsprechend bezeichnen wir Transaktionen, für die dies nicht gilt, als abhängig. In einem Transaktionsfluss mit unabhängigen Transaktionen hängen die generierten Ereignisse einer Transaktion ausschließlich von beobachteten Ereignissen der gleichen Transaktion ab. Somit kann eine Regel für einen solchen Transaktionsfluss einfach aus den Transaktionsregeln komponiert werden.

Dies gilt jedoch nicht für Transaktionsflüsse in voller Allgemeinheit. Das liegt darin begründet, dass bei abhängigen Transaktionen für deren Transaktionsregeln durch die Ausblendung transaktionsfremder Aktionen Annahmen bezüglich der ausgeblendeten Aktionen getroffen werden müssen. Bei der Komposition zweier solcher Transaktionsregeln kann es daher vorkommen, dass diesen gegenseitige Annahmen bezüglich der Existenz von Aktionen zugrundeliegen, die sich widersprechen. Bei unabhängigen Transaktionen dagegen sind solchen Annahmen irrelevant, da die Ereignisse einer Transaktion nur aus dieser selbst abgeleitet werden.

Beispiel. Wir greifen auf die Prozessregel *ha*, mit der wir die Bearbeitung einer Buchungsanfrage bei Holiday Airways beschrieben haben, zurück und bilden diese auf die Transaktion von Frau Huber ab:

$$[ha]_{Ref(ch)}$$

Die projizierte Regel beschreibt nur noch das Verhalten der Fluggesellschaft in Bezug auf die Kundenanfrage Huber. Da in der Anfragebearbeitung von Holiday Airways Abhängigkeiten zwischen den Transaktionen im Hinblick auf die Flugverfügbarkeit bestehen, müssen in der Projektion Annahmen zu anderen Transaktionen getroffen werden. Die projizierte Regel lässt damit folgende Anfragebearbeitungen zu:

- $H_{inq} \xrightarrow{[ha]_{Ref(ch)}} H_1$
- $H_{inq} \xrightarrow{[ha]_{Ref(ch)}} H_2$

Im Anwendungsbereich der Transaktionsregel kann nicht entschieden werden, ob dem Kunden die Verbindung um 12:15 Uhr angeboten werden kann. Die Regel lässt daher in nicht deterministischer Weise sowohl eine Ableitung der Prozesses H_1, in dem der Flug um 12:15 Uhr angeboten wird, als auch des Prozesses H_2, in dem der spätere Flug angeboten wird, zu. Die ursprüngliche Regel *ha* weist dagegen keinen Nichtdeterminismus dieser Art auf, da sie für eine gegebene Anfragekonstellation eindeutig entscheiden kann, welche Flüge den Kunden angeboten werden. Folglich kann *ha* nicht kompositiv aus den einzelnen Transaktionsregeln zusammengesetzt werden.

Der Teilprozess der Preisberechnung ist auf Basis der Regel ha_2 dagegen so ausgelegt, dass sich die Transaktionen nicht gegenseitig beeinflussen und somit unabhängig sind. Dies folgt direkt aus der Definition der Regel $ha_2 = \hat{\bigcap}_{c \in HACustomers} ha_2(c)$, da der Anwendungsbereich von $ha_2(c)$ ausschließlich auf Aktionen des Referenzakteurs c bezogen ist. □

4.4.3 Dienste

Neben der Fokussierung auf einzelne Transaktionen, wie wir sie soeben untersucht haben, ist auch die Betrachtung des Verhaltens einzelner Akteure bei der Analyse von Prozessen von hervorgehobenem Interesse. Damit lassen sich in arbeitsteiligen Konstellationen, wie sie im geschäftlichen Umfeld häufig zu finden sind, die Beiträge einzelner Akteure beziehungsweise von Gruppen von Akteuren zu Prozessen präzise erfassen. Auf diese Weise können auch Leistungsbeziehungen zwischen den Akteuren hergeleitet werden. Wir führen dazu das Konzept eines Dienstes ein. Ein Dienst beschreibt Aktionen, die von einer Gruppe von Akteuren durchzuführen sind, in Abhängigkeit von vorangegangenen Aktionen, die für diese Akteure beobachtbar waren.

Definition 30. Akteursgruppe

Wir bezeichnen eine Menge von Akteuren $Acts \subseteq Actors$ auch als Gruppe von Akteuren oder Akteursgruppe. □

Neben einzelnen Akteuren sind Gruppen von Akteuren insbesondere im geschäftlichen Umfeld relevant, da so Aussagen zum Verhalten ganzer Organisationen getroffen werden können. Im Beispiel der Flugbuchung haben wir bereits drei Akteure in der Organisation von Holiday Airways betrachtet: das Buchungssystem, das System zur Preisberechnung sowie das System zur Angebotserstellung. Wir können diese drei Akteure nun als Gruppe auffassen, um das Gesamtverhalten von Holiday Airways im vereinfachten Kontext unseres Beispiels darzustellen. In der Untersuchung einer Akteursgruppe unterscheiden wir zwischen internen Aktionen und Schnittstellenaktionen.

Definition 31. Interne Aktion

Für eine gegebene Akteursgruppe $Acts \subseteq Actors$ bezeichnen wir alle Aktionen als intern, die von einem Akteur der Gruppe durchgeführt werden und ausschließlich in der Gruppe sichtbar sind.

Formal erfassen wir interne Aktionen anhand einer Funktion $Int \in \mathscr{P}(Actors) \to \mathscr{P}(Actions)$ mit folgender Definition:

$$Int(Acts) = \{(id \star c, T) \in Actions : c \in Acts \wedge T \subseteq Acts\}$$

□

Neben den internen Aktionen sind für eine Gruppe von Akteuren auch solche Aktionen relevant, an denen Akteure außerhalb der Gruppe aktiv oder passiv beteiligt sind. Solche Aktionen nennen wir Schnittstellenaktionen.

Definition 32. Schnittstellenaktion

Für eine gegebene Akteursgruppe $Acts \subseteq Actors$ verstehen wir unter einer Schnittstellenaktion eine nicht interne Aktion, an der ein oder mehrere Akteure der Gruppe aktiv oder passiv beteiligt sind. Formal erfassen wir Schnittstellenaktionen anhand einer Funktion $If \in \mathscr{P}(Actors) \to \mathscr{P}(Actions)$, die wie folgt definiert ist:

$$If(Acts) = ActionsWith(Acts) \setminus Int(Acts)$$

Wir differenzieren zwischen eingehenden und ausgehenden Schnittstellenaktionen und definieren dazu zwei Funktionen $In, Out \in \mathscr{P}(Actors) \to \mathscr{P}(Actions)$ wie folgt:

$$In(Acts) = ActionsFor(Acts) \setminus Int(Acts)$$

$$Out(Acts) = ActionsBy(Acts) \setminus Int(Acts)$$

□

Für eine gegebene Akteursgruppe $Acts \subseteq Actors$ sind die Aktionsmengen $In(Acts)$ und $Out(Acts)$ nicht disjunkt, da generierte Aktionen auch wieder beobachtet werden können. Sie ergeben aber in Vereinigung die Menge der Schnittstellenaktionen: $If(Acts) = In(Acts) \cup Out(Acts)$. Alle Aktionen, die für eine Akteursgruppe weder interne Aktionen noch Schnittstellenaktionen sind, bezeichnen wir als extern. Wir verzichten an dieser Stelle jedoch auf eine formale Definition. Wir bezeichnen Ereignisse, in denen interne Aktionen oder Schnittstellenaktionen durchgeführt werden, analog als interne Ereignisse beziehungsweise Schnittstellenereignisse.

Prozessregeln können im Allgemeinen akteurübergreifend formuliert sein. Sie können durchzuführende Aktionen für bestimmte Akteure beschreiben in Abhängigkeit von Aktionen, die durch die gleichen oder auch andere Akteure erfasst wurden. Um den Beitrag einzelner Akteure schärfer abzugrenzen, fokussieren wir zunächst die Formulierung von Prozessregeln dahingehend, dass sie sich nur noch auf eine bestimmte Gruppe von Akteuren beziehen. Wir bezeichnen eine unter dieser Maßgabe formulierte Regel als akteurorientierte Prozessregel.

Definition 33. Akteurorientierte Prozessregel

Wir verstehen eine Prozessregel r als eine akteurorientierte Regel für eine Gruppe von Akteuren $Acts \subseteq Actors$, falls gilt:

$$Scope(r) \subseteq ActionsWith(Acts) \wedge Delta(r) \subseteq ActionsBy(Acts)$$

□

Akteurorientierte Regeln beschreiben somit das Akteursverhalten ausschließlich anhand von Aktionen, an denen der Akteur aktiv oder passiv beteiligt ist. Aus einer akteurübergreifenden Regel r kann durch Projektion eine akteurorientierte Regel für eine Gruppe von Akteuren $Acts \subseteq Actors$ wie folgt abgeleitet werden:

$$[r]_{ActionsWith(Acts),ActionsBy(Acts)}$$

Durch die Beschränkung des Anwendungsbereichs in der Projektion können potenziell nicht mehr alle Kausalitäten durch die Regel erfasst werden. Dies betrifft alle Aktionen, die kausal von Aktionen abhängen, die nicht durch die Gruppe von Akteuren beobachtet werden können. Im Zuge der Projektion müssen in diesem Fall Annahmen zur Existenz beziehungsweise Nicht-Existenz solcher für die Gruppe nicht sichtbaren Aktionen getroffen werden.

Für die Erfassung des Beitrags einer Gruppe von Akteuren zu einem Prozess können wir nun die Betrachtung noch weiter fokussieren, indem wir alle internen Aktionen der Gruppe ausblenden und folglich nur noch Schnittstellenaktionen betrachten. Auf diese Weise beschreiben wir den für andere Akteure sichtbaren Beitrag einer Akteursgruppe zu einem Prozess in Abhängigkeiten von Aktionen, die diese Akteursgruppe zuvor von anderen Akteuren beobachten konnte. Wir formulieren damit eine Leistungsbeziehung zwischen Akteuren, die wir als Dienstleistung bezeichnen.

Definition 34. Dienst

Wir bezeichnen eine Prozessregel r als eine Dienstleistung, kurz Dienst (Service), einer Gruppe von Akteuren $Acts \subseteq Actors$, falls gilt:

$$Scope(r) \subseteq If(Acts) \wedge Delta(r) \subseteq Out(Acts)$$

□

Ein Dienst beschreibt das Verhalten einer Akteursgruppe aus einer Black-Box-Perspektive an ihrer Schnittstelle. Wir sprechen in diesem Zusammenhang auch von einem Schnittstellenverhalten der Gruppe. Der Dienst einer Akteursgruppe $Acts \subseteq Actors$ kann ebenfalls durch Projektion aus einer Prozessregel r in voller Allgemeinheit abgeleitet werden:

$$[r]_{If(Acts),Out(Acts)}$$

Wir benutzen im Folgenden für eine solche Projektion einer Regel r auf den Dienst einer Akteursgruppe $Acts$ als abkürzende Notation:

$$[r]_{\Cap Acts} = [r]_{If(Acts),Out(Acts)}$$

Das Symbol $\Cap$ steht dabei für die Schnittstelle und wird gefolgt von der Akteursgruppe, für die eine Schnittstellenperspektive angegeben wird.

In komplexen arbeitsteiligen Konstellationen können auf Basis von Diensten systematisch die Beiträge einzelner Akteure erfasst werden und somit das Gesamtverhalten einer größeren Gruppe von Akteuren aus dem Verhalten der Einzelakteure zusammengesetzt werden. Durch die

Black-Box-Perspektive eines Dienstes erfolgt dabei eine Abstraktion von den Vorgängen zur Erbringung der Leistungen der einzelnen Akteure zugunsten einer ergebnisorientierten Beschreibung.

Zur genauen Analyse des Schnittstellenverhaltens differenzieren wir zwischen den eingehenden und den ausgehenden Schnittstellenaktionen. Die eingehenden Schnittstellenaktionen können dabei ursächlich für die ausgehenden Schnittstellenaktionen der betrachteten Akteursgruppe sein. Wir bezeichnen daher die eingehenden Aktionen auch als Dienstauftrag und die ausgehenden Aktionen als Wirkung des Dienstes einer Akteursgruppe.

Definition 35. Dienstauftrag

Wir bezeichnen einen Prozess $P \in Processes$ als eine Dienstauftrag in Bezug auf eine Gruppe von Akteuren $Acts \subseteq Actors$, wenn der Anwendungsbereich des Prozesses ausschließlich eingehende Schnittstellenaktionen für diese Akteursgruppe umfasst:

$$Scope(P) \subseteq In(Acts)$$

□

Die Definition einer Dienstwirkung kann in analoger Weise erfolgen.

Definition 36. Dienstwirkung

Wir bezeichnen einen Prozess $P \in Processes$ als die Wirkung eines Dienstes einer Akteursgruppe $Acts \subseteq Actors$, wenn der Anwendungsbereich des Prozesses ausschließlich ausgehende Schnittstellenaktionen für diese Akteursgruppe umfasst:

$$Scope(P) \subseteq Out(Acts)$$

□

Ein Dienst beschreibt den Zusammenhang zwischen Dienstauftrag und dessen Wirkung. Er wird daher auch als Dienstvertrag aufgefasst. Die Gruppe von Akteuren, die einen Dienst erbringt, bezeichnen wir als Diensterbringer, Dienstanbieter oder Dienstgeber. Die Akteursgruppe, die die durch den Dienstgeber erbrachten Aktionen beobachten kann, bezeichnen wir als Dienstnehmer oder Dienstnutzer. Die Black-Box-Perspektive eines Dienstes auf das Verhalten einer Akteursgruppe kann sowohl im Interesse des Dienstnehmers sein, für den die interne Abläufe des Dienstgebers nicht relevant sind, als auch für den Diensterbringer, der potenziell seine internen Abläufe einem externen Akteur gegenüber nicht offenlegen möchte.

Beispiel. Wir kommen zurück auf unser Beispiel der Flugbuchung und untersuchen nun den Dienst, den Holiday Airways seinen Kunden gegenüber erbringt. Dazu definieren wir zunächst eine Akteursgruppe, die die Organisation von Holiday Airways repräsentiert:

$$HA = \{sb(c), sp(c), so(c) : c \in HACustomers\}$$

Die Akteursgruppe HA enthält alle Instanzen der drei Systeme sb, sp, so, die wir im Kontext der Anfragebearbeitung untersucht haben. Zur formalen Beschreibung des Dienstes dieser Gruppe

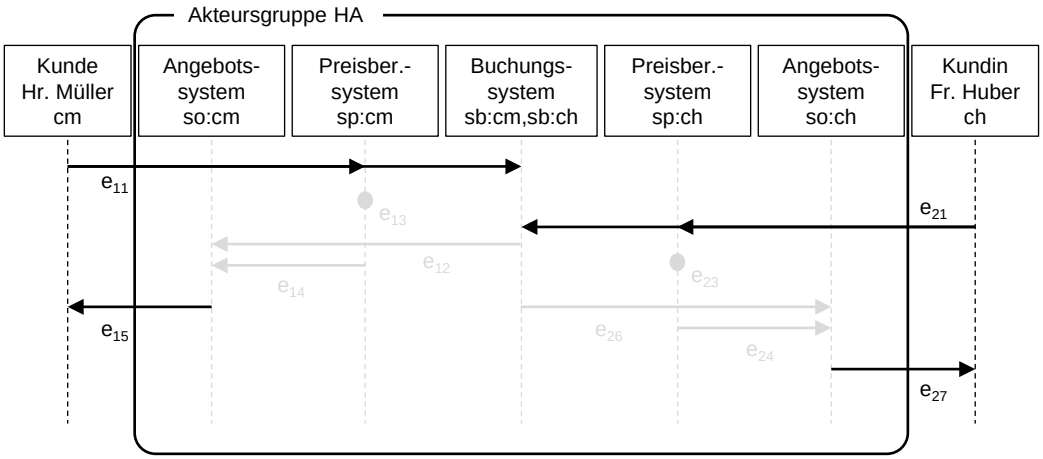

Abbildung 4.15: Schnittstellenbetrachtung zweier Kundenanfragen

projizieren wir die zuvor aufgestellte Prozessregel *ha*, die die Anfragebearbeitung der Fluggesellschaft mitsamt internen Vorgängen definiert, auf die Schnittstelle der Akteursgruppe *HA*:

$$sha = [ha]_{\Uparrow HA}$$

Die dienstorientierte Regel *sha* abstrahiert von sämtlichen internen Vorgängen der Fluggesellschaft und leitet stattdessen aus einer gegebenen Kundenanfrage direkt den Dienst der Fluggesellschaft ab, der aus der Anfrage resultiert.

Wir betrachten nun erneut die zwei Buchungsanfragen der Kunden Müller und Huber, die wir in Abschnitt 4.3.4.2 ausführlich diskutiert haben, in Abbildung 4.15. Aus der Anschauung ergeben sich unmittelbar die internen Ereignisse, die wir grau dargestellt haben, sowie die eingehenden und ausgehenden Schnittstellenereignisse. Es gilt unter anderem:

(1) $M_{inq} \xrightarrow{sha} M_{inq} \cup \{e_{15}\}$

(2) $H_{inq} \xrightarrow{sha} H_{inq} \cup \{e_{25}\}$

(3) $M_{inq} \cup H_{inq} \xrightarrow{sha^*} M_{inq} \cup H_{inq} \cup \{e_{15}, e_{27}\}$

Die Prozesse M_{inq} und H_{inq} sind die Dienstaufträge der beiden Kunden. Die Ereignisse e_{15}, e_{25}, e_{27} stellen als Prozesse betrachtet die Wirkungen des Dienstes der Fluggesellschaft dar, da es sich bei allen drei um ausgehende Schnittstellenereignisse handelt. Die Leistung, die die Fluggesellschaft ihren Kunden gegenüber erbringt, ist die Beantwortung der zuvor gestellten Buchungsanfragen (Aktion $Offer$). In den Fällen (1) und (2) liegt jeweils nur eine Anfrage eines Kunden vor und der Dienst beschreibt entsprechend die Bearbeitung dieser aus einer Black-Box-Sicht. Im Fall (3) fragen beiden Kunden zugleich an und erhalten entsprechende Angebote der Fluggesellschaft. □

Häufig wird ein Dienst auf Grundlage eines Auftrags erbracht, der vom Dienstnehmer ausgeht. So verschickt beispielshalber Holiday Airways als Dienstgeber Flugangebote an seine Kunden als Dienstnehmer in Abhängigkeit der Anfragen, die von eben diesen Kunden empfangen wurden. Jedoch können die Wirkungen eines Dienstes auch von Aktionen abhängen, die nicht vom Dienstnehmer ausgehen.

Der im Beispiel hergeleitete Dienst hat das Gesamtverhalten der Fluggesellschaft gegenüber allen Kunden erfasst. Es ist auch möglich, einen Dienst aus einer Transaktionsregel abzuleiten und damit nur das Verhalten von Holiday Airways gegenüber einem Kunden, auf den sich die zugrundeliegende Transaktion bezieht, zu beschreiben.

Unseren Überlegungen aus den letzten Abschnitten folgend kann eine transaktionsbezogene Dienstbeschreibung nicht deterministische Züge aufweisen, da Abhängigkeiten zu anderen Kundenanfragen nicht mehr erfasst werden können. Eine solche Beeinflussung, die nicht an der Schnittstelle eines Dienstes erfolgt ist, wird auch als Seiteneffekt bezeichnet. Betrachten wir beispielsweise ausschließlich den Dienst, den Holiday Airways gegenüber dem Kunden Huber erbringt, so stellt eine potenziell vorausgegangene Anfrage vom Kunden Müller, die die Verfügbarkeit entsprechend beeinflusst, einen Seiteneffekt für die Diensterbringung gegenüber Frau Huber dar.

4.4.4 Anwendungsfälle

In den letzten Abschnitten haben wir ausführlich die semantische Beschreibung von Prozessen anhand von Spezifikationen und Regeln diskutiert und sind dabei insbesondere auch auf verschiedene Projektionen eingegangen, um eine differenzierte Betrachtung geschäftlicher Abläufe zu ermöglichen.

Allerdings haben wir bislang alle Prozesse als gleichwertige Abläufe angesehen. Dies folgt dem Ansatz einer Trace Semantik, die die einfachste Form der semantischen Beschreibung darstellt. In einer solchen Semantik werden typischerweise auch unvollständige Abläufe als Teil einer Spezifikation angesehen. So ist es im Beispiel der Flugbuchung möglich, auf Basis der Regel *ha* einen Prozess zu entwickeln, der nur eine Preisberechnung, aber noch keine Verfügbarkeitsüberprüfung enthält.

Im betrieblichen Umfeld sind Geschäftsprozesse in der Regel auf die Erreichung eines Ziels, auch Geschäftsziels, ausgerichtet. [EHH+08] Damit kann zwischen der erfolgreichen Ausführung eines Prozesses, bei der das gesetzte Ziel erreicht wird, und der nicht erfolgreichen Ausführung unterschieden wird. Die Motivation für die Formulierung von Zielen im betrieblichen Kontext ist dabei häufig auf den Anspruch zurückzuführen, dass unternehmerisches Handeln Wert für das Unternehmen selbst und/oder dessen Kunden und Partner stiftet. Entsprechend wird die Ausführung geschäftlicher Handlungen, die in Prozessen modelliert sind, gemessen an der Erreichung eines entsprechenden Wertbeitrags. Im Beispiel der Buchungsanfrage könnte unter anderem die Übermittlung eines Flugangebots als Ziel einer Buchungsanfrage angesehen werden.

Wir werden die Betrachtung von Prozessspezifikationen nun weiter differenzieren, um die Erreichung solcher Ziele formal erfassen zu können. Dazu führen wir das Konzept des Anwendungsfalls ein, für das die Ausrichtung an einem Ziel charakteristisch ist. [Coc01]

Definition 37. Anwendungsfall

Wir betrachten eine Spezifikation, in der Prozesse nach Erreichung eines gesetzten Ziels differenziert werden, als einen Anwendungsfall. Formal lässt sich ein Anwendungsfall erfassen als ein Spezifikationstupel

$$(S, S_T) \in Spec \times Spec$$

Dabei stellt S die Spezifikation der Prozesse im Anwendungsfall dar und S_T gibt die Menge alle in der Spezifikation enthaltenen Prozesse an, die das gesetzte Ziel erreichen. Es gilt $S_T \subseteq S$. □

Wir bezeichnen die Menge aller Anwendungsfälle als $UCases$. Anwendungsfälle, wie sie in der Literatur behandelt werden (vgl. [Coc01]), fokussieren typischerweise auf die Beschreibung interaktiver Prozesse. Wir wählen eine breitere Auslegung des Begriffs und lassen jede Form von Prozess zu, da wir interaktive Abläufe als Spezialfall von Prozessen in voller Allgemeinheit betrachten und alle Merkmale eines Anwendungsfalls in der interaktiven Auslegung problemlos auch auf nicht interaktive Abläufe ausgeweitet werden können.

Die Beschreibung eines Anwendungsfalls nach [Coc01] basiert auf der Erfassung von Szenarien. Ein Szenario wird dabei als Folge von Interaktionen zwischen Akteuren, die auch als Stakeholder bezeichnet werden, angesehen. Somit können Szenarien mit unserem Prozessbegriff modelliert werden, wenn wir die zwischen den Stakeholdern ausgetauschten Nachrichten als Aktionen auffassen. Der Anwendungsfall als Menge aller zulässigen Szenarien entspricht prinzipiell einer Prozessspezifikation im Sinne unseres Modells. Jedoch differenzieren Anwendungsfälle zwischen erfolgreichen und nicht erfolgreichen Szenarien. Wir definieren daher einen Anwendungsfall als Spezifikationstupel $(S, S_T) \in UCases$. Dabei betrachten wir einen Prozess $P \in S$ als Szenario und einen Prozess $Q \in S_T$ als Erfolgsszenario des Anwendungsfalls. Die Spezifikation S bezeichnen wir auch als Basisspezifikation und S_T als Erfolgsspezifikation eines Anwendungsfalls.

Für die Beschreibung einer Basisspezifikation gelten die Überlegungen der letzten Abschnitte in analoger Weise. Es stellt sich nun die Frage, wie ein Prozess auf die Erfüllung eines gesetzten Ziels hin untersucht und eine korrespondierende Erfolgsspezifikation aufgestellt werden kann. Wir fassen ein Ziel als eine Anforderung an einen Prozess auf, die sich in Form eines Prädikats ausdrücken lässt. Dazu modellieren wir das Ziel formal als Prädikat $Obj \in Processes \to Bool$. Die Erfolgsspezifikation kann aus einer Basisspezifikation $S \in Spec$ anhand eines entsprechenden Ziels hergeleitet werden als:

$$S_T = \{P \in S : Obj(P)\}$$

Die Formulierung von Zielen ist auch entlang der bereits eingeführten Beschreibungsansätze für Spezifikationen möglich. In der operationellen Semantik kann für eine Automatendefinition

$(St, s_0, \rho) \in Machines$ eine Menge von Zielzuständen $St_T \subseteq St$ angegeben werden. Die Spezifikation S entspricht dabei $OpS(M)$. Die Menge von Erfolgsszenarien S_T ergibt sich aus den Spuren, die in einem Zielzustand enden:

$$\{a_1 \circ \ldots \circ a_n : \exists s_1, \ldots, s_{n-1} \in St, s_n \in St_T : (s_0, a_1, s_1), \ldots, (s_{n-1}, a_n, s_n) \in \rho\}$$

Es gilt offensichtlich $S_T \subseteq S$, da die Spuren, die in Zielzuständen enden, eine Teilmenge aller Spuren darstellen. Entsprechend können Prozessregeln verfeinert werden zu zielorientierten Regeln, die eine Ableitung weitere Aktionen nur zulassen, wenn der resultierende Prozess in der Erfolgsspezifikation enthalten ist.

Beispiel. Wir betrachten erneut die detaillierte Variante der Preisberechnung, die wir auf Basis eines Automatenmodells $HA_2 = (St, s_0, \rho)$ im Abschnitt 4.3.2.2 definiert haben. In den Transitionsregeln des Automaten haben wir bereits einen Zustand $\diamondsuit$ eingeführt, der nach Ausgabe eines berechneten Flugpreises (Aktion $Fare(\langle AmtDue \rangle)$) erreicht wird und damit eine abgeschlossene Preisberechnung repräsentiert. Folglich können wir die Menge der Zielzustände einfach definieren als $St_T = \{\diamondsuit\}$.

Wir können nun die Preisberechnung von Holiday Airways als Anwendungsfall (S, S_T) betrachten. Die Spezifikation S entspricht der operationellen Semantik des Automaten $OpS(HA_2)$ und beinhaltet damit die Menge aller Abläufe zur Preisberechnung. Insbesondere enthält S auch noch nicht abgeschlossene Preisberechnungen. So ist beispielsweise die Kundenanfrage von Herrn Müller M_{inq}, auf die Holiday Airways noch nicht reagiert hat, ebenfalls Teil der Spezifikation. Die Spezifikation S_T enthält alle Prozesse aus S, nach deren Ausführung der zugrundeliegende Automat HA_2 den Zielzustand $\diamondsuit$ erreicht hat. Wir betrachten S_T als Erfolgsszenarien der Preisberechnung, da für alle Prozesse in dieser Menge Holiday Airways einen Flugpreis für eine gegebene Kundenanfrage vollständig ermittelt hat. □

Neben der Unterscheidung nach Erreichung eines gesetzten Ziels können Spezifikationen auch in anderer Weise weiter differenziert werden. In der Definition eines Anwendungsfalls werden neben der Klassifizierung von Erfolgsszenarien häufig auch Fehlerszenarien explizit beschrieben. Der Beschreibungsansatz folgt dabei dem gleichen Prinzip wie bei den Erfolgsszenarien. Für eine systematische Diskussion weiterer semantischer Klassen verweisen wir auf [Gla90].

4.5 Architekturmodell

Eine Architektur beschreibt die Elemente eines Systems sowie deren Beziehungen zueinander (vgl. Abschnitt 2.1 für eine ausführlichere Diskussion). Wir haben bereits festgestellt, dass sich der Architekturbegriff in verschiedenen Kontexten anwenden lässt. Wir werden uns im Folgenden nun mit der Architektur von Prozessen befassen und dabei insbesondere untersuchen, wie Prozesse anhand von Diensten strukturiert werden können im Sinne einer dienstorientierten Architektur.

4.5.1 Prozessarchitektur

Um den per se generischen Architekturbegriff auf Prozesse anwenden zu können, müssen wir zunächst klären, was im Kontext von Prozessen unter den Elementen und Beziehungen einer Architektur zu verstehen ist. Wir haben in unserem Modell eine ereignisorientierte Darstellung von Prozessen gewählt. Den Ereignissen liegen wiederum Aktionen zugrunde, die in einem Ereignis ausgeführt werden. Wir betrachten daher die Aktionen als Elemente einer Prozessarchitektur.

Die Beziehungen zwischen den Aktionen ergeben sich aus der Spezifikation eines Prozesses. Da Abhängigkeiten zwischen Aktionen typischerweise vom Kontext der Aktionsdurchführung abhängen, untersuchen wir diese auf der Ereignisebene. Wir haben solche Beziehungen als Kausalitäten zwischen Ereignissen bereits ausführlich in Abschnitt 4.3.3.1 diskutiert. Wir setzen voraus, dass Prozesse auf Basis von Regeln beschrieben sind, da wir so zwischen ursächlichen Ereignissen und den daraus resultierenden Wirkungen differenzieren können. Kausalitäten können dabei nur in Prozessrepräsentationen festgestellt werden, die ein Zeitmodell besitzen. Wir fokussieren die folgende Diskussion auf kausale Prozesse. Unsere Überlegungen sind übertragbar auf temporale Prozesse, sind allerdings nicht anwendbar für ungeordnete Prozesse auf Grund des fehlenden Zeitmodells.

Definition 38. Prozessarchitektur

Mit Prozessarchitekturen beschreiben wir den Aufbau von Prozessen. Die Architektur eines Prozess resultiert dabei aus den Regeln, die zur Prozessdefinition verwendet werden. Wir verstehen unter einer Prozessarchitektur die Menge aller in einem Prozess vorkommenden Aktionen (Elemente der Architektur) sowie die Menge aller Kausalitäten zwischen den Aktionen (Beziehungen zwischen den Architekturelementen). □

Während die Aktionen als Elemente der Prozessarchitektur direkt aus der Betrachtung eines einzelnen Prozesses ersichtlich werden, ist zur Analyse der Kausalitäten zwischen den Aktionen eine Kenntnis der Prozessregeln, auf Basis derer der Prozess beschrieben ist, erforderlich. Da Kausalitäten auf Ereignisebene erfasst werden, bestehen in Prozessen typischerweise eine Vielzahl von Kausalitäten. Es ist daher nicht immer praktikabel, im Rahmen der Architekturbetrachtung die Menge aller Kausalitäten herzuleiten. Stattdessen können auch durch Approximationen bereits relevante Aussagen zur Prozessarchitektur getroffen werden.

Wir betrachten dazu im Folgenden den Anwendungsbereich von Regeln sowie die *Delta* Funktion als die Menge aller Aktionen, die nach Maßgabe der Regel durchzuführen sind. Der Anwendungsbereich und die *Delta* Funktion bilden zusammen eine obere Schranke für mögliche Kausalitäten, die sich aus einer Regelanwendung ergeben können. Bei der Anwendung einer Regel r können nur Kausalitäten zwischen zwei Ereignissen $e_1, e_2 \in Events$ entstehen, für die gilt $e_1.a \in Scope(r)$ und $e_2.a \in Delta(r)$. Selbstverständlich ist eine solche Kausalität nicht zwingend. Wenngleich diese obere Schranke nur eine grobe Approximation möglicher Kausalitäten darstellt, trägt sie doch dazu bei, ein intuitives Verständnis für den Aufbau eines Prozesses zu erlangen.

Dies lässt sich am besten anhand eines schematischen Beispiels diskutieren. Wir haben in den Abbildungen 4.16 und 4.17 zwei kausale Prozesse dargestellt, die hinsichtlich ihrer temporalen

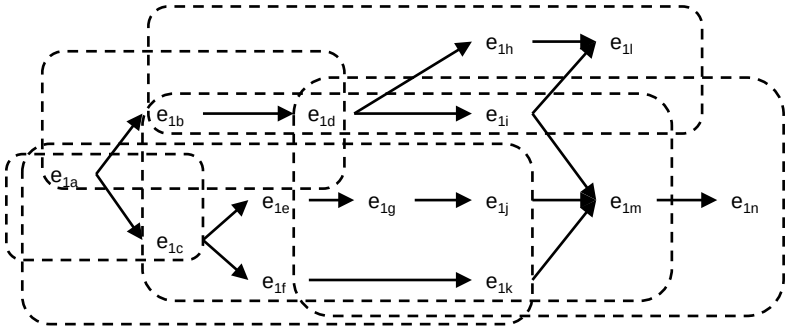

Abbildung 4.16: Schematische Darstellung eines Prozesses mit einer verschachtelten Regelarchitektur (die gestrichelten Abgrenzungen entsprechen den Anwendungsbereichen der eingesetzten Prozessregeln)

Ereignisstruktur kongruent sind, jedoch durch unterschiedliche Regeln beschrieben und damit in unterschiedlicher Weise aufgebaut werden. Die Anwendungsbereiche der Regeln, die zur Beschreibung der Prozesse eingesetzt werden, sind jeweils mit gestrichelten Linien gekennzeichnet.

Aus der Anschauung der beiden Prozesse werden die unterschiedlichen Prozessarchitekturen unmittelbar deutlich. Sie sind zurückzuführen auf unterschiedliche Kausalstrukturen. Der erste Prozess in Abbildung 4.16 wird auf Basis von Regeln beschrieben, die eine größere Menge vorausgegangener Ereignisse mit berücksichtigen, um weitere Aktionen abzuleiten. Daraus ergibt sich die in der Grafik deutlich zu erkennende starke Verschachtelung der Anwendungsbereiche der Prozessregeln. Wir interpretieren dabei Verschachtelung als die durchschnittliche Anzahl von Regeln, in deren Anwendungsbereich ein Ereignis des Prozesses liegt. Die Aktionen der Ereignisse e_{1a}, e_{1b}, e_{1c} sind beispielsweise im Anwendungsbereich von drei Regeln enthalten, das Ereignis e_{1d} wird sogar von vier Regeln berücksichtigt.

Der zweite Prozess in Abbildung 4.17 wird dagegen auf Basis von Regeln definiert, die stärker modularisiert sind und damit eine deutlich geringere Verschachtelung aufweisen. Jedes Ereignis in diesem Prozess liegt maximal im Anwendungsbereich von zwei Prozessregeln.

Der Unterschied im Aufbau der beiden Prozesse liegt also darin begründet, dass die Regeln des ersten Prozesses zur Ableitung neuer Aktionen auf einen größeren Kontext zurückgreifen. Ein größerer Kontext für eine Regelanwendung kann die Beschreibung der Regel vereinfachen, da Aussagen zu neuen Aktionen auf Grundlage einer umfassenderen Kenntnis von bereits durchgeführten Aktionen getroffen werden können. Entsprechend erfordert die Definition von Regeln mit einem stark abgegrenzten Kontext dahingehend eine größere Beschreibungsdisziplin, dass alle notwendigen Kontextinformationen für die Regelanwendung in den wenigen für die Regel sichtbaren Aktionen zum Ausdruck kommen müssen.

Auf der anderen Seite verkompliziert eine hohe Schachtelungstiefe Anpassungen an den Regeln erheblich, da Änderungen an einer Aktion einen Einfluss auf mehrere andere Regeln haben.

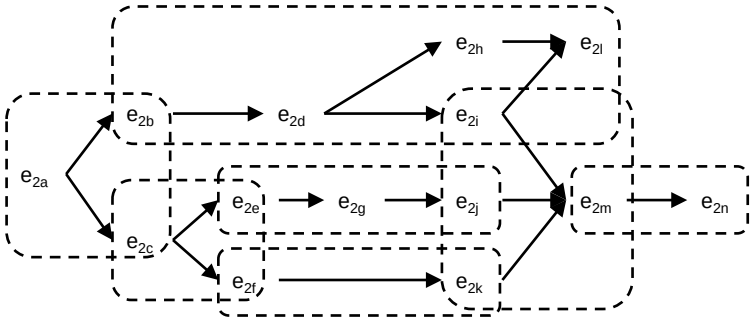

Abbildung 4.17: Schematische Darstellung eines Prozesses mit einer modularisierten Regelarchitektur (die gestrichelten Abgrenzungen entsprechen den Anwendungsbereichen der eingesetzten Prozessregeln)

Dies birgt Konsistenzrisiken und vermindert die Flexibilität einer Prozessdefinition. Insofern ist eine Modularisierung des Prozessaufbaus grundsätzlich wünschenswert. Die Herausforderung in einer modularen Prozessarchitektur liegt darin, die Kontextübergabe zwischen Prozessteilen, die durch unterschiedliche Regeln beschrieben werden, in einem kleinen Anwendungsbereich zu realisieren. Dies entspricht den bereits in Kapitel 2 diskutierten Prinzipien einer starken Kohäsion innerhalb des Anwendungsbereichs einer Regel und einer losen Kopplung zwischen den Regeln.

4.5.2 Dienstorientierte Architektur

Mit einer dienstorientierten Architektur lässt sich ein modularer Aufbau eines Prozesses realisieren. Die Regelarchitektur wird dabei entlang der Beiträge, die die einzelnen am Prozess beteiligte Akteure zum Gesamtablauf erbringen, strukturiert. Dienstorientiert beschriebene Prozesse weisen zudem vorteilhafte Kompositionseigenschaften auf, die die Entwicklung einer dienstorientierten Schichtenarchitektur ermöglichen.

Wir werden uns in den folgenden Abschnitten im Detail damit befassen, wie dienstorientierte Architekturen im Kontext unseres Prozessmodells zu verstehen sind und wie die im Abschnitt 2.2 formulierten SOA-Ziele durch Umsetzung einer dienstorientierten Prozessarchitektur erfüllt werden können.

4.5.2.1 Dienstorientierter Prozessaufbau

In einer dienstorientierten Architektur wird der Aufbau eines Prozesses an den Akteuren ausgerichtet, die am Prozess beteiligt sind. Wir fordern, dass eine Aktion eines Akteurs nur von Aktionen abhängen darf, die entweder vom gleichen Akteur durchgeführt oder von einem anderen Akteur explizit als Dienst dem Akteur gegenüber erbracht wurden. Dies gewährleistet

eine abgegrenzte Beschreibung eines Akteursverhalten und eine Erfassung von Interaktionen zwischen Akteuren an explizit definierten Übergabepunkten in Form von Diensten.

Definition 39. Dienstorientierte Architektur

Wir bezeichnen die Architektur eines Prozesses als dienstorientiert, wenn er ausschließlich auf Basis von akteurorientierten Regeln beschrieben wird. Dies bedeutet, dass alle Aktionen eines beliebigen im Prozess vorkommenden Akteurs nur von seinen eigenen internen Aktionen oder eingehenden Schnittstellenaktionen, die ihm gegenüber als Dienst erbracht werden, kausal abhängen. □

Die dienstorientierte Ablaufbeschreibung vereinfacht den modularen Aufbau komplexer Prozesse erheblich, da die Beiträge der einzelnen Akteure im Bausteinprinzip zusammengesetzt werden können. Zudem ermöglicht die Dienstorientierung eine Abstraktion von internen Aktionen eines Akteurs, da das Verhalten anderer Akteure nur von Schnittstellenaktionen abhängig ist.

Wir bezeichnen eine dienstorientierte Prozessarchitektur als vollständig dienstorientierte Architektur, wenn sich ein Prozess ausschließlich anhand dienstorientierter Regeln beschreiben lässt. Es ist auch möglich, dass in Prozessen eine Dienstorientierung nur für bestimmte Akteure umgesetzt wird. In diesem Fall sprechen wir von partiellen Dienstarchitekturen. In den folgenden Überlegungen gehen wir jedoch von vollständigen dienstorientierten Architekturen aus.

Beispiel. Betrachten wir erneut die Buchungsanfrage von Herrn Müller bei Holiday Airways, die wir bereits zuvor ausführlich diskutiert haben. Seitens der Fluggesellschaft sind drei Systeme als Akteure an der Bearbeitung der Kundenanfrage beteiligt. In Abbildung 4.18 ist der Prozess kausal dargestellt und der Anwendungsbereich für die Regeln, die das Verhalten der drei Akteure beschreiben, gestrichelt gekennzeichnet. Aus der Anschauung wird die Ausrichtung des Prozesses an den Akteuren unmittelbar deutlich.

Der Prozess ist beschrieben anhand der uns bereits bekannten Prozessregel *ha*, die sich wie folgt zusammensetzt:

$$ha = (ha_1 \parallel (ha_2; ha_3)); ha_4$$

Die Regeln, aus denen *ha* komponiert wird, sind akteurorientiert definiert. So beschreibt ha_1 das Verhalten des Buchungssystems *sb*, die Komposition $ha_{23} = ha_2; ha_3$ beschreibt das Verhalten des Systems zur Preisberechnung *sp* und ha_4 beschreibt das Verhalten des Systems zur Angebotserstellung *so*. Für ha_1, ha_{23}, ha_4 gilt

- $Scope(ha_1) \subseteq ActionsWith(sb)$ und $Delta(ha_1) \subseteq ActionsBy(sb)$
- $Scope(ha_{23}) \subseteq ActionsWith(sp)$ und $Delta(ha_{23}) \subseteq ActionsBy(sp)$
- $Scope(ha_4) \subseteq ActionsWith(so)$ und $Delta(ha_4) \subseteq ActionsBy(so)$

Jeder der drei Akteure bei Holiday Airways beobachtet im Verlauf der Anfragebearbeitung ausschließlich die eigenen Aktionen und Schnittstellenaktionen anderer Akteure, die diese dem Akteur gegenüber als Dienst erbringen. Entsprechend überlappen die Anwendungsbereiche der Regeln nur für die zwei Schnittstellenaktionen $Flight$ und $AmtDue$. Dies sind die Dienstwirkungen des Buchungssystems und des Systems zur Preisberechnung, die gegenüber dem System zur Angebotserstellung erbracht werden. □

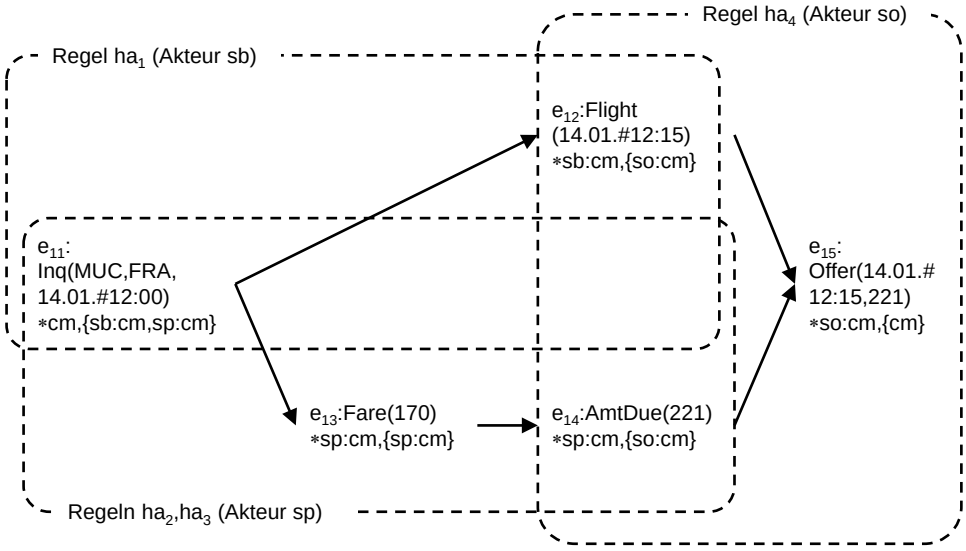

Abbildung 4.18: Kausaler Prozess einer Buchungsanfrage (die gestrichelten Abgrenzungen entsprechen den Anwendungsbereichen der eingesetzten Prozessregeln)

Wenngleich das Beispiel der Anfragebearbeitung eine verhältnismäßig einfache Prozessstruktur aufweist, lässt sich der durch die Dienstorientierung erreichte modulare Prozessaufbau bereits hier deutlich erkennen. Der vorgestellte Ansatz lässt sich problemlos auf komplexere Prozesse ausweiten und entwickelt dabei sein volles Potenzial dahingehend, dass Abhängigkeiten zwischen den Prozessteilen begrenzt werden. Die akteurbezogene Ausrichtung von Regeln in Dienstarchitekturen begünstigt typischerweise das Prinzip der losen Kopplung zwischen den Prozessteilen, da arbeitsteilige Vorgänge zumeist auf möglichst geringe Übergaben zwischen den Akteuren ausgelegt sind, um damit verbundene Transaktionskosten in betriebswirtschaftlicher Hinsicht zu reduzieren.

4.5.2.2 Komposition von Diensten

Die Komposition von Diensten ist ein wichtiges Prinzip in der Umsetzung einer dienstorientierten Architektur. In einer SOA wird die Beschreibung eines Prozesses zusammengesetzt aus den Diensten, die die einzelnen am Prozess beteiligten Akteure im Rahmen der Prozessdurchführung erbringen. Die Komposition einer Prozessregellogik aus feingranularen Diensten erhöht einerseits die Flexibilität im Hinblick auf Prozessanpassungen, da im Idealfall die bereits definierten Dienste nur den veränderten Anforderungen entsprechend neu zusammengestellt werden müssen. Andererseits können Dienste wiederverwendet werden, indem der gleiche Dienst an unterschiedlichen Stellen in eine Prozessregel durch Komposition eingebunden wird.

Da Dienste eine besondere Ausprägung einer Regel darstellen (vgl. Abschnitt 4.4.3), kann die Komposition von Diensten auf Basis der in unserem Modell eingeführten Regeloperatoren er-

folgen. Mit der Anwendung der Operatoren wird eine sequentielle, parallele beziehungsweise alternative Ausführungsreihenfolge der Dienste festgelegt und somit der Bezug der Dienste zueinander bestimmt. Prinzipiell gelten also unser bisherigen Überlegungen zur Komposition von Prozessregeln in gleicher Weise für die Dienstkomposition. Zusätzlich verfügen Dienste über die vorteilhafte Eigenschaft, dass sie auch in der Komposition nach Ausblendung interner Aktionen wieder als Dienst angesehen werden können und somit eine Dienstorientierung in der Komposition erhalten bleibt.

Jede Komposition zweier akteurorientierter Regeln für zwei Akteursgruppen $Act_1, Act_2 \subseteq Actors$ resultiert wiederum in einer akteurorientierten Regel für die Vereinigung dieser Akteursgruppen $Act_1 \cup Act_2$. Dies folgt, da per Definition alle Aktionen der beiden Regeln ausschließlich von Aktionen abhängen, die entweder für einen der Akteure als Schnittstellenaktion beobachtbar und damit auch für die Vereinigung der Akteure sichtbar bleiben oder von einem der Akteure selbst durchgeführt werden. Diese Argumentation lässt sich für alle drei Formen der Komposition anwenden.

Weiterhin gilt für eine sequentielle Komposition zweier akteurorientierter Regeln r_1, r_2, die das Verhalten zweier Akteursgruppen $Act_1, Act_2 \in Actors$ beschreiben, mit $Acts = Act_1 \cup Act_2$:

$$[r_1; r_2]_{\Uparrow Acts} = \left[[r_1]_{\Uparrow Act_1} ; [r_2]_{\Uparrow Act_2} \right]_{\Uparrow Acts}$$

Betrachten wir das Verhalten zweier sequentiell operierender Akteursgruppen aus einer Schnittstellenperspektive als Dienst, so kann dieses auch aus der sequentiellen Komposition der beiden einzelnen Dienste, die die Akteursgruppen anbieten, hergeleitet werden. Dies gilt, da das Verhalten der zweiten Akteursgruppe nur durch Schnittstellenaktionen beeinflusst werden kann und diese im Dienst erfasst sind. Die gleiche Aussage kann auch für die parallele und alternative Komposition getroffen werden, ist in diesem Fall allerdings trivial, da keine Beeinflussung zwischen den Akteursgruppen stattfindet.

Diese Eigenschaft einer dienstorientierten Prozessarchitektur kann die Prozessbeschreibung erheblich vereinfachen. Wenn ein Dienst einer größeren Gruppe von Akteuren, beispielsweise einer Organisationseinheit in einem Unternehmen, in mehrere Schritten aus den Diensten von Teilgruppen komponiert wird, kann in jedem Kompositionsschritt die Betrachtung auf Schnittstellenaktionen fokussiert werden. Auf diese Weise können stufenweise interne Aktionen von Akteuren und ganzen Akteursgruppen in der Betrachtung eliminiert werden. Somit kann die Beschreibung komplexer übergreifender Dienste auf die Beschreibung einfacher Dienste in Komposition zurückgeführt werden.

Beispiel. Wir kommen zurück auf unser Beispiel der Flugbuchung bei Holiday Airways gemäß der Prozessregel

$$ha = (ha_1 \parallel (ha_2; ha_3)); ha_4$$

Wir haben bereits im letzten Abschnitt festgestellt, dass die Regeln ha_1, ha_{23}, ha_4 akteurorientiert formuliert sind und das Verhalten der drei Rechnersysteme sb, sp, so bei Holiday Airways beschreiben. Es lässt sich nun einfach erkennen, dass auch die zusammengesetzte Regel ha akteurorientiert ist und das Gesamtverhalten der Akteursgruppe $\{sb, sp, so\}$ beschreibt. Die Regeln

ha_1 und ha_{23} weisen eine kausale Abhängigkeit zur Kundenanfrage *Inq* auf. Diese ist für die Akteursgruppe $\{sb, sp, so\}$ beobachtbar und damit eine eingehende Schnittstellenaktion. Das in der Regel ha_4 beschriebene Verhalten hängt wiederum von den Aktionen *Flight* und *AmtDue* ab. Diese werden von den Systemen *sb* und *sp* durchgeführt und stellen somit in der Betrachtung der Gruppe $\{sb, sp, so\}$ interne Aktionen dar.

Beschränken wir die Betrachtung von *ha* auf Schnittstellenaktionen, so erhalten wir mit $[ha]_{\Uparrow HA}$ und $HA = \{sb, sp, so\}$ den Dienst, den Holiday Airways gegenüber seinen Kunden anbietet. Dieser Dienst kann als Komposition der Dienste hergeleitet werden, die die drei Teilsysteme anbieten:

$$[ha]_{\Uparrow HA} = \left[\left([ha_1]_{\Uparrow \{sb\}} \parallel [ha_2; ha_3]_{\Uparrow \{sp\}} \right); [ha_4]_{\Uparrow \{so\}} \right]_{\Uparrow HA}$$

Für die Beschreibung des Dienstes von Holiday Airways *ha* gegenüber seinen Kunden sind die internen Aktionen der beteiligten Systeme nicht relevant. Stattdessen kann dieser Dienst aus den Teildiensten, den die drei beteiligten Serversysteme anbieten, gemäß der Regel zusammengesetzt werden. Dies wird möglich durch den dienstorientierten Aufbau des Bearbeitungsprozesses und die damit einhergehende Fokussierung auf Schnittstellenaktionen zur Übergabe von Kontextinformationen zwischen den Serversystemen. □

4.5.3 Dienstorientiertes Architekturmodell

Wir haben im letzten Abschnitt erörtert, wie eine Prozessbeschreibung modular aus der Komposition von Diensten entwickelt werden kann. Bei umfassenden und entsprechend komplexen Prozessen ist es möglich, dass diese auf Basis einer Vielzahl von Diensten beschrieben wird. Der Umgang mit einer großen Anzahl von Diensten erfordert einen systematischen Kompositionsansatz, damit die Verständlichkeit des Prozessaufbaus sowie die Handhabbarkeit im Hinblick auf Prozessweiterentwicklungen gewährleistet ist. Wir werden uns daher in diesem Abschnitt mit dem Aufbau von Schichtenmodellen zur strukturierten Beschreibung von Dienstarchitekturen befassen.

4.5.3.1 Aufbau des Architekturmodells

Im Sinne der Anschaulichkeit führen wir die Diskussion weiterhin am Beispiel unseres bekannten Buchungsprozesses in leicht erweiterter Form. In Abbildung 4.19 stellen wir die Prozessbeschreibung in einem dienstorientierten Schichtenmodell dar. Die Grundidee des Modells besteht daran, basierend auf einfachen Basisdiensten mit stark abgegrenztem funktionalen Umfang sukzessive den Vorgang der Buchungsbearbeitung zusammenzustellen. Dies geschieht in mehreren Schritten, die in den Ebenen des Modells dargestellt sind. Jede Ebene greift dabei auf zuvor definierte Dienste zurück und bindet diese in die Definition komplexerer Dienste mit ein.

Beginnen wir die Betrachtung auf der untersten Ebene. Hier sind zwei einfache Berechnungsvorgänge *GetDist* und *GetDestSurchg* definiert. Wir gehen davon aus, dass diese durch Prozessregeln beschrieben sind. *GetDist* ermittelt für eine gegebene Kundenanfrage

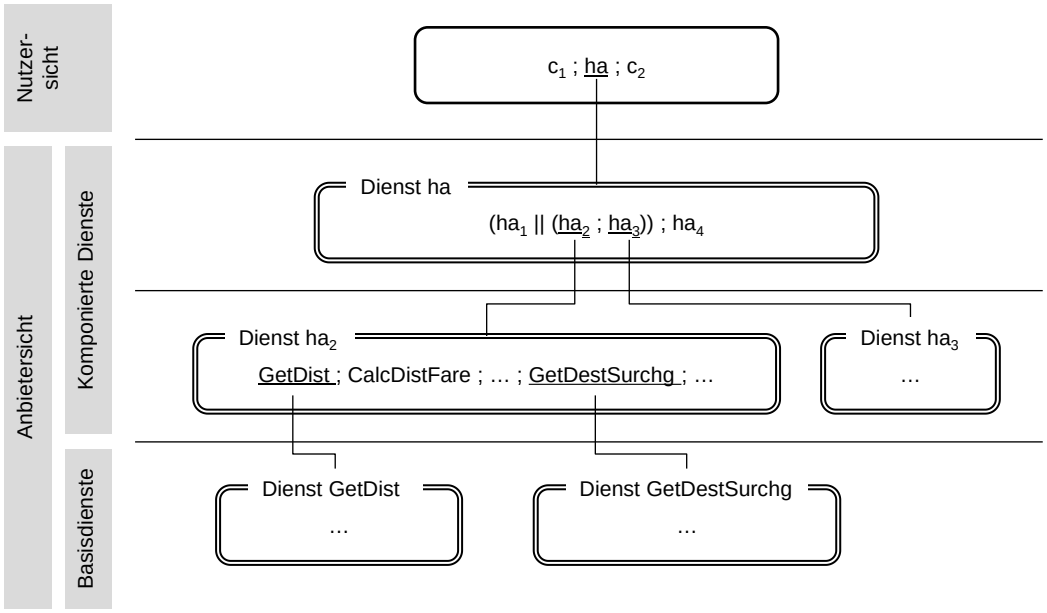

Abbildung 4.19: Dienstorientierte Prozessarchitektur, dargestellt in Schichtenmodell

$Inq(\langle Orig\rangle, \langle Dest\rangle, \langle Date\rangle)$ die Strecke zwischen Abflug- und Zielflughafen und gibt diese in Aktion $Dist(\langle Value\rangle)$ aus. *GetDestSurchg* ermittelt für eine gegebene Anfrage den Zuschlag, der für den Zielflughafen zu berechnen ist, und gibt diesen als $DestSurchg(\langle Value\rangle)$ aus. Wir fokussieren rein auf die Dienste dieser Berechnungsvorgänge, also die Relation zwischen der Anfrage *Inq* und den entsprechenden Resultaten *Dist* beziehungsweise *DestSurchg*, und abstrahieren von internen Aktionen, die zur Berechnung erforderlich sind.

Diese beiden Dienste können nun eingebunden werden in die Definition der Grundpreisberechnung ha_2 auf der zweiten Ebene. Wir verwenden an dieser Stelle abweichend von unserem Ansatz in Abschnitt 4.3.2.2, in dem wir die Grundpreisberechnung auf Basis eines formalen Automatenmodells spezifiziert haben, eine Regelkomposition zur Ablaufbeschreibung für die Preisberechnung. Die Komposition ist sequentiell aufgebaut und kann als Algorithmus betrachtet werden. Wir beschränken uns dabei auf die exemplarische Betrachtung zweier Ausschnitte der Grundpreisberechnung:

$$ha_2 = [GetDist]_{\Uparrow\{s_1\}}; CalcDistSurchg; \ldots; [GetDestSurchg]_{\Uparrow\{s_2\}}; \ldots$$

Wir gehen davon aus, dass der Dienst *GetDist* von einem System s_1 und der Dienst *GetDestSurchg* von einem System s_2 erbracht wird. In der graphischen Darstellung kennzeichnen wir Prozessteile, die als Dienst eingebunden werden, durch einen Unterstrich und verbinden diese mit der Definition des Dienstes. Im Ablauf zur Preisberechnung wird zunächst anhand des Dienstes von s_1 die Strecke *Dist* ermittelt. Im Anschluss wird durch die Prozessregel *CalcDistSurchg* die Strecke *Dist* mit einer Kilometerpauschale multipliziert und so der Entfernungspreis ermittelt. Zu einem späteren Zeitpunkt im Prozess wird durch den Dienst von s_2 der

Zuschlag für den Zielort ermittelt. Wir nehmen an, dass die übrigen Prozessteile von ha_2 sowie die Berechnung der Steuern und Gebühren in ha_3 in analoger Weise beschrieben sind.

In der dritten Ebene können nun wiederum die Dienste, die aus den Preisberechnungen der zweiten Ebene resultieren, weiter zusammengesetzt werden. Wir verwenden dazu die bereits bekannte Regelkomposition zur Bearbeitung einer gesamten Kundenanfrage. In diese binden wir die Dienste zur Grundpreisberechnung $[ha_2]_{\Cap SP_1}$ und zur Berechnung der Steuern $[ha_3]_{\Cap SP_2}$, die auf der zweiten Ebene erbracht werden, mit ein. SP_1 und SP_2 stehen dabei für die Rechnersysteme, die die jeweiligen Dienste anbieten.

Schließlich lässt sich der gesamte Bearbeitungsvorgang wiederum als Dienst $[ha]_{\Cap HA}$ betrachten, der nun in der vierten Ebene in den Prozess des Kunden eingebunden werden kann. Wir stellen den Kundenprozess lediglich illustrativ dar wie folgt:

$$c_1; [ha]_{\Cap HA}; c_2$$

Die Regel c_1 beschreibt dabei die Vorarbeiten des Kunden, bevor er die Anfrage stellt, beispielsweise die Auswahl eines Reisezeitraums und -ziels. Der Kunde nutzt dann den Dienst der Fluggesellschaft, um für die gewünschten Reisepräferenzen ein Angebot einzuholen. Die Regel c_2 steht schließlich für die Nachbereitung seitens des Kunden.

4.5.3.2 Eigenschaften des Architekturmodells

Im soeben diskutierten Beispiel haben wir anhand eines Schichtenmodells in einem mehrstufigem Vorgehen Dienste in systematischer Weise zusammengesetzt. Dabei haben wir ausgehend von einfachen Diensten mit beschränktem Funktionsumfang in der untersten Schicht komplexere und umfassende Dienste in den höheren Schichten gebildet. Die Dienste stellen dabei jeweils die Funktionalität für die Beschreibung der nächsthöheren Schicht zur Verfügung. [BS99] Die Dienste in der untersten Ebene bezeichnen wir auch als Basisdienste. In den weiteren Ebenen sprechen wir von zusammengesetzten oder auch komponierten Diensten.

Die Basisdienste stellen typischerweise Grundfunktionalitäten zur Verfügung, die in den oberen Schichten zunehmend angereichert werden. Prinzipiell kann jeder Dienst nicht nur einmalig in einer höheren Schicht eingebunden werden, sondern an mehreren Stellen in Kompositionen zum Einsatz kommen. Diese Form der Wiederverwendung ist ein zentrales Prinzip dienstorientierter Architekturen. Der systematische Aufbau von Diensten aus einfachen Basisdiensten erweist sich dabei als vorteilhaft. Die Dienste der unteren Schichten können auf Grund des abgrenzten Funktionsumfangs weitestgehend unabhängig vom Anwendungskontext definiert werden. Solche Dienste werde auch als agnostisch bezeichnet. [Erl08] In unserem Beispiel könnte der Dienst zur Streckenberechnung $[GetDist]_{\Cap\{s_1\}}$ auch in anderen Prozessteilen eingesetzt werden, in denen eine solche Berechnung erforderlich ist. In den höheren Schichten dagegen findet eine zunehmende Spezialisierung der Dienste auf den Anwendungskontext statt, die eine Wiederverwendung erschwert.

In allen Schichten verwenden wir Prozessregeln zur präzisen Beschreibung von Prozessabläufen. Die Dienste stellen die Verknüpfung zwischen den Ebenen dar, indem sie die Regel der

darunterliegenden Schicht aus einer Schnittstellenperspektive in die Regeln einbinden. Auf diese Weise erreichen wir eine zunehmende Abstraktion von Details der Prozessdurchführung. Der Bezug zwischen den Ebenen ist jedoch durch die formale Definition des Dienstes gewährleistet. Wir können die Ebenen eines Architekturmodells dabei als unterschiedliche Sichten auf einen Prozess betrachten. Die Übersetzung zwischen den Sichten erfolgt anhand der Projektionsfunktion wie in Abschnitt 4.4.3 definiert. Dabei werden jeweils die internen Aktionen ausgeblendet. Die Projektion als homomorphe Abbildung gewährleistet eine konsistente Betrachtungsweise über die Ebenen hinweg.

Die Prozessregeln zur Beschreibung der Dienste müssen nicht zwingend nur aus Diensten einer zugrundeliegenden Schicht zusammengesetzt sein. Es können wie im Beispiel gezeigt auch ergänzend Regeln in der Komposition eines Dienstes zum Einsatz kommen, die nur in diesem Kontext eingesetzt werden. So haben wir im Rahmen der Preisberechnung nach der dienstbasierten Ermittlung der Strecke zwischen Abflug- und Zielflughafen den Entfernungspreis durch die Regel *CalcDistFare* ermittelt. Diese haben wir nicht als Basisdienst definiert, sondern verwenden sie ausschließlich in der Beschreibung des Dienstes ha_2. Wenn jeder Dienst ausschließlich aus anderen Diensten zusammengesetzt wird, muss eine große Anzahl von Diensten definiert werden. Dies führt zu einer Dienstarchitektur mit einer Vielzahl von Abhängigkeiten. Es kann sich daher durchaus als zweckmäßig erweisen, nur diejenigen Prozessteile in Diensten zu beschreiben, bei denen eine Wiederverwendung zu erwarten oder eine flexibler Aufbau im Hinblick auf künftige Anpassungen wünschenswert ist.

In unserem Beispiel stellen die untersten drei Ebenen die Perspektive des Anbieters Holiday Airways und die vierte Ebene die Kundenperspektive dar. Dies zeigt, dass Diensthierarchien sich über Organisationen hinweg erstrecken können und Dienstleistungen einer Organisation wiederum als Basis für die Dienste einer anderen Organisation herangezogen werden können.

Aus der systematischen dienstorientierten Beschreibung von Prozessen in Schichtenmodellen ergeben sich hierarchische Dienststrukturen. Bei der Definition neuer Prozesse kann auf die bestehende Dienste zurückgegriffen werden und diese können durch Komposition zu neuen Dienstangeboten erweitert werden. Die Gesamtheit aller Dienste einer Organisation wird auch als Dienstlandschaft bezeichnet. Dies ist verbunden mit der Idee einer systematischen und strukturierten architekturellen Betrachtung aller in einer Organisation verfügbaren Dienstleistungen. Es existieren verschiedene Ansätze zur Klassifizierung von Diensten in Dienstlandschaften (vgl. dazu [Erl08, FR07]).

4.5.3.3 Entwicklung eines Architekturmodells

Wir haben bislang Dienstarchitekturen – dem Fokus unserer Arbeit entsprechend – aus einer konzeptuellen Sicht betrachtet. Für den Einsatz von SOA sind neben einem präzisen konzeptuellen Verständnis auch methodische Aspekte von Interesse. Dabei steht das Vorgehen zur Entwicklung eines dienstorientierten Architekturmodells im Mittelpunkt. Wir werden im Folgenden einen kurzen Ausblick geben zu den Fragestellungen, die in methodischer Hinsicht zu adressieren sind, und entlang des Überblicks illustrativ beleuchten, wie unser formales Konzeptmodell eine methodische Diskussion im Bereich SOA unterstützen kann.

Entwicklungsansatz Es gibt prinzipiell zwei Ansätze, ein dienstorientiertes Architekturmodell, so wie wir es in den letzten Abschnitten diskutiert haben, aufzubauen. Im top-down Vorgehen wird das Architekturmodell ausgehend von der obersten Ebene entwickelt. Der bottom-up Ansatz beginnt dagegen auf der untersten Ebene mit den Basisdiensten.

In der top-down Entwicklung erfolgt eine iterative Analyse einer umfassenden unternehmerischen Aufgabe einer Organisation in einzelne Dienste, die von verschiedenen Akteuren innerhalb der Organisation erbracht werden (vgl. [QDS04, KA07, CK07, KD07] für eine ausführlichere Diskussion). In unserem Beispiel würden wir dazu ausgehen von der Dienstleistung $[ha]_{\Uparrow HA}$, die Holiday Airways in der obersten Ebene gegenüber seinen Kunden erbringt, und diese dann Ebene für Ebene zerlegen in Teilaufgaben bis hin zu den Basisdiensten. Die Zerlegung eines Dienstes in Teilaufgaben wird auch als Dekomposition bezeichnet. Im Zuge der Dekomposition müssen Annahmen bezüglich der Struktur der Organisation getroffen werden, um die gegebene Aufgabe in Teilbereiche zu gliedern und den entsprechenden Akteuren zuzuweisen.

In der bottom-up Entwicklung werden zunächst Dienste einzelner Akteure festgelegt und anschließend per Komposition iterativ komplexere Ablaufstrukturen ganzer Akteursgruppen beschrieben (vgl. [KSR04, ZLY05, Off08] für eine ausführlichere Diskussion). In unserem Beispiel würden wir dazu von Basisdiensten wie der Streckenberechnung $[GetDist]_{\Uparrow\{s_1\}}$ in der untersten Schicht ausgehen, und diese dann sukzessive zusammensetzen. Wie in den letzten Abschnitten bereits diskutiert wird dabei durch Ausblendung interner Aktionen in jeder Kompositionsstufe von Details der Leistungserbringung zunehmend abstrahiert.

Naturgemäß besteht ein enger Zusammenhang zwischen den beiden Entwicklungsweisen einer SOA. Der top-down Ansatz ist häufig geprägt durch die Idee der Entwicklung einer Ziel- oder Referenzarchitektur (Blueprint), die unabhängig von bestehenden technischen Lösungen formuliert wird. Die bottom-up Methode richtet sich dagegen häufig an bestehenden Softwarelösungen und Fähigkeiten der beteiligten Akteure aus.

Die konzeptuell formale Perspektive unseres Modells hilft dabei, sowohl bei der Dekomposition (top-down Vorgehen) als auch bei der Komposition (bottom-up Vorgehen) einen präzisen Bezug zwischen den Ebenen herzustellen. Durch Einsatz der Kompositionsoperatoren kann dabei in Regeln exakt formuliert werden, wie die Dienste der zugrundeliegenden Schicht zu einer umfassenderen Dienstleistung zusammengesetzt werden.

Verfeinerung Bei der top-down Entwicklung einer dienstorientierten Architektur ist es zumeist nicht praktikabel, auf der obersten Ebene die Dienstleistung einer Organisation bereits in voller Präzision zu beschreiben. Auf Grund der Komplexität einer solchen Dienstleistung erweist es sich häufig als zweckmäßig, diese zunächst nur approximativ zu erfassen und im Zuge der Dekomposition die Teildienste zunehmend zu präzisieren, die auf Grund ihres kleineren funktionalen Umfangs einfacher zu beschreiben sind.

In unserem Beispiel wäre es denkbar, zunächst den Dienst von Holiday Airways gegenüber seinen Kunden in Form einer schwach formulierten Bedingung zu fassen, dass die Fluggesellschaft auf jede Anfrage Inq mit einem Angebot $Offer$ zu reagieren hat, ohne dabei Aussagen

zu den Angaben zu treffen, die im Angebot enthalten sind. Im Zuge der Dekomposition kann dieser Dienst dann, wie im Beispiel diskutiert, in die einzelnen Teildienste der Verfügbarkeitsermittlung, Preisberechnung und Angebotserstellung zerlegt werden. Für diese Teilabläufe kann nun die Beschreibung präzisiert werden, indem Anforderungen zur Beschaffenheit des Ergebnisses in Abhängigkeit der Eingaben gestellt werden. Zuletzt ist es erforderlich, dass die in den Teildiensten gewonnenen Erkenntnisse auch präzise wieder in die übergreifenden Dienste der höheren Schichten propagiert werden können.

Unser Modell kann eingesetzt werden, um solche Verfeinerungsbeziehungen formal zu erfassen. Im Zuge der Verfeinerung wird eine regelbasierte Prozessspezifikation als Menge zulässiger Prozessabläufe zunehmend eingeschränkt und damit präzisiert. Dies kann exakt dargestellt werden anhand der Verfeinerungsrelation, die wir in Abschnitt 4.2.4.1 eingeführt haben. Zudem können Verfeinerungen, die auf einer bestimmten Ebene vorgenommen wurden, durch den formalen Bezug zwischen den Schichten direkt auch in andere Ebenen übersetzt werden.

Granularität und Schnittmuster Im Zuge einer Architekturentwicklung müssen Designentscheidungen hinsichtlich der Granularität und des Schnitts von Diensten getroffen werden. Im Hinblick auf die Granularität gilt, dass prinzipiell jeder Dienst wiederum in Teildienste zerlegt werden kann und insofern eine Entscheidung bezüglich der anzustrebenden Detaillierungstiefe zu treffen ist. Zudem besteht ein großer Gestaltungsspielraum bezüglich der Unterteilung eines Dienstes in Teilabläufe. Typischerweise kann ein Dienst in vielfältiger Weise in Teildienste zerlegt werden. Wir bezeichnen eine solche Zerlegung auch als Schnitt und die daraus resultierende Struktur der Teildienste als Schnittmuster.

Für eine gegebene unternehmerische Aufgabe existiert folglich keine eindeutige dienstorientierte Architektur, die diese abbildet. Vielmehr können durch die Variationsmöglichkeiten hinsichtlich Granularität und Dienstschnitt eine Vielzahl von Architekturen konstruiert werden. Die hier zu treffenden Designentscheidungen erfordern eine Abwägung zwischen unterschiedlichen Zielen. Im Sinne einer hohen Flexibilität ist beispielsweise eine möglichst feingranulare Dienststruktur wünschenswert. Umgekehrt führt eine zu hohe Anzahl von Diensten zu einem unverhältnismäßig hohen Beschreibungs- und Wartungsaufwand. Die Designentscheidungen werden häufig geleitet durch die in Abschnitt 2.2.2 diskutierten Gestaltungsprinzipien einer Dienstarchitektur, die unter anderem eine lose Kopplung und eine starke Kohäsion der Dienste vorsehen.

Unser Modell kann solche Designentscheidungen unterstützen, indem aus der formalen Definition Metriken abgeleitet werden, auf deren Basis die nötigen Abwägungen getroffen werden (vgl. [SSP09]). So kann die Kopplung zwischen Diensten und die Kohäsion innerhalb von Diensten beispielshalber an der Anzahl von Kausalitäten erfasst werden, die zwischen den entsprechenden Ereignissen bestehen.

Technische Kopplung Zuletzt muss neben der fachlichen Modellierung einer SOA auch die Übersetzbarkeit in eine technische Sicht gewährleistet sein. Wie wir bereits ausführlich diskutiert haben, ergeben sich die Vorteile einer dienstorientierten Architektur aus einer verbesserten Ausrichtung von fachlicher und technischer Architektur, die auch in dynamischen Umgebungen mit sich häufig ändernden prozessualen Anforderungen aufrecht erhalten werden kann. Die

Dienste dienen dabei als Bezugspunkte zwischen der fachlichen und technischen Sicht. Wann immer Dienste in der fachlichen Architektur neu zusammengestellt werden, kann dies durch analoge Dienstumbildungen in der technischen Architektur nachgebildet werden und so die Kongruenz zwischen Fachlichkeit und technischer Realisierung erhalten werden.

Dies setzt voraus, dass fachlich definierte Dienste präzise auf technische Dienste abgebildet werden können, damit auch bei Anpassungen die Semantik in der fachlichen und technischen Domäne konsistent gehalten werden kann. Unser Modell ermöglicht durch seine mathematische Fundierung eine exakte Übersetzung fachlicher Dienste in ihre technischen Pendants. Die Übersetzung wird zudem vereinfacht, wenn Dienste, wie in Abschnitt 4.3.2.2 diskutiert, bereits in operationeller Weise definiert werden und somit ein formales Automatenmodell als Grundlage für die Implementierung herangezogen werden kann.

4.6 Zusammenfassung des Modells

In diesem Kapitel haben wir ein formales Modell zur Darstellung geschäftlicher Abläufe entwickelt und daraus Konzepte zur Beschreibung dienstorientierte Architekturen abgeleitet. Das Modell realisiert eine präzise Abgrenzung zwischen den Konzepten, indem wir uns zunächst ausführlich mit der formalen Definition geschäftlicher Vorgänge in voller Allgemeinheit befassen, die wir als Prozesse bezeichnen, und alle weiterführenden spezialisierten Konzepte einer SOA aus dem Prozessbegriff ableiten. Wir greifen dabei die in Kapitel 3 vorgeschlagene Struktur auf und führen das Modell systematisch in einem dreistufigen Vorgehen ein.

In der ersten Stufe diskutieren wir in einem abstrakten Grundmodell die formalen Grundlagen für die Erfassung von Prozessen. Dazu betrachten wir Prozesse zunächst unabhängig von einer bestimmten Repräsentation und fordern lediglich eine partielle Ordnungsbeziehung zwischen den Prozessen in Form der Teilprozessrelation. Wir betrachten dabei einen Teilprozess als einen beliebigen Ausschnitt eines Ablaufs. Auf Basis dieser abstrakten Prozesssicht lässt sich bereits die semantische Definition eines Prozesses, die wir durch Prozessmengen erfassen, untersuchen. Anhand von Prozessregeln, die aus bereits durchgeführten Prozessen weitere durchzuführende Aktionen ableiten und damit Prozesse entlang der Teilprozessordnung entwickeln, können solche Prozessmengen systematisch beschrieben werden. Dazu wenden wir Regeln auf eine Ausgangsmenge von Prozessen an, die wir als Annahme voraussetzen, und bilden eine transitive Hülle aller Prozesse, die sich aus der Ausgangsmenge regelbasiert entwickeln lassen.

In der zweiten Stufe des Modells führen wir eine konkrete Prozessrepräsentation in drei Varianten ein. Wir basieren unsere Prozessbetrachtung auf Ereignismengen, wobei ein Ereignis die einmalige Durchführung einer Aktion repräsentiert. Eine Aktion wird dabei durch einen Akteur ausgeführt und kann von anderen Akteuren beobachtet werden. Die drei Prozessvarianten unterscheiden sich im zugrundeliegenden Zeitmodell. Für ungeordnete Prozesse treffen wir keine Aussage bezüglich des Eintrittszeitpunkts der Ereignisse. Bei kausalen Prozessen wenden wir eine relative Zeitordnung an, indem wir Ereignisse bezüglich ihres Auftrittszeitpunkts zueinander in Relation stellen. Weiterhin betrachten wir temporale Prozesse mit einem quantitativen Zeitmodell, in dem jedem Ereignis ein bestimmter Eintrittszeitpunkt zugeordnet ist. Die in der

ersten Stufe des Modells hergeleiteten semantischen Grundlagen lassen sich dabei direkt auf die ereignisorientierte Prozessrepräsentation anwenden. Mit den Konzepten der zweite Stufe lassen sich geschäftlicher Abläufe in voller Allgemeinheit präzise modellieren.

In der dritten Stufe des Modells führen wir schließlich weiterführende Konzepte zur differenzierten Betrachtung geschäftlicher Abläufe ein und fokussieren uns dabei insbesondere auf Konzepte zur Beschreibung dienstorientierter Architekturen. Wir nehmen eine akteurorientierte Instanzierung von Prozessen vor, um einzelne Instanzen als Transaktionen abgrenzen zu können. Wir differenzieren dabei zwischen Transaktionen, die sich gegenseitig beeinflussen und unabhängigen Transaktionen. Auf Basis von Diensten erfassen wir das Verhalten von Akteuren beziehungsweise von Gruppen von Akteuren. Ein Dienst stellt dabei eine Prozessregel dar, die das Schnittstellenverhalten einer Akteursgruppe erfasst und damit eine präzise Beschreibung der Beiträge von Akteuren zu Prozessen ermöglicht. Mit Anwendungsfällen nehmen wir zuletzt noch eine Differenzierung von Prozessen in einer Spezifikation nach Erfüllung einer geschäftlichen Zielvorgabe vor.

Zum Abschluss des Kapitels haben wir uns mit der Architektur von Prozessen befasst. Wir verstehen dabei die Aktionen eines Prozesses als Elemente der Architektur und bestehende Kausalitäten zwischen den Aktionen als die in der Architektur erfassten Beziehungen. Auf diese Weise lässt sich die Ablaufstruktur eines Prozesses darstellen. In dienstorientierten Architekturen werden Prozesse auf Basis von akteurorientierten Regeln und Diensten beschrieben und somit an Akteuren ausgerichtet. Dies vereinfacht die Prozesszusammensetzung, da Dienste auch in der Komposition nach Ausblendung von internen Aktionen wieder Diensteigenschaften aufweisen. Auf diese Weise können in einem mehrstufigen Vorgehen aus Basisdiensten sukzessive umfassende Dienste gebildet werden. Die Kompositionszusammenhänge stellen wir in einem dienstorientierten Schichtenmodell dar.

Übergreifend betrachten wir die in unserem Modell vorgestellte formale Notation stellt als eine mögliche mathematische Repräsentation von Prozessen, anhand der wir die präzise Abgrenzung und Einordnung der unterschiedlichen Konzepte zur Darstellung von betrieblichen Abläufen vornehmen konnten. Die gewonnenen Erkenntnisse bezüglich der unterschiedlichen Konzepte und deren Beziehungen lassen sich auch in andere Darstellungsformen übertragen und sind somit nicht an die Formalisierung dieses Kapitels gebunden.

5 Modellanwendung

In diesem Kapitel steht die Anwendung des im letzten Kapitel vorgestellten formalen Modells im Fokus. Dazu nehmen wir zunächst eine Einordnung mit anderen Ansätzen zur Modellierung geschäftlicher Abläufe vor. Wir werden dazu Referenzansätze mit einer vergleichbaren Intension auf Gemeinsamkeiten und Unterschiede hin untersuchen und daraus Anwendungsmöglichkeiten unseres Modells ableiten.

Im zweiten Teil des Kapitels werden wir uns mit der informellen Modellierung befassen. Eine formale mathematische Abbildung von Prozessen ist in der Praxis nicht immer realisierbar. Wir werden daher diskutieren, wie die gewonnenen Erkenntnisse auch in einer informellen Notation adäquat umgesetzt werden können.

5.1 Einordnung des Modells

Für die Modellierung betrieblicher Abläufe sind bereits eine Reihe unterschiedlicher Notationen etabliert. Wir werden im Folgenden eine repräsentative Auswahl dieser Notationen als Referenzansätze untersuchen. Die Referenzansätze weisen bezüglich ihres Formalisierungsgrades ein breites Spektrum auf von einer informell graphischen Darstellungsweise bis zu einer vollständig formal definierten Semantik. Zur systematischen Diskussion führen wir zunächst wesentliche Eigenschaften eines Modells zur Ablaufbeschreibung ein und betrachten anschließend die Ansätze im Kontext dieser Eigenschaften.

Im Anschluss werden wir unser Modell in Bezug zu den vorgestellten Ansätzen einordnen und anhand dessen diskutieren, in welchen Bereichen der Einsatz unseres Modells vorteilhaft ist und zusätzliche Erkenntnisse im Vergleich zu den besprochenen Referenzansätzen ermöglichen kann.

5.1.1 Kriterien

Wir geben zunächst einen Überblick über die Kriterien, anhand der wir die Einordnung und Diskussion der Referenzansätze vornehmen werden.

- *Prozessdarstellung*
  Wir differenzieren Referenzansätze zunächst nach der gewählten Repräsentation, anhand der Prozesse modelliert werden. Dabei unterscheiden wir im Wesentlichen vier Arten der Prozessdarstellung. Zustandsorientierte Modelle erfassen einen Prozess anhand der Folge der aufgetretenen Zustände, die sich aus den Handlungen ergeben. Im Kontrast zu den

übrigen drei Arten stehen bei zustandsorientierten Modellen die Aktionen eines Prozesses nicht im Fokus. Aktionsorientierte Prozesse modellieren Abläufe als sequentielle und parallele Verknüpfung von Aktionen. Nachrichtenorientierte Prozesse sind eine spezielle Ausprägung von aktionsorientierten Prozessen, bei denen alle Aktionen als Nachrichten betrachtet werden und somit der Fokus auf Interaktionen liegt. Prozessalgebren nehmen schließlich einen sehr formalen Standpunkt ein und beschreiben Prozesse operationell. Auch sie können als spezielle Ausprägung eines aktionsorientierten Prozesses angesehen werden, indem die einzelnen Anweisungen einer algebraischen Spezifikation als Aktionen aufgefasst werden.

- *Zeitmodell*
  Prozesse stellen Handlungsabläufe über den Verlauf der Zeit hinweg dar. Insofern ist in der Betrachtung von Referenzansätzen ebenfalls von Interesse, welches Zeitmodell diesen zugrundeliegt und wie Zeit folglich in den Prozessen erfasst wird. Wir unterscheiden dabei im Wesentlichen drei Ausprägungen. Prozessdarstellungen ohne Zeitmodell lassen nur eine eingeschränkte Prozessbeschreibung zu, da Abläufe auf Grund des fehlenden Zeitbegriffs nicht modelliert werden können. Prozesse mit qualitativen Zeitmodellen treffen relative Aussagen zu den Zeitpunkten der Aktionen beziehungsweise Zuständen, indem sie deren Abfolge festlegen. Quantitative Zeitmodelle ordnen den Aktionen eines Prozesses konkrete Zeitpunkte zu, zu denen diese ausgeführt werden.

- *Akteure*
  Wir haben Akteure als aktiv und passiv an den Prozessen beteiligte Personen und Systeme kennengelernt. Insofern untersuchen wir Referenzmodelle auch hinsichtlich der Darstellbarkeit von Akteuren und diskutieren jeweils, wie die Zuordnung der Handlungen zu den Akteuren erfolgt.

- *Instanzen*
  In Prozessmodellen, die eine Betrachtung von Akteuren unterstützen, ist prinzipiell eine Erfassung unterschiedlicher Instanzen möglich. Dabei muss unterschieden werden zwischen der Abbildung einer einzelnen Instanz, beispielsweise der Bearbeitung einer Kundenanfrage, und der Darstellung aller Instanzen als Gesamtbetrachtung eines Systems. Wir untersuchen daher, inwieweit die Referenzansätze eine explizite Modellierung einzelner Instanzen ermöglichen und ob Wechselwirkungen zwischen den Instanzen im Modell erfasst werden können.

- *Dienste*
  Wir betrachten die Referenzansätze weiterhin im Hinblick auf Dienste. Wir untersuchen, ob in den Ansätzen ein Dienstbegriff vorgesehen ist und betrachten, wenn dies der Fall ist, die Auslegung von Diensten in den Modellen. Dabei ist insbesondere die Einordnung und auch Abgrenzung im Vergleich zum Prozessbegriff von Interesse.

- *Semantische Fundierung*
  Zuletzt betrachten wir die semantische Fundierung der Referenzansätze. Alle vorgestellten Prozessmodelle unterstützen nicht nur die Darstellung einzelner Prozessabläufe, sondern auch die Modellierung einer den Abläufen zugrundeliegenden Prozesslogik im Sinne einer Spezifikation. Die Referenzansätze verwenden dazu unterschiedliche syntaktische

Notationen zur Angabe einer Spezifikation. Wir werden für die einzelnen Ansätze untersuchen, inwieweit mit den eingeführten syntaktischen Notationen ein formales semantisches Modell verbunden ist, das der syntaktischen Darstellung eines Prozessablaufs in den Prozessmodellen eine eindeutige Bedeutung zuweist.

5.1.2 Referenzmodelle

Wir untersuchen und vergleichen die folgenden Referenzansätze anhand der soeben vorgestellten Kriterien, die einen repräsentativen Überblick zu bestehenden Prozessmodellen ermöglichen:

(1) Petri-Netze [Rei91]

(2) Zustandsorientierte Workflows [LSS10]

(3) Quasar Enterprise [EHH+08]

(4) ArchiMate 2.0 [Ope12a]

(5) Unified Modeling Language 2.0, Aktivitätsdiagramme [Obj05a, Obj05b]

(6) Business Process Model and Notation (BPMN) [Obj11]

(7) Ereignisgesteuerte Prozessketten (EPK) [Sch92, KSN92, Sch01]

(8) Prozessnetze [Thu04]

(9) Unified Modeling Language 2.0, Sequenzdiagramme [Obj05a, Obj05b]

(10) Calculus of Communicating Systems (CCS) [Mil80]

(11) Communicating Sequential Processes (CSP) [Hoa85]

(12) Orc [KCM06]

Tabelle 5.1 zeigt die Eigenschaften der Referenzansätze, gegliedert nach Art der Prozessdarstellung, im Überblick.

5.1.2.1 Zustandsorientierte Prozessmodelle

Wir betrachten mit Petri-Netzen (1) und zustandsorientierten Workflows (2) zwei Ausprägungen von zustandsorientierten Prozessmodellen. Petri-Netze, die benannt sind nach dem Informatiker Carl Adam Petri, stellen Abläufe anhand von Tokens dar, die auf Basis einer einfachen Regellogik durch ein Petri-Netz bewegt werden können. Ein Petri-Netz besteht aus Stellen und Transitionen, die mit gerichteten Kanten verbunden sind. Dabei kann eine Stelle nur mit einer Kante und umgekehrt verbunden werden. Folglich kann ein Petri-Netz formal erfasst werden als Tripel (P,T,F) mit P als Menge der Stellen, T als Menge der Transitionen und $F \subseteq (P \times T) \cup (T \times P)$ als Relation zur Darstellung der Kanten. [Rei91]

Eine Stelle kann eine beliebige Anzahl von Tokens aufnehmen. Die aktuelle Token-Belegung aller Stellen spiegelt den Zustand eines Netzes wider. Der Zustand eines Netzes wird verändert durch das Schalten von Transitionen. Beim Schalten einer Transition werden aus den Stellen

	Modellierungs-ansatz	Prozess-darstellung	Zeitmodell	Akteure	Instan-zen	Dienste	Semantische Fundierung
Zustands-orientiert	(1) Petri-Netze	Sequentielle Zustandsfolge	qualitativ	nein	nein	nein	operationelle Semantik
	(2) Zustands-orient. Workflows	Sequentielle Zustandsfolge	qualitativ	nein	nein	nein	operationelle Semantik
Aktions-orientiert	Modell in dieser Arbeit	Ereignismenge mit Zeitmodell	qualitativ und quantitativ	ja	ja	ja	formalisiert
	(3) Quasar Enterprise	Aggregation von Aktionen	keines	ja	nein	ja	informell
	(4) ArchiMate	Aktionensequenz	keines	ja	nein	ja	informell
	(5) UML Aktivi-tätsdiagramme	Aktionenfolge (seq./parallel)	qualitativ	ja	nein	nein	informell, formal mit Zusätzen
	(6) BPMN	Aktionenfolge (seq./parallel)	qualitativ	ja	ja	rein technisch	informell, formal mit Zusätzen
	(7) EPK	Funktionen und Er-eignisse (Zustände)	qualitativ	ja	nein	nein	informell, formal mit Zusätzen
Nachrichten-orientiert	(8) Prozessnetze	Ein-/Ausgabe-verhalten	keines	ja	nein	nein	formalisiert
	(9) UML Se-quenzdiagramm	Nachrichtenfolge	qualitativ	ja	nein	nein	informell, formal mit Zusätzen
Prozess-algebren	(10) CCS	Operationelle Ak-tionsbeschreibung	qualitativ	nein	nein	nein	operationelle Semantik
	(11) CSP	Operationelle Ak-tionsbeschreibung	qualitativ	nein	nein	nein	denotationelle Semantik
	(12) Orc	Operationelle Ak-tionsbeschreibung	qualitativ	nein	ja	nein	operationelle Semantik

Tabelle 5.1: Prozessmodelle im Überblick

mit Kanten, die zur Transition hinführen, Tokens entnommen und gleichzeitig zu den Stellen, zu denen Kanten von der Transition führen, hinzugefügt. Die Kanten können gewichtet sein und so die Anzahl der Tokens festlegen, die bei der Schaltung entfernt beziehungsweise hinzugefügt werden. Die Summe der entfernten Tokens muss dabei nicht der Summe der hinzugefügten Tokens entsprechen.

Basierend auf einer Anfangsmarkierung der Stellen im Petri-Netz kann nun im Greedy-Verfahren jede Transition schalten, solange genug eingehende Tokens vorhanden sind. Die Semantik eines Petri-Netzes kann im Sinne einer operationellen Semantik als Menge von sequentiellen Zustandsfolgen angegeben werden, die sich aus der Startbelegung der Stellen ableiten lassen. Jede Zustandsfolge lässt sich dabei als Prozess auffassen, der anschaulich anhand der Token-Verschiebungen ausgeführt wird. Petri-Netze ermöglichen somit einfache, intuitive und realisierungsnahe Abbildungen von Abläufen, die auf einem präzisen semantischen Modell basieren.

Abläufe werden in Petri-Netzen auf einer abstrakten Ebene beschrieben. Zur Darstellung von Geschäftsprozessen ist eine Interpretation der Stellen und Transitionen erforderlich, unter anderem durch Ergänzung von Datentypen (vgl. [AS11]). Petri-Netze in ihrer Grundform unterstützen keine Modellierung von Akteuren, so dass keine Zuordnung von Transitionen zu Akteuren erfolgen kann. Entsprechend können Prozesse, die mehrere Instanzen enthalten, nicht ohne weitere Interpretation in solchen Netzen dargestellt werden. Auch ist keine Modellierung von Diensten vorgesehen.

Petri-Netzen liegt ein qualitatives Zeitmodell zugrunde. Durch die Betrachtung von Zustandsfolgen können relative Aussagen zum Auftrittszeitpunkt der Zustände getroffen werden. Diese können anhand sogenannter Occurrence Netze nachvollzogen werden, in denen sich die möglichen Zustandsfolgen eines Petri-Netzes abbilden lassen (vgl. [Haa99, GP09]). Die Kausalität zwischen Zustandsübergängen kann sowohl intensional aus den Transitionen, als auch extensional aus der Menge aller Zustandsfolgen abgeleitet werden. Es existieren zahlreiche Erweiterungen, auf Basis derer auch quantitativ temporale Aussagen in Petri-Netzen bezüglich des Eintrittszeitpunkts von Zuständen und der Dauer von Transitionen getroffen werden können (vgl. [GMMP91, CC88]).

Als zweiten Referenzansatz betrachten wir zustandsorientierte Workflows. Diese bauen prinzipiell auf Petri-Netzen auf, erweitern diese allerdings um verschiedene Beschreibungselemente und erhöhen somit die Ausdruckskraft der Prozessmodellierung. In den Workflows werden Zustände auf Basis von typisierten Variablen beschrieben. Ein Prozess wird analog zum Petri-Netz als sequentielle Zustandsfolge aufgefasst, wobei in zustandsorientierten Workflows jeder Zustand durch eine Belegung der eingeführten Variablen entsprechend ihres Typs charakterisiert ist.

Die semantische Definition von Prozessen erfolgt durch Festlegung von Start- und Endzuständen für einen Prozess, die als Kontrollpunkte angesehen werden, sowie durch Angabe einer Zustandsübergangsrelation, in der eine direkte Überführung einer Variablenbelegung in eine andere Belegung beschrieben wird. Die Variablen werden zudem differenziert in Eingabe- und Ausgabevariablen sowie interne Variablen. Zum Start des Prozesses wird eine Belegung der Eingabevariablen vorausgesetzt. Die Semantik des zustandsorientierten Workflows ergibt eine Menge von sequentiellen Zustandsfolgen, die ausgehend von einer Belegung der Eingabevariablen alle im Prozess auftretenden Zustände bis hin zu einem Endzustand beschreiben, in dem

die Ausgabevariablen belegt sind. Neben solchen endlichen Abläufen werden auch unendliche Abläufe ohne Endzustand unterstützt. Die Semantik von zustandsorientierten Workflows kann als operationelle Semantik einer Zustandsmaschine aufgefasst werden.

Die Eigenschaften der Workflows entsprechen im Wesentlichen denen von Petri Netzen. Das Modell sieht keine explizite Erfassung von Akteuren, Instanzen und Diensten vor. Das Zeitmodell der Workflows ist qualitativ aufgebaut, Kausalitäten zwischen Zustandsübergängen werden jedoch im Kontrast zu Petri-Netzen rein extensional anhand der Menge aller Zustandsfolgen erfasst.

5.1.2.2 Aktionsorientierte Prozessmodelle

Wir wenden uns nun den aktionsorientierten Ansätzen zu, bei denen nicht die Zustände, sondern die Zustandsübergänge im Fokus stehen, die durch Aktionen vollzogen werden. Unser in den letzten Kapiteln vorgestelltes Prozessmodell ist aktionsorientiert definiert. Wir werden uns allerdings zunächst auf die Diskussion von Referenzansätzen konzentrieren und im folgenden Abschnitt im Rahmen der Bewertung auf die Einordnung unseres Modells zurückkommen. Wir betrachten fünf aktionsorientierte Referenzansätze.

Die ersten beiden Referenzansätze sind Konzeptmodelle, die Aussagen zur Prozessmodellierung treffen. Das Konzeptmodell in Quasar Enterprise (3) betrachtet einen Prozess als eine Menge von Schritten, die auch als Aktivitäten bezeichnet werden. Die Schritte können in mehreren Stufen zunehmend in Teilschritte zerlegt und damit weiter verfeinert werden. [MF11] führt dazu eine Konzepthierarchie ein, die einzelne elementare Schritte zu Aufgaben und Aufgaben wiederum zu Prozessen aggregiert. Es existiert allerdings keine präzise Abgrenzung der Ebenen. Quasar Enterprise nimmt keine formale Differenzierung zwischen der Beschreibung einzelner Prozesse und der zugrundeliegenden Semantik vor und erfasst semantische Aspekte von Prozessen nur in informeller Weise.

Das Konzeptmodell beinhaltet kein explizites Zeitmodell und entspricht somit unseren ungeordneten Prozessen. Zeitliche Aspekte werden implizit durch die Angabe von UML Sequenz- und Aktivitätsdiagrammen erfasst, die wir im Folgenden noch genauer betrachten werden. Quasar Enterprise definiert Akteure, stellt aber keinen expliziten Bezug zwischen den Handlungen in einem Prozess und den Akteuren her. Das Konzeptmodell beinhaltet eine Dienstdefinition und betrachtet Dienste prinzipiell als Außensicht auf einen Geschäftsprozess. Da Prozesse aber nur informell als Aktivitätenmenge charakterisiert sind und die Aktivitäten nicht auf Akteure bezogen werden, lässt sich die Außensicht und damit ein Dienst nicht präzise abgrenzen. In Quasar Enterprise ist keine Differenzierung zwischen möglichen Instanzen eines Prozesses vorgesehen.

Als weiteres Konzeptmodell betrachten wir ArchiMate (4). ArchiMate führt unterschiedliche Konzepte zur Beschreibung von geschäftlichem Verhalten ein. Prozesse werden in ArchiMate als eine Folge von Aktivitäten angesehen. ArchiMate enthält allerdings keine syntaktische und semantische Definition von Prozessen und verweist dafür auf BPMN, das wir im Folgenden diskutieren werden. Entsprechend enthält ArchiMate auch kein Zeitmodell für Prozesse.

ArchiMate lässt die Modellierung von Akteuren zu und bezieht diese auf Prozesse, differenziert aber nicht zwischen ausführenden und beteiligten oder beobachtenden Akteuren. Mehrere

Instanzen eines Prozesses sind im Modell nicht explizit adressiert. In ArchiMate werden Dienste einerseits als Schnittstellenperspektive auf einen Prozess betrachtet, andererseits stellen aber auch Dienste Funktionalitäten in der Prozessausführung zur Verfügung. Allerdings wird auch in ArchiMate keine präzise Abgrenzung zwischen Dienst und Prozess vorgenommen. Dies liegt auch darin begründet, dass der Schnittstellenbegriff nur informell definiert ist.

UML Aktivitätsdiagramme (5) und BPMN (6) stellen zwei weit verbreitete Formen der graphischen Prozessmodellierung dar, die durch die Object Management Group verwaltet werden. Da die beiden Ansätze weitreichende Parallelen aufweisen, werden wir sie gemeinsam diskutieren. Wir haben die UML Notation bereits in unseren Beispielen des letzten Kapitels eingesetzt. Für Details zur in UML und BPMN eingesetzten Notation verweisen wir auf [Obj05a, Obj11]. UML Aktivitätsdiagramme und die BPMN beschreiben graphisch die sequentielle und parallele Verknüpfung von Aktionen. Damit eignen sie sich sowohl zur Darstellung einzelner Prozessabläufe wie auch als syntaktische Spezifikation von Prozessen.

Die Semantik für Prozessspezifikationen, die mit UML oder BPMN Diagrammen modelliert sind, ist von der Object Management Group nur in informeller Weise festgelegt. Sie basiert auf einer Erfassung des Kontroll- beziehungsweise Datenflusses entlang der Aktionen eines Diagramms. UML und BPMN stellen dazu unterschiedliche Kontrollstrukturen und Schleifen bereit, beispielsweise Verzweigungen in Abhängigkeit einer Bedingung. Den Kontrollstrukturen liegt die Annahme zugrunde, dass die Aktionen Zustandsübergänge darstellen und in einer Kontrollstruktur der aktuelle Zustand des Ablaufs abgefragt werden kann. Die Zustände können explizit durch Datenflüsse modelliert werden, werden in vielen Fällen zugunsten einer vereinfachten Darstellung aber nur implizit vorausgesetzt. Dies birgt das Risiko von Ambiguitäten in der Auslegung eines unpräzise modellierten UML/BPMN Diagramms. Es existieren zahlreiche Ansätze zur Festlegung einer formalen Semantik für UML und BPMN Diagramme (vgl. [GGW98, EW01, WG08]), die als operationelle Semantik (Übersetzung in Automatenmodelle oder Petri-Netze) oder denotationelle Semantik ausgelegt sind.

Die Prozessdiagramme wenden ein qualitatives Zeitmodell an, indem sie sequentielle oder parallele Abfolgezusammenhänge zwischen den Aktionen darstellen. Die graphischen Prozessdarstellungen ermöglichen eine intensionale Erfassung von Kausalitäten zwischen Aktionen, da Abhängigkeiten direkt in graphischer Weise modelliert werden.

Sowohl UML Aktivitätsdiagramme als auch BPMN Diagramme unterstützen die Modellierung von Akteuren. Dazu werden die Aktionen eines Diagramms zeilen- oder spaltenweise in Verantwortungsbereiche bestimmter Akteure (engl. Swimlanes) aufgeteilt. Die Akteurszuordnung beschränkt sich auf ausführende Akteure; beteiligte oder beobachtende Akteure können nur indirekt durch Objektflüsse modelliert werden. Die Modellierung von Instanzen ist in UML nicht vorgesehen, BPMN hält hierzu dagegen eine spezielle Notation bereit. Allerdings werden auch Instanzen nur informell erfasst. In UML Aktivitätsdiagrammen sind keine besonderen Beschreibungskonstrukte zur Darstellung von Diensten vorgesehen. Die BPMN Notation kennt Dienste, interpretiert diese allerdings in einem technischen Sinne als rechnergestützte Realisierung einer Aktion in einem Prozess. Dienste werden dabei unabhängig von Akteuren betrachtet.

Ereignisgesteuerte Prozessketten (7) wurden im Rahmen des ARIS Referenzmodells [Sch92] eingeführt und dienen zur graphischen Modellierung geschäftlicher Abläufe. Wenngleich inzwi-

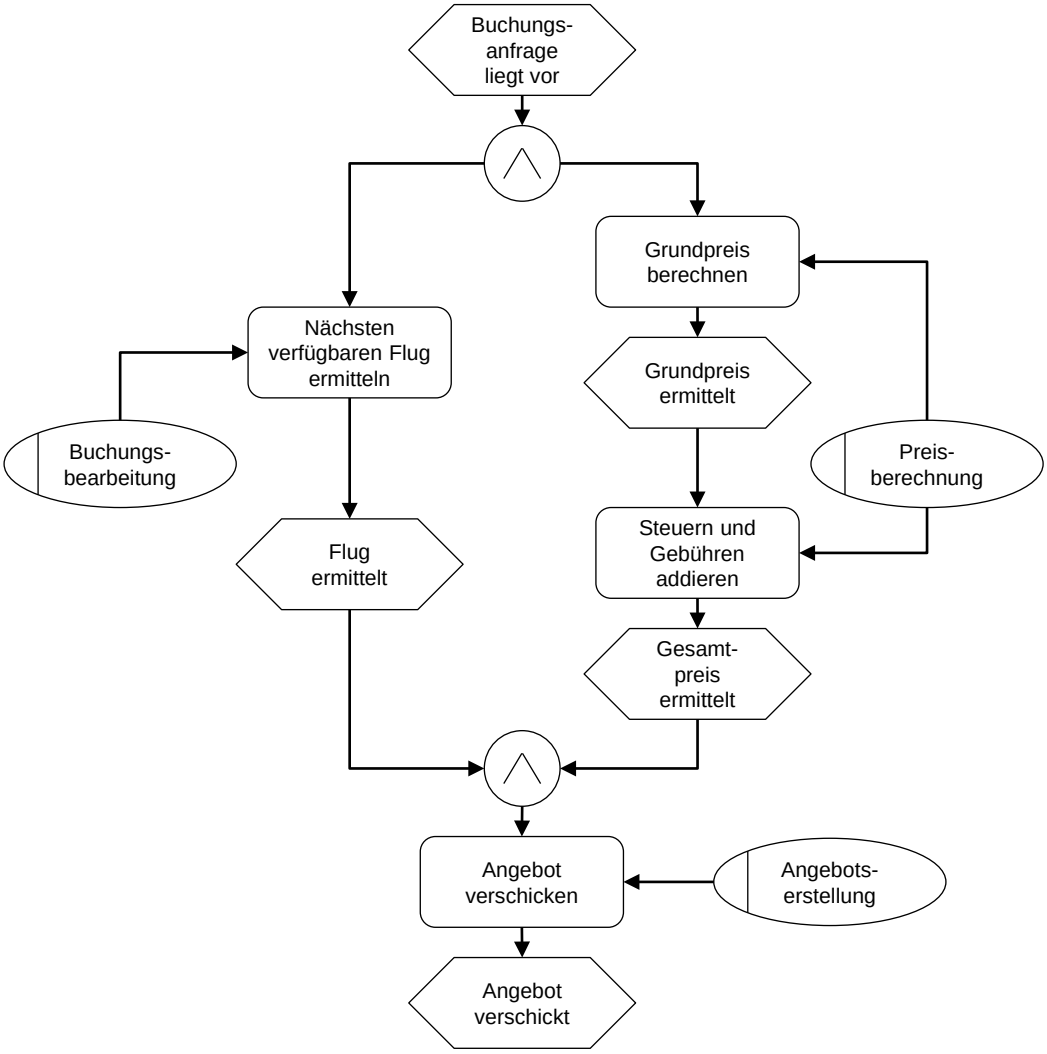

Abbildung 5.1: Spezifikation der Anfragebearbeitung bei Holiday Airways, dargestellt als ereignisgesteuerte Prozesskette (EPK)

schen auch BPMN Diagramme in ARIS zum Einsatz kommen, sind Prozessketten nach wie vor eine etablierte Form der Prozessdarstellung. Sie werden in ihrer heutigen Form auch als erweiterte ereignisgesteuerte Prozessketten (eEPK) betrachtet. Wir haben in Abbildung 5.1 exemplarisch den Prozess der Anfragebearbeitung durch Holiday Airways als EPK modelliert. Vergleichbar mit UML Aktivitätsdiagrammen und BPMN Diagrammen basiert die Prozessbeschreibung in EPK ebenfalls auf einer sequentiellen und parallelen Verknüpfung von Aktionen. Diese werden dargestellt als Rechtecke mit abgerundeten Ecken. Darüber hinaus wird der Zustand, der sich aus der Durchführung einer Aktion ergibt, in Sechsecken explizit modelliert. In EPK werden Aktionen als Funktionen und Zustände als Ereignisse bezeichnet. Funktionen und Ereignisse kommen in EPK grundsätzlich in alternierender Reihenfolge vor, um jeweils das Ergebnis einer Funktionsausführung als Ereignis festzuhalten. Die sequentielle Ausführung wird durch gerichtete Kanten dargestellt. Die parallele Ausführung von Funktionen wird, wie in der Abbildung gezeigt, durch ein logisches Und-Symbol gekennzeichnet, das am Anfang und Ende des parallelen Ablaufs zum Einsatz kommt.

In einer EPK kann analog zur UML und BPMN Notation sowohl ein einzelner Ablauf eines Prozesses als auch dessen Spezifikation dargestellt werden. Wir haben in Abbildung 5.1 bewusst eine in dieser Hinsicht zweideutige Prozessdarstellung gewählt, um die Dualität zu verdeutlichen. Dies erhöht einerseits die Flexibilität in der Modellierung, lässt allerdings auch Ambiguitäten in der Auslegung eines Prozessmodells zu. Für die Spezifikation von Prozessen stehen in EPK ebenfalls graphische Notationen zur Beschreibung von Kontrollstrukturen und Schleifen zur Verfügung. Vergleichbar mit den anderen graphischen Notationen ist die Semantik einer EPK-basierten Spezifikation nur informell beschrieben. Jedoch existieren auch für EPK verschiedene Ansätze zur Formalisierung der Semantik (vgl. [CS94, NR02]).

EPK basieren genauso wie die bereits diskutierten graphischen Notationen auf einem qualitativen Zeitmodell, das eine Darstellung von intensionalen Kausalitäten zwischen Funktionen durch sequentielle Darstellung im Modell ermöglicht. Den Funktionen einer EPK können Akteure in Ellipsen mit einer vertikalen Linie zugeordnet werden. Dabei wird nicht zwischen den die Funktion ausführenden und den an dieser beteiligten Akteuren differenziert. EPK sehen keine Instanzierung von Abläufen vor und verfügen über keine Notation zur expliziten Darstellung von Diensten.

5.1.2.3 Nachrichtenorientierte Prozessmodelle

Nachrichtenorientierte Prozessmodelle sind eine spezielle Ausprägung von aktionsorientierten Modellen, bei denen alle Aktionen als Nachrichten betrachtet werden. Wir untersuchen zwei nachrichtenorientierte Referenzansätze, die Prozessnetze (8) und UML Sequenzdiagramme (9).

Im Kontext der Prozessnetze wird ein Prozess als Funktion aufgefasst. Ein Prozess wird beschrieben anhand einer Menge von typisierten Ein- und Ausgabeports sowie einer Verhaltensfunktion, die Eingaben an den Eingabeports auf Ausgaben an den Ausgabeports abbildet. Aus der Verhaltensfunktion kann direkt die formale Semantik eines Prozesses als Menge von Ein-/Ausgabeabbildungen abgeleitet werden. Mehrere Prozesse können zu Prozessnetzen komponiert werden. Ein Prozessnetz lässt sich wiederum als Prozess auffassen. Dazu werden auch für

das Prozessnetz typisierte Ein- und Ausgabeports definiert. Jeder Ein- und Ausgabeport eines Prozesses im Netz wird entweder mit einem Port eines anderen Prozesses im Netz verbunden oder aber mit einem Port des Prozessnetzes. Im ersten Fall ist der Port ein interner Port des Netzes, im zweiten Fall ein externer Port, der wiederum mit anderen Prozessnetzen verbunden werden kann.

In Prozessnetzen ist kein explizites Zeitmodell definiert. Insbesondere werden die Ein- und Ausgaben eines Prozesses nicht über die Zeit hinweg erfasst, sondern punktuell in einer Mitte-zu-Mitte-Logik. Es können jedoch Kausalitäten zwischen den Nachrichten aus den Ein-/Ausgabefunktionen abgeleitet werden, wenn davon ausgegangen wird, dass eine Ausgabe als Wirkung einer Eingabe anzusehen ist. Anhand der Struktur eines Prozessnetzes können die Abhängigkeiten zwischen Nachrichten präzise nachvollzogen werden. Ein Prozessnetz lässt keine Zyklen zu, da für diese auf Grund der schwachen Kausalität nicht zwingend ein Fixpunkt existiert.

Den Prozessen in Prozessnetzen können Akteure zugewiesen werden. Die Verhaltensfunktion der Prozesse in Prozessnetzen charakterisiert dabei das Verhalten der Akteure. Der Referenzansatz differenziert nicht zwischen mehreren Instanzen eines Prozesses und enthält keine explizite Definition von Diensten. Jedoch zeigt die Auslegung eines Prozesses als Verhaltensfunktion und die Zuweisung von Akteuren zu Prozessen weitreichende Parallelen zur Dienstdefinition, wie wir sie in unserem Modell eingeführt haben.

Eine weitere Form der nachrichtenorientierten Prozessmodellierung sind UML Sequenzdiagramme. Diese eignen sich besonders zur Darstellung von Interaktionen zwischen Akteuren. Wir haben Sequenzdiagramme bereits in einigen Beispielen unseres Modells eingesetzt. Ein Sequenzdiagramm zeichnet einem qualitativen Zeitmodell folgend Nachrichten auf, die zwischen Akteuren versendet werden. Die Nachrichten werden dabei als gerichtete Kanten in ihrer zeitlichen Reihenfolge erfasst. Eine Sequenz von Nachrichten kann dabei als Prozess aufgefasst werden. Prinzipiell können in einem Sequenzdiagramm sowohl einzelne Prozesse als auch Spezifikationen für Prozesse angegeben werden. Die UML Definition für Sequenzdiagramme enthält unter anderem auch Kontrollstrukturen und Schleifen. Jedoch ist deren Semantik nur informell definiert. Insofern eignen sich Sequenzdiagramme vor allem zur Darstellung einzelner Interaktionsfolgen.

Akteure können in Sequenzdiagramme explizit anhand sogenannter Lifelines dargestellt werden. Die Kanten der Nachrichten gehen von diesen Lifelines ab oder sind auf diese gerichtet. Die Darstellung mehrerer Prozessinstanzen ist in Sequenzdiagrammen nicht vorgesehen. Genauso sind auch Dienste nicht explizit modellierbar, jedoch sind Dienstaufträge und -wirkungen im Sinne unseres Modells auf Grund der Ausrichtung an Akteuren einfach in solchen Diagrammen zu erfassen als alle Kanten, die zwischen Akteuren verlaufen.

5.1.2.4 Prozessalgebren

Zum Abschluss betrachten wir noch Prozessalgebren (10, 11, 12) als formale Modelle zur syntaktischen und semantischen Beschreibung von Prozessen. Da die drei Prozessalgebren weitreichende Parallelen aufweisen, diskutieren wir sie gemeinsam. Prozessalgebren verfolgen einen operationellen Ansatz zur Darstellung von Prozessen. Die einzelnen Schritte eines Prozesses

werden dabei als elementare Aktionen oder Ereignisse aufgefasst. Die elementaren Aktionen können anhand unterschiedlicher Kompositionsoperatoren zu Abläufen zusammengesetzt werden. Die Prozessalgebren stellen dazu eine syntaktische Notation bereit. Typische Kompositionsoperationen sind die Präfixkomposition zur Ergänzung einer einzelnen Aktion, die synchrone und asynchrone parallele Komposition sowie die deterministische und nicht deterministische Auswahl zwischen zwei Prozessen. Zusätzlich werden auch Operatoren zur Transformation von Prozessen bereitgestellt sowie primitive Prozesskonstrukte wie ein Symbol für einen abgeschlossenen Prozess.

Alle Prozessalgebren verbinden die formale syntaktische Notation mit einer Semantik. Bei CCS und Orc (10, 12) wird diese als operationelle Semantik angegeben, so dass eine Prozessspezifikation in der jeweiligen Prozessalgebra direkt in ein Automatenmodell übersetzt werden und auf diese Weise die Menge aller zulässigen Abläufe bestimmt werden kann. Zu CSP (11) existieren zahlreiche semantische Fundierungen, beispielsweise eine denotationelle und operationelle Semantik in [RHB98]. Prozessalgebren folgen einem qualitativen Zeitmodell. Zeitliche Abhängigkeiten zwischen den Aktionen können direkt als intensionale Kausalitäten aus den Kompositionsoperatoren der Prozessalgebra gefolgert werden.

Prozessalgebren nehmen keine explizite Modellierung von Akteuren vor, interpretieren aber Aktionen beziehungsweise Ereignisse als Interaktionen zwischen Systemen. In den Algebren wird keine explizite Dienstdefinition vorgenommen. In Orc (12) ist allerdings ein dezidiertes Sprachkonstrukt zur Modellierung von Prozessinstanzen vorgesehen.

5.1.3 Bewertung

Die Diskussion im letzten Abschnitt zeigt illustrativ anhand ausgewählter Beispiele, dass ein breites Spektrum unterschiedlicher Modelle zur Darstellung von Prozessen existiert. Die diskutierten Referenzansätze setzen dabei unterschiedliche Schwerpunkte in der Betrachtung von Prozessen. Prozessalgebren sowie Petri-Netze ermöglichen einer präzise syntaktische Erfassung von Abläufen und verbinden diese mit einem fundierten semantischen Modell, so dass Prozesse formal präzise beschrieben werden können. Konzeptmodelle fokussieren auf die Einordnung des Prozessbegriffs mit anderen Konzepten zur Verhaltensbeschreibung im betrieblichen Kontext, beispielsweise Diensten. Dafür verzichten sie auf eine formale Definition von Prozessen. Die graphischen Ansätze, die wir besprochen haben, sind in der Praxis in großer Breite etabliert, da sie eine anschauliche und intuitive Beschreibung von Prozessabläufen ermöglichen. Sie verzichten auf eine formale Semantik zugunsten einer höheren Flexibilität in der Prozessmodellierung, die allerdings auch zu Ambiguitäten führen kann.

Im Vergleich zu den Referenzansätzen sehen wir den Hauptbeitrag unseres Modells in der Integration der unterschiedlichen soeben vorgestellten Schwerpunkte. Unser Modell weist einerseits eine formale syntaktische und semantische Fundierung geschäftlicher Abläufe auf und ermöglicht deren Darstellung auf Basis von drei unterschiedlichen Zeitmodellen. Andererseits nehmen wir im Modell auch eine präzise Einordnung und Abgrenzung der verschiedenen Konzepte zur Beschreibung von Unternehmensarchitekturen im Sinne eines Konzeptmodells vor. Beides ist erforderlich, um präzise konzeptuelle Aussagen zu dienstorientierten Architekturen treffen zu

können. In der Betrachtung geläufiger SOA-Begriffsauslegungen in Kapitel 2 haben wir gesehen, dass im Kontext dienstorientierter Architekturen eine Vielzahl von Konzepten zur Handlungsbeschreibung eingesetzt werden, die es präzise voneinander abzugrenzen gilt. Eine solche Abgrenzung ist aber nur auf Grundlage eines exakten Konzeptverständnisses möglich. Wir setzen daher unser formales Prozessmodell als Referenzpunkt ein, um alle weiteren Konzepte im SOA-Umfeld an diesem auszurichten und auf diese Weise auch Aussagen zur gegenseitigen Abgrenzung der Konzepte zu treffen.

Zudem haben wir in der Begriffsbildung und Modellierung besonderes Augenmerk darauf gelegt, dass unsere Terminologie für die eingeführten Konzepte dem intuitiven Begriffsverständnis entspricht. Die konsequente Diskussion des Modells anhand eines praxisnahen Beispiels hat die Anwendbarkeit des Modells gezeigt. Mit dem Einsatz von SOA geht der Anspruch einher, Abläufe auf fachlicher und technischer Ebene an Diensten auszurichten. Auf der technischen Ebene ist dies, wie in Kapitel 2 diskutiert, bereits ausreichend mit Implementierungsansätzen wie Web Services adressiert. Unser Modell ermöglicht auf der fachlichen Ebene eine präzise Einordnung und Beschreibung von Diensten und Prozessen und somit eine konsistente ebenenübergreifende Betrachtung einer dienstorientierten Architektur.

Im Kontrast zu einigen vorgestellten Ansätzen beschränken wir unsere Diskussion auf diejenigen Konzepte, die zur präzisen konzeptuellen Erfassung einer dienstorientierten Architektur erforderlich sind. Die im letzten Abschnitt vorgestellten Konzeptmodelle sowie auch die graphischen Ansätze verfolgen eine breitere Ausrichtung und ermöglichen somit auch die Erfassung weiterer Aspekte einer Unternehmensarchitektur wie zum Beispiel detaillierter Informationsmodelle. Für die Beschreibung einer dienstorientierten Architektur ist dies von nachgelagerter Bedeutung. Im Sinne der Übersichtlichkeit unseres formalen Modells haben wir solche Aspekte daher nicht berücksichtigt.

5.2 Informelle Modellanwendung

Wir haben im letzten Abschnitt einige Referenzansätze diskutiert, die eine anschauliche und damit intuitive Prozessdarstellung ermöglichen, beispielsweise UML Aktivitätsdiagramme oder BPMN. Wir haben allerdings auch festgestellt, dass graphische Notationen Ambiguitäten in sich tragen und damit die Aussagekraft des Modells reduzieren können. Die beiden Zielsetzungen der Anschaulichkeit und formalen Präzision sind zumindest in einem gewissen Maße konfliktär, da eine stärkere mathematische Fundierung in der Regel zu einer weniger intuitiven Darstellung führt und umgekehrt eine anschaulich, flexibel einsetzbare Notation häufig nicht die erforderlichen mathematischen Grundlagen für eine exakte Beschreibung aufweist und damit mehr Interpretationsspielraum zulässt.

Wir haben in der Diskussion unseres Modells im Kapitel 4 den Schwerpunkt auf die mathematische Fundierung gelegt, um die gesetzten Modellierungsziele einzuhalten und eine präzise Begriffsdefinition sowie Einordnung der Konzepte zu ermöglichen. Die im vorangegangenen Kapitel eingesetzten Abbildungen haben jedoch bereits gezeigt, dass formale Korrektheit nicht zwingend im Widerspruch zu einer anschaulichen Darstellung stehen muss.

Wir werden in diesem Kapitel nun noch einen Ausblick auf informelle Darstellungsformen von Prozessen und Regeln geben, in dem wir die bereits eingeführten Notation aufgreifen und insbesondere im Kontext der Regeln ergänzen. Wir werden die informellen Notationen in engen Bezug zu unserem Modell einführen, so dass sie konform zu den formalen Grundlagen eingesetzt werden können. Auf diese Weise ermöglichen wir eine vereinfachte graphische beziehungsweise tabellarische Darstellung von Prozessen und der zugrundeliegenden Prozesslogik, in der sich auf Grund des direkten Bezugs zum formalen Modell eine hohe Präzision erreichen lässt.

5.2.1 Anforderungen

Um die angesprochene Abwägung zwischen einer präzisen und einfach zu handhabenden graphischen beziehungsweise tabellarischen Darstellungsweise von Prozessen und Prozessregeln zu adressieren, stellen wir folgende drei Anforderungen an die informelle Modellierung:

- *Eindeutige Rückführbarkeit der informellen Notation auf das formale Modell*
  Die graphische beziehungsweise tabellarische Darstellung von Prozessen und Regeln soll dahingehend erfolgen, dass ein direkter Bezug der informellen Modellelemente zum formalen Modell besteht. Dazu fordern wir, dass eine Übersetzung der graphischen beziehungsweise tabellarischen Darstellung in das formale Modell gewährleistet ist. Auf diese Weise können auch auf der informellen Ebene präzise Aussagen zur Bedeutung der eingesetzten Modellelemente getroffen und diese gegeneinander entsprechend abgegrenzt werden.
- *Flexibilität im Grad der Formalisierung*
  Die informelle Darstellung der geschäftlichen Handlungen soll auf Basis der soeben geforderten Abbildung eine vollständige Formalisierung ermöglichen, allerdings auch approximative Darstellungsweisen unterstützen. Damit soll dem Modellierenden die Flexibilität gegeben werden, den Grad der Formalisierung je nach gegebenem Einsatzfeld und Anforderungsprofil der Modellierungsaufgabe gestalten zu können.
- *Intuitiv erfassbare Darstellung*
  Zuletzt soll die informell Darstellung ein intuitives Verständnis des Modells fördern. Ziel der bildlichen Darstellung ist, dass – im Gegensatz zur formalen mathematischen Notation – Abläufe und deren Zusammenhänge ohne Kenntnisse des zugrundeliegenden mathematischen Modells schnell und intuitiv erfasst werden können. In diesem Zusammenhang werden wir uns soweit möglich an etablierten Notationen orientieren und ausrichten.

5.2.2 Informelle Prozessdarstellung

Wir haben in den Abbildungen zum Kapitel 4 bereits drei unterschiedliche graphische Darstellungsformen für Prozesse eingesetzt: Hasse-Diagramme, UML Aktivitätsdiagramme sowie Sequenzdiagramme. Wir werden diese nun nochmals systematisch aufgreifen und ihre Eigenschaften und Anwendungsmöglichkeiten im Hinblick auf die Darstellung von Prozessen und Spezifikationen erläutern.

Graphischer Beschreibungsansatz	Darstellung von Prozessen möglich?	Darstellung von Spezifikationen möglich?
Hasse-Diagramm	ja	nein
UML Aktivitätsdiagramm	ja	ja (unter Einhaltung von Konventionen)
UML Sequenzdiagramm	ja	eingeschränkt

Tabelle 5.2: Graphische Darstellbarkeit unseres Prozessmodells

Tabelle 5.2 zeigt die Anwendbarkeit der graphischen Beschreibungsansätze für unser Modell im Überblick. Wir beschränken uns neben den Hasse-Diagrammen auf die Betrachtung von UML Diagrammen. Auf die im letzten Abschnitt eingeführten EPK sowie BPMN lassen sich jedoch die Überlegungen, die wir im Folgenden für UML Aktivitätsdiagramme anstellen, auf Grund der weitreichenden Parallelen direkt übertragen.

5.2.2.1 Hasse-Diagramm

Hasse-Diagramme eignen sich zur anschaulichen Darstellung kausaler Prozesse. Jeder kausale Prozess $(E, \leq) \in Processes_K$ kann als gerichteter Graph aufgefasst werden. Dabei entspricht die Ereignismenge E der Menge der Knoten und die temporale Ordnung $\leq$ den gerichteten Kanten im Graphen. Ein Hasse-Diagramm stellt wiederum eine einfache graphische Repräsentation eines gerichteten Graphen dar (vgl. [DP02]). Somit kann eine kanonische Übersetzung eines kausalen Prozesses in ein Hasse-Diagramm vorgenommen werden. Auch die Umkehrung gilt, da Hasse-Diagramme eindeutig einem gerichteten Graphen zugeordnet werden können.

Die Darstellung von Prozessen in Hasse-Diagrammen basiert auf Ereignissen, die in den Knoten erfasst werden. Häufig werden jedoch vereinfachend die Ereignisse nicht explizit referenziert, sondern stattdessen die Aktionen, die in den Ereignissen zur Ausführung kommen, in den Knoten dargestellt. In Abbildung 5.2 haben wir den uns bereits bekannten Prozess der Anfragebearbeitung bei Holiday Airways als Hasse-Diagramm modelliert. Die Abbildung stellt dabei einen konkreten Bearbeitungslauf für einen Kunden, den wir in diesem Fall nicht näher spezifiziert haben, dar. In den Knoten verweisen wir auf Aktionen. Dies ist so zu interpretieren, dass jeder Knoten des Hasse-Diagramms für ein bestimmtes, aber nicht benanntes Ereignis steht, in dem die notierte Aktion durchgeführt wird.

In der Abbildung beschreiben wir jede Aktion sowohl in der formalen, in Kapitel 4 eingeführten Notation, als auch informell in kursiver Schrift. Zur Übersetzung in unser Prozessmodell ist die formale Notation ausreichend. Da die formalen Aktionsbezeichnungen jedoch ohne Kenntnis der entsprechenden Definitionen nur bedingt verständlich sind, ergänzen wir eine textuelle Beschreibung zur Verbesserung der intuitiven Erfassbarkeit. Hasse-Diagramme können auch rein informell aufgebaut werden. Dabei besteht allerdings die Gefahr, dass Aktionen in informeller Notation nicht eindeutig zugeordnet werden können und damit die Prozessdarstellung Ambiguitäten in sich birgt.

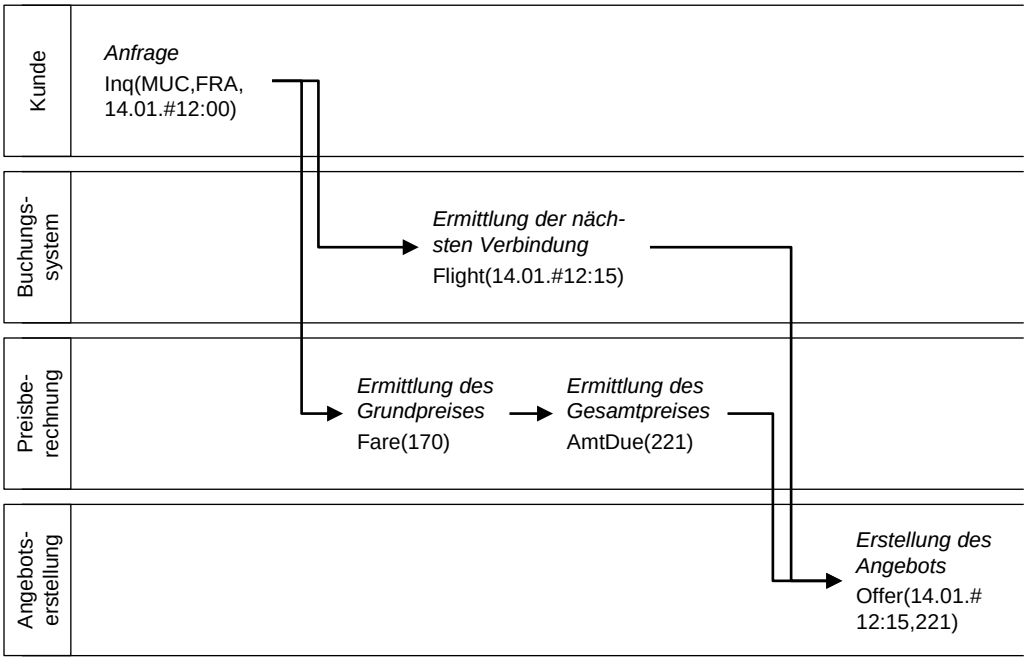

Abbildung 5.2: Buchungsanfrage eines Kunden bei Holiday Airways,
dargestellt als Hasse-Diagramm und gegliedert nach ausführenden Akteuren

In Hasse-Diagrammen können Akteure explizit erfasst werden. Zum einen ist eine direkte Angabe der Akteure in den Knoten gemäß der in Kapitel 4 eingeführte Notation möglich, beispielsweise $Inq(\text{MUC}, \text{FRA}, 14.01.\#12{:}00) \star cm, \{sb(cm), sp(cm)\}$ als Anfrage des Kunden Herr Müller cm an die entsprechenden Instanzen des Buchungs- und Preissystems $sb(cm), sp(cm)$ bei Holiday Airways. Diese Darstellungsweise ermöglicht die größte Flexibilität, vermindert jedoch die Lesbarkeit des Diagramms. In Abbildung 5.2 haben wir daher auf eine Angabe der Akteure in den Knoten verzichtet und stattdessen das Diagramm entlang der Zeilen (Swimlanes) gegliedert nach ausführenden Akteuren. Diese Form der Darstellung erweist sich als intuitiver, da sie aus UML bekannt ist. Sie ermöglicht jedoch keine explizite Modellierung der Akteure, die eine bestimmte Aktion beobachten können.

Allerdings kann für die Sichtbarkeit von Aktionen in einem solchen nach Swimlanes gegliederten Diagramm eine Konvention getroffen werden. Wir gehen dabei davon aus, dass eine Aktion für all diejenigen Akteure sichtbar ist, die der temporalen Ordnung folgend unmittelbar nach der Aktion eine weitere Aktion durchführen. In unserem Beispiel ist so die Anfrage des Kunden für das Buchungssystem und das System zur Preisberechnung sichtbar. Diese Konvention begünstigt insbesondere die Darstellung von Prozessen, in denen Zusammenhänge zwischen Aktionen akteurorientiert nach der Definition in Abschnitt 4.4.3 beschrieben werden. Jede Aktion folgt einer Aktion, die entweder durch den gleichen Akteur durchgeführt wird oder aber nach Konvention für den Akteur sichtbar sein muss. Die Darstellung ermöglicht zudem eine einfache Erfassung von Diensten, die die Akteure erbringen, an den Übergängen der Swimlanes.

Hasse-Diagramme eignen sich nicht zur syntaktischen Erfassung von Prozessspezifikationen, da definitionsgemäß keine Kontrollstrukturen oder Schleifen vorgesehen sind.

5.2.2.2 Aktivitätsdiagramm

Mit UML Aktivitätsdiagrammen können sowohl einzelne Prozesse wie auch deren Spezifikation dargestellt werden. In der Anwendung eines Aktivitätsdiagramms muss daher sorgfältig unterschieden werden, ob ein einzelner Ablauf oder die diesem zugrundeliegende Logik erfasst werden soll.

Die Abbildung einzelner Prozesse in UML Aktivitätsdiagrammen erfolgt prinzipiell nach den gleichen Regeln wie bei den Hasse-Diagrammen. Es bestehen lediglich geringfügige Unterschiede in der Darstellung. In UML werden Aktionen typischerweise in Rechtecken mit abgerundeten Ecken eingefasst. Parallelität wird mit vertikalen Linien, an denen die Kanten sich verzweigen beziehungsweise zusammengeführt werden, modelliert. Die Parallelen zwischen den beiden Diagrammtypen lassen sich direkt in der Abbildung 5.2 erkennen. Zudem haben wir Aktivitätsdiagramme als syntaktische Notation von Prozessen auch bereits zur Illustration unseres Beispiels in Kapitel 4, unter anderem in Abbildung 4.2, eingesetzt. Unsere Überlegungen bezüglich der Modellierung von aktiven und passiven Akteuren aus dem letzten Abschnitt gelten für UML Aktivitätsdiagramme in gleicher Weise. Dienste können somit anhand der Schnittstellenaktionen an den Übergängen der Swimlanes erfasst werden.

Aktivitätsdiagramme eignen sich auch zur Darstellung von Prozessspezifikationen, jedoch ist deren Semantik in den UML Spezifikationsdokumenten [Obj05a, Obj05b] nur informell definiert.

Wie bereits im Abschnitt 5.1 diskutiert finden sich aber in der Literatur zahlreiche Ergänzungen der UML Definition, die eine formale Semantik für die Diagramme einführen. Dazu müssen eine Reihe von Modellierungsannahmen getroffen werden. Wir beschränken uns in diesem Abschnitt auf einen kurzen Überblick zu diesen Annahmen und verweisen für eine ausführliche Diskussion auf [GGW98, EW01].

Eine in einem Aktivitätsdiagramm modellierte Prozessspezifikation kann formal als Spezifikation eines Akzeptors für Prozessabläufe betrachtet werden. Das Diagramm gibt somit zulässige Aktionenfolgen an. Ausgehend von einem Startzustand bewirkt dabei die Ausführung einer jeden Aktion einen Zustandsübergang. Der Zustand ist insbesondere von Bedeutung, wenn Kontrollstrukturen zum Einsatz kommen. Die Bedingung einer Kontrollstruktur in einem Aktivitätsdiagramm wird in Abhängigkeit des Zustands formuliert. Insofern ist für eine formale Erfassung einer Spezifikation in UML eine präzises Verständnis der Auswirkung von Aktionsdurchführungen auf den Zustand essentiell.

Anhand des exemplarischen Automatenmodells, das wir im Abschnitt 4.3.2.2 im Kontext kausaler Prozesse diskutiert haben, wird der Zusammenhang zwischen der Aktionenfolge im Aktivitätsdiagramm und den damit verbundenen Zustandsübergängen deutlich. Die Preisberechnung von Holiday Airways basiert dabei auf einer zu Beginn des Ablaufs erfolgten Kundenanfrage. Die Kundenanfrage versetzt das Preisberechnungssystem der Fluggesellschaft in den Zustand, auf dessen Grundlage die weiteren Aktionen zur Preisberechnung durchgeführt werden können. So setzt beispielsweise die Berechnung des Entfernungspreises voraus, dass die Zustandsvariablen für Abflug- und Zielflughafen belegt sind. Die Berechnung der Aufschläge hängt wiederum von den Zustandsvariablen Datum und Zielflughafen ab. Durch die Ausführung dieser Aktionen erfolgen jeweils Zustandsübergänge, in denen der Grundpreis sukzessive kalkuliert wird. Die Aktion zur Ausgabe des Grundpreises $Fare(\langle Value \rangle)$ bewirkt keinen Zustandsübergang, weist aber ebenfalls eine Abhängigkeit zu einer zuvor errechneten Zustandsvariable auf.

In der Praxis wird in der Modellierung von UML Aktivitätsdiagrammen häufig auf explizite Aussagen zum Zustand verzichtet und stattdessen ein implizites Zustandsverständnis vorausgesetzt. Dies kann jedoch zu Unklarheiten in der Deutung der Auswirkungen einer Aktion führen. Die Diskussion unseres Beispiels zur Preisberechnung hat gezeigt, dass eine exakte Übersetzung eines Aktivitätendiagramms in eine Spezifikation im Sinne unseres Prozessmodells nur erfolgen kann, wenn die durch die Aktionen veranlassten Zustandsübergänge präzise in formaler oder strukturiert informeller Weise beschrieben sind.

Als Mittel zur informellen Zustandsdarstellung haben wir in unseren Abbildungen, beispielsweise in Diagramm 4.4, auf die Objektflussnotation von UML zurückgegriffen. Auf diese Weise lässt sich explizit darstellen, welcher Zustand für die Ausführung der Folgeaktionen vorausgesetzt wird.

5.2.2.3 Sequenzdiagramm

UML Sequenzdiagramme eignen sich insbesondere zur Darstellung interaktiver Prozesse, die auf Nachrichten basieren. Sie sind ausgerichtet an miteinander kommunizierenden Akteuren,

die anhand von vertikalen Linien, den sogenannten Lifelines des Akteurs, dargestellt werden. Im Beispiel des Abschnitts 4.3.4.2 haben wir ein solches Diagramm bereits eingesetzt.

Im Hinblick auf unser Modell lassen sich sequentielle und temporale Prozesse darstellen, deren Aktionen als Nachrichten interpretiert werden können. In sequentiellen Prozessen werden die Nachrichten von oben nach unten in der gegebenen Ereignisreihenfolge modelliert. In temporalen Prozessen muss zusätzlich eine Zeitskala im Sequenzdiagramm eingeführt werden, um die Eintrittszeitpunkte der Ereignisse erfassen zu können. Nachrichten mit genau einem Empfänger werden als Pfeile erfasst, die von der Lifeline des sendenden Akteurs ausgehen und auf die Lifeline des empfangenden Akteurs gerichtet sind. Für Nachrichten mit mehreren Empfängern müssen entsprechend mehrere Pfeile eingezeichnet werden. Somit ist im Sequenzdiagramm eine Nachricht an mehrere Empfänger nicht von einer Menge einzelner Nachrichten an jeweils einen dieser Empfänger unterscheidbar. Interne Nachrichten eines Akteurs können als Punkte auf der entsprechenden Lifeline dargestellt werden.

Die UML Definition von Sequenzdiagrammen enthält auch Beschreibungselemente für Kontrollstrukturen. Damit können grundsätzlich Spezifikationen in Sequenzdiagrammen modelliert werden. Dies ist jedoch nur im eingeschränkten Rahmen möglich, da bestimmte Sprachkonstrukte wie zum Beispiel Schleifen fehlen und Kausalitäten zwischen Ereignissen nicht systematisch erfasst werden können. Zudem sind die Beschreibungselemente für Sequenzdiagramme wiederum nur informell definiert. Wir sehen daher den Hauptanwendungsbereich von Sequenzdiagrammen in der Erfassung nachrichtenorientierter Prozesse. Durch den Fokus auf Interaktionen können insbesondere Abläufe von Diensten zwischen Akteuren anschaulich erfasst werden. Die den Abläufen zugrundeliegende Dienstlogik kann in Form von Regeln beschrieben werden, auf deren informelle Formulierung wir im nächsten Abschnitt eingehen werden.

5.2.3 Informelle Regeldarstellung

Wir haben in unserer bisherigen Diskussion Prozessregeln formal als Relation zwischen einem Prozess und einer Fortsetzung dieses Prozesses betrachtet. Der fortgesetzte Prozess kann dabei zusätzliche Ereignisse gegenüber dem ursprünglichen Prozess enthalten. Eine solche Regeldarstellung ist in formaler Hinsicht vorteilhaft, um Prozessentwicklungen präzise erfassen zu können, entspricht jedoch nicht dem intuitiven Verständnis einer Regel als Ursache-/Wirkungszusammenhang. Typischerweise werden Geschäftsregeln in einer Wenn-Dann-Formulierung angegeben, so dass Wirkungen in Abhängigkeit von bestimmten ursächlichen Aktionen erfasst werden können.

Wir werden daher nun eine tabellarische Form der Regeldarstellung einführen, die die Idee eines Ursache-/Wirkungszusammenhangs aufgreift. Die vorgestellte Notation stellt eine Simplifizierung gegenüber der formalen Regeldefinition dar. In der informellen Notation sind somit nicht alle Prozessregeln abbildbar. Wir werden jedoch anhand unseres Praxisbeispiels der Fluggesellschaft Holiday Airways zeigen, dass die Ausdruckskraft der Notation für geschäftliche Abläufe wie im Beispiel formuliert ausreichend ist.

5.2.3.1 Basisnotation

Wir interpretieren die tabellarische Notation einer Prozessregel dahingehend, dass auf der linken Seite ursächliche Aktionen aufgeführt werden und auf der rechten Seite die Aktionen stehen, die aus den ursächlichen Aktionen folgen. Wir stellen eine Regel r mit ursächlichen Aktionen $a_1, a_2, \ldots$ und daraus abgeleiteten Wirkungen $b_1, b_2, \ldots$ wie folgt dar:

r	
$a_1, a_2, \ldots$	$b_1, b_2, \ldots$

Wir legen die Notation so aus, dass alle Aktionen auf der linken Seite eingetreten sein müssen, bevor die Aktionen auf der rechten Seiten folgen. Die Reihenfolge der Aktionen auf einer Seite ist dagegen irrelevant. Die Simplifizierung der tabellarischen Notation besteht darin, dass die ursächlichen Aktionen nicht im Kontext betrachtet werden. Für die Anwendung der Regel fordern wir ausschließlich, dass im beobachteten Prozess alle Aktionen $a_1, a_2, \ldots$ in beliebiger Reihenfolge aufgetreten sind. Wir fassen somit eine Regel in der tabellarischen Schreibweise als Aktionenrelation auf.

Die soeben eingeführte tabellarische Darstellung der Regel r lässt sich wie folgt in eine formale Prozessregel $\xrightarrow{r} \in \mathit{Rules}$ überführen:

$$\begin{aligned}\xrightarrow{r} \quad = \quad & \left\{\left(\left(E', \emptyset\right), \left(E', \emptyset\right)\right) : \exists E \in Ev(a_1, a_2, \ldots) : E' \subset E\right\} \\ & \cup \left\{\left((E, \emptyset), (E \cup F, E \times F)\right) : E \in Ev(a_1, a_2, \ldots) \wedge F = Ev(b_1, b_2, \ldots)\right\} \\ & \cup \left\{\left((E \cup F, E \times F), (E \cup F, E \times F)\right) : E \in Ev(a_1, a_2, \ldots) \wedge F = Ev(b_1, b_2, \ldots)\right\}\end{aligned}$$

Dabei übersetzt die Funktion $Ev \in \mathit{Actions} \times \mathit{Actions} \times \ldots \rightarrow \mathscr{P}(\mathit{Processes}_N)$ eine Sequenz von Aktionen in die Menge aller korrespondieren ungeordneten Prozesse, die die Aktionen in beliebiger Reihenfolge enthalten:

$$Ev(a_1, a_2, \ldots) = \{\{e_1, e_2, \ldots \in \mathit{Events}\} : e_1.a = a_1 \wedge e_2.a = a_2 \wedge \ldots\}$$

Die Übersetzung der tabellarischen Notation in eine formale Prozessregel besteht aus drei Teilen. Alle Prozesse, die (noch) nicht alle ursächlichen Aktionen enthalten, bleiben bei der Anwendung der Regel unverändert. Sobald ein Prozess alle ursächlichen Aktionen beinhaltet, ergänzt die Anwendung der Regel alle Wirkungen, die gemäß der Regel aus den Ursachen resultieren. Dabei wird eine temporale Ordnungsbeziehung zwischen Ursachen und Wirkungen eingeführt. Zuletzt lässt die Regelanwendung alle Prozesse, in denen bereits Wirkungen aus den Ursachen abgeleitet wurden, wiederum unberührt.

Aus der Definition der Regel r lässt sich direkt ableiten, dass für den Anwendungsbereich der Regel $\mathit{Scope}(r) = \{a_1, a_2, \ldots, b_1, b_2, \ldots\}$ gilt und die Regel ausschließlich die Aktionen $b_1, b_2, \ldots$ im Prozess ergänzt: $\mathit{Delta}(r) = \{b_1, b_2, \ldots\}$. Wir betrachten die Regel r zudem typischerweise als Regel mit erweitertem Anwendungsbereich $\hat{r}$, so dass zu Prozessen mit beliebigen Aktionen eine Entwicklungsaussage getroffen werden kann (vgl. Abschnitt 4.3.3.2).

In der tabellarischen Notation einer Regel r können mehrere Ursache-/Wirkungsbeziehungen in Zeilen untereinander dargestellt werden:

r	
$a_1, a_2, \ldots$	$b_1, b_2, \ldots$
$c_1, c_2, \ldots$	$d_1, d_2, \ldots$
$\vdots$	$\vdots$

Die Ableitungen erfolgen in diesem Fall alternativ. Wenn ein Prozess die Aktionen $a_1, a_2, \ldots$ enthält, leitet die Regel die Aktionen $b_1, b_2, \ldots$ ab. Enthält ein Prozess die Aktionen $c_1, c_2, \ldots$, so leitet die Regel die Aktionen $d_1, d_2, \ldots$ daraus ab. Für die Übersetzung in das formale Modell bedeutet dies, dass die Regel r als Vereinigung aus den einzelnen Ursache-/Wirkungsbeziehungen konstruiert werden kann:

$$\xrightarrow{r} = \xrightarrow{r_1} \cup \xrightarrow{r_2} \cup \ldots$$

Die Regeln $r_1, r_2, \ldots$ werden dabei gemäß der oben diskutierten Übersetzungsvorschrift für die einzelnen Zeilen der Tabelle konstruiert.

Die Ausdrucksmächtigkeit der bisher eingeführten tabellarischen Prozessregeln ist noch begrenzt, da Beziehungen zwischen konkreten Aktionen formuliert werden müssen. Häufig ist es jedoch wünschenswert, mit Prozessregeln Aussagen über Klassen von Aktionen treffen zu können. Dazu führen wir Variablen ein und lassen auf Basis von explizit formulierten Eigenschaften Aussagen über diese Variablen zu.

Im Sinne der Anschaulichkeit definieren wir die tabellarische Notation anhand eines konkreten Anwendungsbeispiels. Wir gehen davon aus, dass die Aktion $Msg(\langle content \rangle)$ den Versand einer Nachricht $\langle content \rangle$ bedeutet. Wir stellen die Variable in Spitzklammern dar, um sie syntaktisch von der Bezeichnung der Aktion abzugrenzen. Wir betrachten eine Regel r, die wie folgt definiert ist:

r	
$Msg(\langle input \rangle) : \langle input \rangle \in \mathbb{N}$	$Msg(\langle output \rangle) :$ $\langle output \rangle = \langle input \rangle + 1$

Die Regel schreibt vor, dass für jede erhaltene natürliche Zahl eine Nachricht versendet wird, in der die Zahl um eins inkrementiert wird. Die Regel ist somit semantisch äquivalent zu folgender Regel:

r	
$Msg(0)$	$Msg(1)$
$Msg(1)$	$Msg(2)$
$\vdots$	$\vdots$

Wir haben in der kompakten variablenbasierten Version der Regel eine formale Notation zur Beschreibung der Aktion gewählt. Dies ist nicht zwingend erforderlich. Die Aktionen können stattdessen auch anhand semiformaler oder textueller Aussagen definiert werden.

Wir kommen nun zurück auf unser Praxisbeispiel einer Buchungsanfrage bei Holiday Airways und werden jetzt die in Kapitel 4 formal definierten Prozessregeln ha_1, ha_2, ha_3, ha_4 in der informellen Notation diskutieren. Wir übernehmen dazu die bereits bekannte Bezeichnungen der fünf zur Buchungsanfrage gehörigen Aktionen und definieren die vier Prozessregeln wie folgt:

- ha_1 als Regel zur Ermittlung der Flugverfügbarkeit:

ha_1	
$Inq(\langle Orig\rangle, \langle Dest\rangle, \langle RDate\rangle)$	$Flight(\langle Date\rangle) : \langle Date\rangle$ ist der Zeitpunkt der nächsten verfügbaren Flugverbindung auf der Strecke von $\langle Orig\rangle$ nach $\langle Dest\rangle$ ab dem Zeitpunkt $\langle RDate\rangle$

- ha_2 als Regel zur Grundpreisberechnung:

ha_2	
$Inq(\langle Orig\rangle, \langle Dest\rangle, \langle Date\rangle)$	$Fare(\langle Fare\rangle) : \langle Fare\rangle$ ist der Entfernungspreis für die Strecke von $\langle Orig\rangle$ nach $\langle Dest\rangle$, zuzüglich 20%, falls $\langle Date\rangle$ ein Wochentag ist, und zuzüglich des Aufschlags für die Destination

- ha_3 als Regel zur Ermittlung der Gesamtpreises einschließlich Steuern und Gebühren:

ha_3	
$Fare(\langle Fare\rangle)$	$AmtDue(\langle AmtDue\rangle) :$ $\langle AmtDue\rangle = \langle Fare\rangle * 1.3$

- ha_4 als Regel zur Angebotserstellung für den Kunden:

ha_4	
$Flight(\langle Date\rangle),$ $AmtDue(\langle AmtDue\rangle)$	$Offer(\langle Date\rangle, \langle AmtDue\rangle)$

Die vier Prozessregeln zeigen, dass die tabellarische Notation in unterschiedlichen Formalitätsgraden eingesetzt werden kann. Während in den ersten beiden Regeln die Aktionszusammenhänge deskriptiv textuell beschrieben sind, kommt in den letzten beiden Regeln eine formale Notation zum Einsatz.

5.2.3.2 Akteurbezug

In der tabellarischen Regelbeschreibung haben wir bislang von Akteuren abstrahiert. Selbstverständlich sind Akteure aber auch in der informellen Formulierung von Regeln von großem Interesse. Wir legen die tabellarischen Regeln dahingehend aus, dass sie grundsätzlich aus der Perspektive eines Akteurs oder einer Gruppe von Akteuren aufgestellt werden. Somit ist jede Regel akteurorientiert definiert. Wir verwenden dazu folgende Notation, die wir am Beispiel der Regel zur Ermittlung der Flugverfügbarkeit einführen:

ha_1 (Buchungssystem *sb*)

$Inq(\langle Orig\rangle, \langle Dest\rangle, \langle RDate\rangle)$	$Flight(\langle Date\rangle) : \langle Date\rangle$ ist der Zeitpunkt der nächsten verfügbaren Flugverbindung auf der Strecke von $\langle Orig\rangle$ nach $\langle Dest\rangle$ ab dem Zeitpunkt $\langle RDate\rangle$
◀ Kunde *c*	▶ Angebotserstellung *so*

Wir notieren den Akteur oder die Gruppe von Akteuren, auf die die Regel bezogen ist, in der Kopfzeile in Klammern hinter der Regelbezeichnung. Wir gehen davon aus, dass der Akteur, auf den die Regel bezogen ist, die für die Regelanwendung ursächlichen Aktionen auf der linken Seite beobachten kann. Wir merken unter den Aktionen auf der linken Seite an, von welchem Akteur diese ausgehen und verwenden dazu das Symbol ◀. Die Wirkungen auf der rechten Seite der tabellarischen Notation werden von dem Akteur durchgeführt, auf den die Regel bezogen ist. In diesem Fall notieren wir unter den Aktionen anhand des Symbols ▶, für wen diese Aktionen sichtbar sind. Dabei kann es sich sowohl um einzelne Akteure, als auch um eine Akteursgruppe handeln. Im dargestellten Beispiel reagiert das Buchungssystem *sb* von Holiday Airways auf eine Anfrage, die von einem Kunden *c* erhalten wird, und leitet die ermittelte Flugverbindung intern an die Angebotserstellung *so* der Fluggesellschaft weiter.

Die eingeführte Notation für Akteure lässt sich problemlos in die formale Darstellung von Aktionen übersetzen. Dabei ist die linke Seite zu interpretieren als $Inq(\langle Orig\rangle, \langle Dest\rangle, \langle RDate\rangle) \star c, Act$ mit $Act \subseteq Actors$ und $sb \in Act$ und die rechte Seite entspricht einer Aktion $Flight(\langle Date\rangle) \star sb, \{so\}$.

Wenn eine Regel wie im diskutierten Beispiel ausschließlich Schnittstellenaktionen eines Akteurs erfasst, kann die tabellarische Darstellung direkt als Definition eines Dienstes betrachtet werden. Sie beschreibt in diesem Fall in intuitiver Weise die durch den Akteur zu erbringende Leistung (rechte Seite) in Abhängigkeit eines Dienstauftrags (linke Seite). In unserem Beispiel ist die Leistung, die das Buchungssystem innerhalb der Fluggesellschaft erbringt, als Dienst in der Regel ha_1 festgelegt.

Tabellarische formulierte Regeln können aber auch interne Abläufe eines Akteurs beschreiben. In diesem Fall kann das Verhalten eines Akteurs anhand von mehreren Regeln strukturiert beschrieben und zusammengesetzt werden. Dies ist der Fall für die Regeln ha_2, ha_3 zur Preisberechnung, die das Verhalten des entsprechenden Rechnersystems von Holiday Airways erfassen.

5.2.3.3 Komposition

Zuletzt betrachten wir noch die Komposition von tabellarisch formulierten Regeln. Auch in der tabellarischen Notation können Regeln mittels unterschiedlicher Operatoren zusammengesetzt werden. Dabei kann ein eindeutiger Bezug zu den korrespondierenden formalen Kompositionsoperatoren hergestellt werden. Wir betrachten zunächst die alternative und die parallele Komposition. Dabei gehen wir davon aus, dass zwei tabellarische Regeln wie folgt formuliert sind:

r_1	
$a_1, a_2, \ldots$	$b_1, b_2, \ldots$

r_2	
$a_1, a_2, \ldots$	$c_1, c_2, \ldots$

Diese zwei Regeln lassen sich in alternativer beziehungsweise paralleler Weise komponieren zu:

$r_1 \mid r_2$	
$a_1, a_2, \ldots$	$b_1, b_2, \ldots$
$a_1, a_2, \ldots$	$c_1, c_2, \ldots$

$r_1 \parallel r_2$	
$a_1, a_2, \ldots$	$b_1, b_2, \ldots, c_1, c_2, \ldots$

In der alternativen Komposition kann eine der Ableitungen, die in den Einzelregeln formuliert sind, gewählt werden. In der parallelen Komposition setzen wir voraus, dass in beiden Regeln eine Aussagen zu den gleichen ursächlichen Aktionen getroffen wird. Die parallele Komposition fordert, dass aus einer gegebenen Ursache die Vereinigung der Wirkungen der beiden Einzelregeln abgeleitet wird. Die tabellarische Auslegung der alternativen und parallelen Komposition entspricht damit dem formalen Kompositionsverhalten $r_1 \mid r_2$ und $r_1 \parallel r_2$.

In der Betrachtung der sequentiellen Komposition gehen wir von einer dritten Regel aus, die wie folgt definiert ist:

r_3	
$b_1, b_2, \ldots$	$c_1, c_2, \ldots$

In der tabellarischen Notation kann der Zwischenschritt der sequentiellen Komposition nicht dargestellt werden, da wir keine kausale Ordnung zwischen den Ursachen und den Wirkungen vorgesehen haben. Wir führen die sequentielle Komposition daher in leicht abgewandelter Form folgendermaßen ein:

$r_1;;r_3$	
$a_1, a_2, \ldots$	$c_1, c_2, \ldots$

Auf diese Weise abstrahieren wir vom Zwischenschritt und leiten aus den Ursachen der ersten Regel direkt die Wirkungen der zweiten Regel ab unter der Anforderung, dass die Wirkungen den ersten Regel wiederum ursächlich für die Wirkungen der zweiten Regel sind. Die sequentielle Komposition in der tabellarischen Notation lässt sich wie folgt unter Ausblendung des Zwischenschrittes in die uns bekannte formale Definition des sequentiellen Kompositionsoperators übersetzen:

$$r_1;;r_3 = [r_1;r_3]_{\prod_{a_1,a_2,\ldots,c_1,c_2,\ldots}}$$

Wir können nun die vorhin eingeführten tabellarischen Prozessregeln von Holiday Airways nach dem bereits bekannten Kompositionsschema $ha = (ha_1 \parallel (ha_2;;ha_3));;ha_4$ zusammensetzen, wobei wir für die sequentielle Komposition die abgewandelte Variante anwenden. Wir beginnen mit der Komposition der Regeln für die Preisberechnung $ha_2;;ha_3$:

$ha_2;;ha_3$	
$Inq(\langle Orig\rangle, \langle Dest\rangle, \langle Date\rangle)$	$AmtDue(\langle AmtDue\rangle)$: $\langle AmtDue\rangle$ ist der Gesamtpreis für die Strecke von $\langle Orig\rangle$ nach $\langle Dest\rangle$ am $\langle Date\rangle$, der nach Maßgabe der Regeln ha_2 und ha_3 ermittelt wird

Die Komposition $ha_2;;ha_3$ beschreibt den Dienst, den das System zur Preisberechnung gegenüber der Organisation der Fluggesellschaft erbringt. Wir komponieren diesen im nächsten Schritt in paralleler Weise mit dem Dienst des Buchungssystems:

$ha_1 \parallel (ha_2;;ha_3)$	
$Inq(\langle Orig\rangle, \langle Dest\rangle, \langle RDate\rangle)$	$Flight(\langle Date\rangle)$:$\langle Date\rangle$ ist der Zeitpunkt der nächsten verfügbaren Flugverbindung auf der Strecke von $\langle Orig\rangle$ nach $\langle Dest\rangle$ ab dem Zeitpunkt $\langle RDate\rangle$, $AmtDue(\langle AmtDue\rangle)$: $\langle AmtDue\rangle$ ist der Gesamtpreis für die Strecke von $\langle Orig\rangle$ nach $\langle Dest\rangle$ am $\langle RDate\rangle$, der nach Maßgabe der Regeln ha_2 und ha_3 ermittelt wird

Die Komposition stellt nun den Dienst der Akteursgruppe aus Buchungssystem und Preisberechnungssystem dar. Wir komponieren diesen Dienst schließlich noch sequentiell mit der Angebotserstellung:

ha	
$Inq(\langle Orig\rangle, \langle Dest\rangle, \langle RDate\rangle)$	$Offer(\langle Date\rangle, \langle AmtDue\rangle):\langle Date\rangle$ ist der Zeitpunkt der nächsten Flugverbindung und $\langle AmtDue\rangle$ der Preis für die angefragte Strecke in *Inq*, die nach Maßgabe der Regeln ha_1, ha_2 und ha_3 ermittelt wurden

Die Regel *ha* beschreibt somit den Dienst, den Holiday Airways gegenüber einem Kunden im Rahmen der Anfragebearbeitung erbringt. Wir haben damit aus einer strukturierten Beschreibung einzelner Prozessschritte durch deren systematische Komposition eine Gesamtperspektive auf den Dienst der Fluggesellschaft abgeleitet. Das gewählte Beispiel haben wir im Sinne der Anschaulichkeit möglichst kurz gehalten und konnten somit in einer vergleichsweise einfachen Kompositionsoperation bereits den Dienst herleiten. Die einzelnen Prozessschritte können aber problemlos weiter detailliert und ergänzt werden, so dass auch eine komplexe Prozessbeschreibung in einem mehrstufigem Vorgehen anhand der eingeführten Beschreibungselemente durchgeführt werden kann.

Die Diskussion des Beispiels hat gezeigt, dass die Erkenntnisse des formalen Modells auch in der informellen Darstellungsweise angewendet werden können, solange die Modellierung auf der informellen Ebene in einer Weise erfolgt, die vereinbar ist mit den Prinzipien des formalen Modells. Wir haben dies in der Definition der tabellarischen Notation dadurch gewährleistet, dass wir Übersetzungen der einzelnen Beschreibungselemente in die formale Notation angegeben und somit einen direkten Bezug zwischen der tabellarischen und der mathematischen Sichtweise auf Prozessregeln hergestellt haben.

6 Ausblick

In diesem Kapitel werden wir die Arbeit nochmals gesamthaft betrachten und einige abschließende Beobachtungen festhalten. Zum einen bewerten wir die Arbeit im Hinblick auf die Forschungsbeiträge, die aus ihr gewonnen werden können. Zum anderen diskutieren wir auch Anknüpfungspunkte für weitere Forschung in den Bereichen, die wir in den letzten Kapiteln nicht oder nur punktuell adressieren konnten. Wir schließen das Kapitel ab mit einem kurzen Fazit zum Einsatz dienstorientierter Architekturen.

6.1 Zusammenfassung des Beitrags

Unser Forschungsansatz motiviert sich aus der Überzeugung heraus, dass die SOA-Zielsetzung einer verbesserten Ausrichtung von Fachlichkeit und Implementierung mittels einer modularen akteurbezogenen Strukturierung von Funktionalitäten nur erreicht werden kann, wenn neben dem bereits vorhandenen technischen Dienstverständnis auch auf fachlicher Ebene Dienste präzise erfasst und abgegrenzt werden können.

Dazu haben wir in den letzten vier Kapiteln eine formale konzeptuelle Perspektive auf dienstorientierte Architekturen mit fachlichem Schwerpunkt erarbeitet. Wir haben in systematischer Weise die verschiedenen Konzepte, die zur fachlichen Beschreibung einer SOA erforderlich sind, hergeleitet und zueinander in Beziehung gestellt. In unserer Diskussion ist dabei von besonderer Bedeutung das Konzept des Prozesses, den wir als Grundbegriff für die Beschreibung geschäftlicher Handlungen ansehen und auf dessen Basis wir unsere Dienstauslegung entwickeln. Unser Modell ermöglicht eine exakte fachliche Beschreibung dienstorientierter Prozessarchitekturen in Organisationen und leistet auf diese Weise einen Beitrag zur Erreichung der im SOA-Ansatz angestrebten Harmonisierung der fachlichen und technischen Architekturperspektive.

In Kapitel 2 haben wir uns zunächst mit den Grundlagen zu Unternehmensarchitekturen befasst und das aktuelle Verständnis dienstorientierter Architekturen abgeleitet. In Kapitel 3 nehmen wir eine Begriffsbildung vor, indem wir alle für die Beschreibung einer SOA relevanten Konzepte anhand eines Praxisbeispiels in intuitiver Weise einführen und zueinander in Bezug stellen. In Kapitel 4 entwickeln wir in einem mehrstufigen Vorgehen ein formales mathematisches Modell, um die identifizierten Konzepte im SOA-Kontext präzise beschreiben zu können. In Kapitel 5 nehmen wir schließlich noch eine Einordnung unseres Ansatzes mit anderen Modellen vor und geben einen Ausblick, wie unser Modell auch in informeller Weise effektiv eingesetzt werden kann.

Aus der Diskussion in diesen Kapiteln folgen im Wesentlichen die folgenden fünf Beiträge:

- *Erarbeitung eines systematischen Überblicks zu bestehenden SOA-Definitionen mit deren Zielsetzungen und Prinzipien*
  Zur Beurteilung bestehender SOA-Ansätze haben wir zunächst vier zentrale SOA-Zielsetzungen identifiziert: Effizienzgewinne, die Steigerung von Qualität, die Erhöhung der Agilität sowie die fachliche Orientierung. Diese werden erreicht durch eine optimierte Ausrichtung von Softwareimplementierungen an der Fachlichkeit anhand von modularen und komponierbaren Diensten. Wir haben festgestellt, dass bestehende Ansätze in technischer Hinsicht diese Zielsetzung bereits umfassend adressieren. In fachlicher Hinsicht bestehen jedoch noch weitreichende Deutungsspielräume und Unschärfen, die sich erst im Übergang zur Implementierungssicht auflösen.

- *Aufbau eines formalen Modells zur Darstellung geschäftlicher Abläufe auf fachlicher Ebene*
  Zur Adressierung der Unschärfen in der fachlichen Beschreibung von Unternehmensarchitekturen haben wir ein aktionsorientiertes formales Beschreibungsmodell für geschäftliche Abläufe eingeführt. Wir haben das Modell in mehreren Stufen eingeführt, um eine möglichst hohe Flexibilität zu erzielen. Dabei lassen wir unterschiedliche Zeitmodelle zu, differenzieren zwischen der Betrachtung von Instanzen und den zugrundeliegenden Schemata und ermöglichen auch die Erfassungen von Transaktionen in Gesamtabläufen. Wir betrachten dabei den Prozess als Basiskonzept zur Erfassung von Abläufen und leiten alle weiteren Konzepte aus diesem ab.

- *Ableitung einer präzisen fachlichen Sicht auf Dienste und dienstorientierte Architekturen aus dem Prozessmodell*
  Auf Basis unseres formal gefassten Prozessmodells führen wir eine exakte fachliche Dienstdefinition ein. Wir beziehen Dienste auf Akteure und betrachten einen Dienst als die Beschreibung der Aktionen, die ein Akteur oder eine Gruppe von Akteuren zu einem Prozess beiträgt. Die Beschreibung erfolgt aus einer Außensicht und kann von zuvor erfolgten Aktionen, die für die diensterbringenden Akteure sichtbar waren, abhängen. Wir beschreiben zudem, wie durch Komposition mehrerer Dienste ein Prozess systematisch in einer dienstorientierten Struktur beschrieben und somit eine dienstorientiertes Architekturmodell aufgebaut werden kann.

- *Verknüpfung des formalen Modells mit dem intuitiven Begriffsverständnis*
  Formale Modelle sind häufig weniger intuitiv zu verstehen als informelle Darstellungen. Wir wirken diesem Effekt gezielt entgegen, indem wir unser formales Modell auf Grundlage einer ausführlichen Begriffsbildung aufbauen, in der wir praxisnah die zu modellierenden Konzepte vorstellen. Wir richten die Bezeichnungen der Konzepte dabei an dem in der Praxis etablierten Verständnis aus, um eine intuitive Erfassbarkeit zu gewährleisten. Zudem diskutieren wir jedes formal definierte Konzept auch im Kontext eines Praxisbeispiels, um die Anwendbarkeit des Modells zu demonstrieren.

- *Diskussion von informellen Beschreibungsansätzen*
  Zum Abschluss der Arbeit haben wir informelle Beschreibungsansätze untersucht, um die Anwendung unseres Modells in der Praxis zu vereinfachen. Dazu haben wir einerseits bestehende graphische Modellierungsansätze betrachtet und deren Einsetzbarkeit im

Kontext unseres Modells bewertet. Andererseits haben wir eine informelle Notation zur Erfassung von Prozessregeln eingeführt, die sich an der in der Umgangssprache verbreiteten Darstellung einer Regel als Wenn-Dann-Zusammenhang orientiert.

6.2 Anknüpfungspunkte für weitere Forschung

Mit unserem Modell zur fachlichen Darstellung von dienstorientierten Architekturen behandeln wir ein verhältnismäßig breites Anwendungsgebiet. Dies erfordert eine Fokussierung in der Betrachtung, um eine entsprechende Diskussionstiefe insbesondere auch im Hinblick auf unseren formalen Ansatz erreichen zu können. Entsprechend haben wir bestimmte Bereiche aus der Diskussion ausgenommen.

Wir haben in unserem Modell eine konzeptuelle Perspektive eingenommen und damit methodische Aspekte zur Beschreibung einer SOA nur ansatzweise behandelt. Weiterhin haben wir ausschließlich funktionale Aspekte von Prozessen und Diensten betrachtet und dabei von allen weiterführenden nicht funktionalen Eigenschaften abstrahiert. Daraus ergeben sich mehrere Anknüpfungspunkte für die weitere Forschung.

Entwicklung eines Vorgehensmodells In unserem Modell haben wir uns konzentriert auf eine präzise konzeptuelle Erfassung von dienstorientierten Architekturen und dazu die Elemente einer SOA sowie deren Beziehungen untersucht. Zum Abschluss der Architekturdiskussion in Abschnitt 4.5.3.3 haben wir in einem Ausblick dargestellt, dass in der Modellierung von SOA auch in methodischer Hinsicht eine Reihe interessanter Fragestellungen zu adressieren sind.

Zum einen können unterschiedliche Entwicklungsansätze für Dienste gewählt werden. Sie können top-down in einem analytischen Vorgehen aus einer übergreifenden unternehmerischen Aufgabe durch iterative Dekomposition abgeleitet werden. Es ist aber auch eine bottom-up Formulierung von Diensten auf Basis bestehender Prozessdefinitionen oder technischer Lösungen möglich. Schließlich sind auch hybride Ansätze denkbar, in denen Dienste sowohl aus geschäftliche Anforderungen als auch aus existierenden Artefakten heraus entwickelt werden. Weiterhin haben wir die Aspekte der Granularität und des Dienstschnitts angesprochen. Hier stellt sich die Frage, wie der Funktionsumfang von Diensten dahingehend bestimmt werden kann, dass eine größtmögliche Flexibilität bei einem zugleich möglichst geringen Definitions- und Wartungsaufwand der Dienste erreicht werden kann. Zuletzt ist auch die technische Kopplung von Interesse, bei der fachlich in unserem Modell definierte Dienste in eine bestehenden Implementierungstechnologie, wie zum Beispiel Web Services, übersetzt werden.

Entsprechend besteht ein möglicher nächster Schritt in unserer Forschung darin, ein Vorgehensmodell zu entwickeln, das aufbauend auf der konzeptuellen Perspektive die soeben diskutierten Fragen adressiert. Dabei sind neben einer qualitativen Einordnung möglicher Vorgehensmuster auch quantitative Metriken von Interesse, anhand der die verschiedenen zu treffenden Designentscheidungen – beispielsweise zur Granularität von Diensten – getroffen werden können.

Erweiterung des Konzeptmodells Für die Erfassung eines Dienstes sind neben der Verhaltensbeschreibung von Akteuren, wie wir sie in unserem Modell adressiert haben, auch nicht funktionale Eigenschaften von Bedeutung. So können für Dienste unter anderem Anforderungen bezüglich der Verfügbarkeit, Performanz, Zuverlässigkeit, Skalierbarkeit und Sicherheit gestellt werden. [PTDL08] Insbesondere in föderalen Umgebungen, in denen Prozesse von mehreren unabhängigen Akteuren erbracht werden, ist ein klares Verständnis bezüglich des zu erwartenden Leistungsniveaus der einzelnen Beteiligten in nicht funktionaler Hinsicht essentiell. So sind für die Auswahl passender Dienste neben der funktionalen Eignung die soeben beschriebenen nicht funktionalen Aspekte wichtige Kriterien.

Die Formulierung nicht funktionaler Eigenschaften von Diensten wird in der Literatur bereits umfassend adressiert. In [ADE+07] wird beispielsweise eine Erweiterung von Diensten zu sogenannten Rich Services vorgeschlagen. Dabei wird die rein funktionale Auslegung eines Dienstes, so wie wir sie in dieser Arbeit diskutiert haben, ergänzt um eine Infrastrukturebene, die unter anderem die Verschlüsselung, Protokollierung einer Dienstausführung, Formatkonvertierung sowie die Vorgabe von Richtlinien ermöglicht.

Entsprechend besteht ein Anknüpfungspunkt für unsere Forschung in einer Erweiterung unseres Modells, so dass nicht funktionale Aspekte beschrieben und in der Anwendung von Diensten berücksichtigt werden können.

Einordnung in bestehende Referenzmodelle In Kapitel 2 haben wir verschiedene in der Praxis etablierte Referenzmodelle für die Beschreibung von Unternehmensarchitekturen kennengelernt. In diesen Modellen wird die Dienstorientierung – wenn überhaupt – nur in informeller Art und Weise eingeführt. Wir haben uns in den letzten Kapiteln ausführlich mit der präzisen Einordnung und Abgrenzung von Diensten befasst. Insofern besteht ein weiterer nächster Schritt der Forschung darin, die Erkenntnisse aus unserer Diskussion wiederum in die Referenzmodelle einfließen zu lassen.

Im TOGAF Framework basiert beispielshalber die Architekturdefinition auf einer Analyse der Fähigkeiten in einem Unternehmen (Business Capabilities), um zu gewährleisten, dass die zu modellierenden geschäftlichen Abläufe durch die Akteure ausgeführt werden können. Ein solcher Bezug zwischen Akteuren und Prozessen lässt sich mit unserem konzeptuellen Modell präzise erfassen. Wir haben Dienste als Prozessregeln eingeführt, die den Beitrag von Akteuren zu einer Prozessausführung beschreiben. Insofern können wir die Fähigkeiten eines Akteurs als die Menge von Diensten betrachten, die dieser erbringen kann. Die Durchführbarkeit eines Prozesses ist folglich gegeben, wenn sich dieser als eine Komposition von Diensten der Akteure in der entsprechenden Organisation erfassen lässt.

Weiterhin können die in unserem konzeptuellen Modell gewonnen Erkenntnisse auch in bestehende Artefaktenmodelle übernommen werden, um eine präzise Abgrenzung der Artefakte zu erreichen. In der Diskussion der Artefaktenmodelle haben wir festgestellt, dass bestimmte Konzepte wie Geschäftsprozess, Geschäftsaufgabe und Aktivitäten nicht überschneidungsfrei definiert sind. Durch den Abgleich mit dem formalen Modell können bestehende Ambiguitäten adressiert und eine entsprechende Abgrenzung vorgenommen werden.

6.3 Fazit

Zum Abschluss ziehen wir ein Fazit zu den Erkenntnissen, die wir zu dienstorientierten Architekturen in dieser Arbeit gewonnen haben. Die Einführung einer SOA zielt auf eine verbesserte Ausrichtung der Anwendungslandschaft eines Unternehmens an dessen geschäftlicher Architektur ab. Damit verbunden ist zumeist eine klare Erwartung hinsichtlich einer Steigerung von Qualität, Agilität und Effizienz in der betrieblichen Informationsverarbeitung.

Die verbesserte Ausrichtung wird erreicht durch eine analoge Strukturierung der geschäftlichen und technischen Architektur anhand von feingranularen Bausteinen, den Diensten. Auf Grund der feingranularen Struktur kann eine Kongruenz zwischen den Ebenen auch bei häufigen Anpassungen einfacher aufrecht erhalten werden. Dies setzt jedoch ein präzises Verständnis von Diensten sowohl auf fachlicher als auch technischer Ebene voraus. In der Untersuchung bestehender SOA-Definitionen haben wir gesehen, dass in der fachlichen Auslegung nach wie vor konzeptuelle Unschärfen in der Definition von Diensten bestehen. Diese lösen sich erst auf der technischen Ebene in der Betrachtung konkreter Lösungsansätze, wie zum Beispiel Web Services, auf.

Wir haben mit unserem Modell eine präzise konzeptuelle Beschreibung von Diensten ermöglicht, indem wir diese systematisch auf Basis eines formalen Prozessmodells hergeleitet haben. Wir betrachten dabei Dienste als eine ergebnisorientierte Verhaltensbeschreibung eines Akteurs. Diese lässt sich formalisieren, indem wir die Aktionen, die ein Akteur ausführt, in Relation stellen zu vorangegangenen Aktionen, die durch den diensterbringenden Akteur beobachtbar waren. Durch Komposition können zudem umfassende Dienste aus einfacheren Diensten zusammengesetzt und so auch das Verhalten ganzer Akteursgruppen beschrieben werden. Das Dienstmodell ermöglicht somit eine präzise fachliche Modellierung dienstorientierter Architekturen und kann damit einen wichtigen Beitrag zur Realisierung der SOA-Vorteile leisten.

Allerdings ist eine formale Modellierung auch mit größeren Aufwänden verbunden, die die angestrebten Effizienzgewinne potenziell relativieren. Dies liegt vor allem begründet in der weniger intuitiven mathematischen Darstellung sowie dem hohen Detailgrad, der mit einem formalen Modell einhergeht. Insofern sind bei der Entwicklung einer SOA Kompromissentscheidungen hinsichtlich der Präzision der Dienstmodellierung erforderlich. Wir betrachten in dieser Hinsicht unser Konzeptmodell nicht nur als Zielpunkt, was die formale Erfassung einer Dienstarchitektur betrifft, sondern auch als Ausgangspunkt für die informelle Modellierung von Diensten. Unsere Diskussion graphischer und tabellarischer Beschreibungsansätze im letzten Kapitel hat gezeigt, dass die Erkenntnisse aus dem formalen Konzeptmodell auch in der informellen Darstellung dienstorientierter Architekturen anwendbar sind und somit auch jenseits der formalen Perspektive zu einer Präzisierung der fachlichen Auslegung von SOA beitragen können.

Index

Literaturverzeichnis

[ADE+07] ARROTT, Matthew ; DEMCHAK, Barry ; ERMAGAN, Vina ; FARCAS, Claudiu ; FARCAS, Emilia ; KRÜGER, Ingolf H. ; MENARINI, Massimiliano: Rich Services: The integration piece of the SOA puzzle. In: *IEEE International Conference on Web Services (ICWS)* IEEE, 2007, S. 176–183

[AS11] AALST, Wil van d. ; STAHL, Christian: *Modeling Business Processes: A Petri Net-Oriented Approach*. MIT Press, 2011

[Bal98] BALZERT, Heide: *Lehrbuch der Software-Technik*. Bd. 2. Spektrum Akademischer Verlag, 1998

[BB96] BRASSARD, Gilles ; BRATLEY, Paul: *Fundamentals of algorithmics*. Bd. 524. Prentice Hall New York, 1996

[BCHM+05] BASKERVILLE, Richard ; CAVALLARI, Marco ; HJORT-MADSEN, Kristian ; PRIES-HEJE, Jan ; SORRENTINO, Maddalena ; VIRILI, Francesco: Extensible architectures: the strategic value of service oriented architecture in banking. In: *European Conference on Information Systems Proceedings* (2005)

[BDMS11] BUCKL, Sabine ; DIERL, Thomas ; MATTHES, Florian ; SCHWEDA, Christian M.: Complementing The Open Group Architecture Framework with Best Practice Solution Building Blocks. In: *44th Hawaii International Conference on System Sciences (HICSS)* IEEE, 2011, S. 1–9

[BEL+07] BUCKL, Sabine ; ERNST, Alexander M. ; LANKES, Josef ; SCHNEIDER, Kathrin ; SCHWEDA, Christian M.: A pattern based approach for constructing enterprise architecture management information models. In: *Wirtschaftsinformatik* 2 (2007), S. 145–162

[BKM07] BROY, Manfred ; KRÜGER, Ingolf ; MEISINGER, Michael: A formal model of services. In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 16 (2007), Nr. 1, S. 5

[BN84] BIRRELL, Andrew D. ; NELSON, Bruce J.: Implementing remote procedure calls. In: *ACM Transactions on Computer Systems (TOCS)* 2 (1984), Nr. 1, S. 39–59

[BO01] BROY, Manfred ; OLDEROG, Ernst-Rüdiger: Trace-oriented models of concurrency. In: *Handbook of process algebra* (2001), S. 101–195

[Bro98] BROY, Manfred: *Informatik: Eine grundlegende Einführung, Teil 2 Systemstrukturen und Theoretische Informatik*. 2. Auflage. Springer Verlag Berlin, 1998

[Bro10] BROY, Manfred: Relating time and causality in interactive distributed systems. In: *European Review* 18 (2010), Nr. 04, S. 507–563

[BS99] BROY, Manfred (Hrsg.) ; SPANIOL, Otto (Hrsg.): *Informatik und Kommunikationstechnik: VDI Lexikon*. Springer, 1999

[BS01] BROY, Manfred ; STØLEN, Ketil: *Specification and development of interactive systems: focus on streams, interfaces, and refinement*. Springer Verlag, 2001

[BWM07] BRUNING, Stefan ; WEISSLEDER, Stephan ; MALEK, Miroslaw: A fault taxonomy for service-oriented architecture. In: *High Assurance Systems Engineering Symposium (HASE)* IEEE, 2007, S. 367–368

[CC88] CARLIER, Jacques ; CHRÉTIENNE, Philippe: Timed Petri net schedules. In: *Advances in Petri Nets 1988* (1988), S. 62–84

[CDM+04] CABRAL, Liliana ; DOMINGUE, John ; MOTTA, Enrico ; PAYNE, Terry ; HAKIMPOUR, Farshad: Approaches to semantic web services: an overview and comparisons. In: *The semantic web: Research and applications*. Springer, 2004, S. 225–239

[CK07] CHANG, Soo ; KIM, Soo: A Systematic Approach to Service-Oriented Analysis and Design. In: *Product-Focused Software Process Improvement* (2007), S. 374–388

[CLRS01] CORMEN, Thomas H. ; LEISERSON, Charles E. ; RIVEST, Ronald L. ; STEIN, Clifford: *Introduction to Algorithms*. MIT Press, 2001

[Coc01] COCKBURN, Alistair: *Writing effective use cases*. Bd. 1. Addison-Wesley Reading, MA, 2001

[CS94] CHEN, Rong ; SCHEER, August-Wilhelm: *Modellierung von Prozessketten mittels Petri-Netz-Theorie*. Institut für Wirtschaftsinformatik, Saarland, 1994 (107)

[DP02] DAVEY, Brian A. ; PRIESTLEY, Hilary A.: *Introduction to lattices and order*. Cambridge University Press, 2002

[EHH+08] ENGELS, Gregor ; HESS, Andreas ; HUMM, Bernhard ; JUWIG, Oliver ; LOHMANN, Marc ; RICHTER, Jan-Peter ; VOSS, Markus ; WILLKOMM, Johannes: *Quasar Enterprise: Anwendungslandschaften serviceorientiert gestalten*. Dpunkt-Verlag, 2008

[Erl08] ERL, Thomas: *SOA - Entwurfsprinzipien für serviceorientierte Architektur*. Addison-Wesley Reading, MA, 2008

[EW01] ESHUIS, Rik ; WIERINGA, Roel: A formal semantics for UML Activity Diagrams-Formalising workflow models / University of Twente, Centre for Telematics and Information Technology. 2001. – Forschungsbericht

[FR07] FRÖSCHLE, Hans-Peter ; REINHEIMER, Stefan: *Serviceorientierte Architekturen*. Dpunkt-Verlag, 2007

[GGW98] GEHRKE, Thomas ; GOLTZ, Ursula ; WEHRHEIM, Heike: The Dynamic Models of UML: Towards a Semantics and its Application in the Development Process / Universität Hildesheim, Institut für Informatik. 1998. – Forschungsbericht

[Gla90] GLABBEEK, Rob J v.: The linear time-branching time spectrum. In: *Proceedings of the Theories of Concurrency: Unification and Extension* Springer-Verlag, 1990, S. 278–297

[GMMP91] GHEZZI, Carlo ; MANDRIOLI, Dino ; MORASCA, Sandro ; PEZZÈ, Mauro: A unified high-level Petri net formalism for time-critical systems. In: *IEEE Transactions on Software Engineering* 17 (1991), Nr. 2, S. 160–172

[GP09] GLABBEEK, Rob J v. ; PLOTKIN, Gordon D.: Configuration structures, event structures and Petri nets. In: *Theoretical Computer Science* 410 (2009), Nr. 41, S. 4111–4159

[Haa99] HAAR, Stefan: On Occurrence Net Semantics of Petri Nets / INRIA. 1999 (3718). – Forschungsbericht

[HN01] HANSEN, Hans R. ; NEUMANN, Gustaf: *Wirtschaftsinformatik 1. Grundlagen und Anwendungen.* Lucius & Lucius, 2001

[Hoa85] HOARE, Charles Antony R.: *Communicating Sequential Processes*. Prentice Hall, 1985

[HP04] HOLMQVIST, Magnus ; PESSI, Kalevi: Process Integration and Web Services: A Case of Evolutional Development in a Supply Chain. In: *Scandinavian Journal of Information Systems* 16 (2004), Nr. 1, S. 2

[Hum00] HUME, David ; NORTON, David F. (Hrsg.) ; NORTON, Mary (Hrsg.): *A Treatise of Human Nature*. Oxford: Clarendon Press, 2000

[IBM12] IBM: *Service Oriented Architecture*. http://www-01.ibm.com/software/solutions/soa/, 7 2012. – Heruntergeladen am 18. Juli 2012

[IEE11] IEEE: Systems and software engineering – Architecture description. In: *ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000)* (2011), 1, S. 1 –46

[KA07] KOHLMANN, Falk ; ALT, Rainer: Business-Driven Service Modeling-A Methodological Approach from the Finance Industry. In: *Business Process and Services Computing (BPSC)* 7 (2007), S. 180–193

[KBS07] KRAFZIG, Dirk ; BANKE, Karl ; SLAMA, Dirk: *Enterprise SOA. Best Practices für Serviceorientierte Architekturen – Einführung, Umsetzung*. Praxis, Heidelberg (mitp-Verlag), 2007. – 347 S.

[KCM06] KITCHIN, David ; COOK, William ; MISRA, Jayadev: A language for task orchestration and its semantic properties. In: *CONCUR 2006–Concurrency Theory* (2006), S. 477–491

[KD07] KIM, Yukyong ; DOH, Kyung-Goo: The Service Modeling Process Based on Use Case Refactoring. In: *Business Information Systems* (2007), S. 108–120

[KDK07] KUMAR, Sanjeev ; DAKSHINAMOORTHY, Vijay ; KRISHNAN, MS: Does SOA improve the supply chain? An empirical analysis of the impact of SOA adoption on electronic supply chain performance. In: *40th Annual Hawaii International Conference on System Sciences (HICSS)* IEEE, 2007, S. 171b

[Kle52] KLEENE, Stephen C.: *Introduction to Metamathematics*. North-Holland Publishing Co. Amsterdam, P. Noordhoff N.V. Groningen, 1952

[KSN92] KELLER, Gerhard ; SCHEER, August-Wilhelm ; NÜTTGENS, Markus: *Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK)*. Institut für Wirtschaftsinformatik, Saarland, 1992

[KSR04] KAABI, Rim S. ; SOUVEYET, Carine ; ROLLAND, Colette: Eliciting service composition in a goal driven manner. In: *Proceedings of the 2nd international conference on Service oriented computing* ACM, 2004, S. 308–315

[LLJ+05] LAWLER, James ; LI, Zheng ; JAVED, Nasir ; ANDERSON, Dennis ; HILL, Jonathan ; HOWELL-BARBER, Hortense: A study of web services projects in the financial services industry. In: *Information Systems Management* 22 (2005), Nr. 1, S. 66–76

[LSS10] LEUXNER, Christian ; SITOU, Wassiou ; SPANFELNER, Bernd: A formal model for work flows. In: *8th IEEE International Conference on Software Engineering and Formal Methods (SEFM)* IEEE, 2010, S. 135–144

[Mat08] MATTHES, Florian: Softwarekartographie. In: *Informatik-Spektrum* 31 (2008), Nr. 6, S. 527–536

[Mel08] MELZER, Ingo: *Service-orientierte Architekturen mit Web Services*. 3. Spektrum Akademischer Verlag, 2008

[MF11] MÉNDEZ FERNÁNDEZ, Daniel: *Requirements Engineering: Artefact-Based Customisation*, Technische Universität München, Diss., 2011

[Mic12] MICROSOFT: *Distributed Component Object Model (DCOM) Remote Protocol Specification*. 3 2012

[Mil80] MILNER, Robin: *A Calculus of Communication Systems*. Lecture Notes in Computer Science 92, 1980

[MVAR07] MÜLLER, Benjamin ; VIERING, Goetz ; AHLEMANN, Frederik ; RIEMPP, Gerold: Towards Understanding the Sources of the Economic Potential of Service-Oriented Architecture: Findings from the automotive and banking industry. In: *European Conference on Information Systems*, 2007

[NL05] NEWCOMER, Eric ; LOMOW, Greg: *Understanding SOA with web services*. Addison-Wesley Professional, 2005

[NR02] NÜTTGENS, Markus ; RUMP, Frank J.: Syntax und Semantik ereignisgesteuerter Prozessketten (EPK). In: *Promise* Bd. 2, 2002, S. 64–77

[OAS04] OASIS: *UDDI Version 3.0.2*. http://uddi.org/pubs/uddi-v3.0.2-20041019.pdf, 10 2004. – Heruntergeladen am 20. Juli 2012

[OAS06] OASIS: *Reference Model for Service Oriented Architecture 1.0*. http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf, 12 2006. – Heruntergeladen am 18. Juli 2012

[Obj05a] OBJECT MANAGEMENT GROUP: *Unified Modeling Language: Infrastructure, Version 2.0*. http://www.omg.org/spec/UML/2.0/Infrastructure/PDF/, 07 2005. – Heruntergeladen am 6. März 2013

[Obj05b] OBJECT MANAGEMENT GROUP: *Unified Modeling Language: Superstructure, Version 2.0*. http://www.omg.org/spec/UML/2.0/Superstructure/PDF/, 08 2005. – Heruntergeladen am 20. Februar 2013

[Obj11] OBJECT MANAGEMENT GROUP: *Business Process Model And Notation: Version 2.0*. http://www.omg.org/spec/BPMN/2.0/PDF, 01 2011. – Heruntergeladen am 4. März 2013

[Off08] OFFERMANN, Philipp: SOAM – Eine Methode zur Konzeption betrieblicher Software mit einer Serviceorientierten Architektur. In: *Wirtschaftsinformatik* 50 (2008), S. 461–471. – ISSN 0937–6429

[Ope09] OPENGROUP: *The Open Group Architecture Framework (TOGAF)*. 9. 2009

[Ope11] OPENGROUP: *Common Object Request Broker Architecture (CORBA)*. http://www.omg.org/spec/CORBA/, 11 2011. – Heruntergeladen am 10. Juli 2012

[Ope12a] OPENGROUP: *ArchiMate 2.0*. https://www2.opengroup.org/ogsys/catalog/c118, 1 2012. – Heruntergeladen am 5. März 2012

[Ope12b] OPENGROUP: *Service-Oriented Architecture (SOA)*. https://collaboration.opengroup.org/projects/soa/pages.php?action=show&ggid=1574, 7 2012. – Heruntergeladen am 18. Juli 2012

[PTDL08] PAPAZOGLOU, Michael P. ; TRAVERSO, Paolo ; DUSTDAR, Schahram ; LEYMANN, Frank: Service-oriented computing: A research roadmap. In: *International Journal of Cooperative Information Systems* 17 (2008), Nr. 2, S. 223–255

[PVDH07] PAPAZOGLOU, Michael P. ; VAN DEN HEUVEL, Willem-Jan: Service oriented architectures: approaches, technologies and research issues. In: *The VLDB Journal* 16 (2007), Nr. 3, S. 389–415

[QDS04] QUARTEL, Dick ; DIJKMAN, Remco ; SINDEREN, Marten van: Methodological support for service-oriented design with ISDL. In: *Proceedings of the 2nd international conference on Service oriented computing (ICSOC)*. New York, NY, USA : ACM, 2004, S. 1–10

[Rei91] REISIG, Wolfgang: Petri nets and algebraic specifications. In: *Theoretical Computer Science* 80 (1991), Nr. 1, S. 1 – 34

[RHB98] ROSCOE, Andrew W. ; HOARE, Charles A. ; BIRD, Richard: *The theory and practice of concurrency*. Bd. 1. Prentice Hall Engelwood Cliffs, NJ, 1998

[RLPB07] REINHEIMER, Stefan ; LANG, Florian ; PURUCKER, Jörg ; BRÜGMANN, Hinnerk: 10 Antworten zu SOA. In: *HMD-Praxis der Wirtschaftsinformatik* 44 (2007), Nr. 253, S. 7–17

[Sch92] SCHEER, August-Wilhelm: *Architektur integrierter Informationssysteme: Grundlagen der Unternehmensmodellierung*. Springer, 1992

[Sch01] SCHEER, August-Wilhelm: *ARIS-Modellierungsmethoden, Metamodelle, Anwendungen*. Springer, 2001

[Smi09] SMITH, Roger: *InformationWeek Analytics: State Of SOA*. http://www.informationweek.com/news/214501922, Feb 2009. – Heruntergeladen am 8. Juni 2012

[SSP09] SINDHGATTA, Renuka ; SENGUPTA, Bikram ; PONNALAGU, Karthikeyan: Measuring the Quality of Service Oriented Design. In: *Service-Oriented Computing* (2009), S. 485–499

[Tar55] TARSKI, Alfred: A lattice-theoretical fixpoint theorem and its applications. In: *Pacific journal of Mathematics* 5 (1955), Nr. 2, S. 285–309

[Thu04] THURNER, Veronika: *Formal fundierte Modellierung von Geschäftsprozessen*. Logos-Verlag, Berlin, 2004

[Thu09] THURLOW, Robert: *RPC: Remote Procedure Call Protocol Specification Version 2*. http://tools.ietf.org/html/rfc5531, 5 2009

[VM07] VIERING, Goetz ; MÜLLER, Benjamin: *Economic Potential of Service-Oriented Architecture*. Josef Eul Verlag, 2007

[Wat96] WATT, David A.: *Programming language syntax and semantics*. Prentice Hall PTR, 1996

[WBF+09] WINTER, Robert ; BROCKE, Jan vom ; FETTKE, Peter ; LOOS, Peter ; JUNGINGER, Stefan ; MOSER, Christoph ; KELLER, Wolfgang ; MATTHES, Florian ; ERNST, Alexander: Patterns in der Wirtschaftsinformatik. In: *Wirtschaftsinformatik* 51 (2009), Nr. 6, S. 535–542

[WBMS10] WINTER, Katharina ; BUCKL, Sabine ; MATTHES, Florian ; SCHWEDA, Christian M.: Investigating the state-of-the-art in enterprise architecture management method in literature and practice. In: *Proceedings of the 5th mediterranean conference on information systems (MCIS)*, 2010

[WF07] WINTER, Robert ; FISCHER, Ronny: Essential layers, artifacts, and dependencies of enterprise architecture. In: *Journal of Enterprise Architecture* 3 (2007), Nr. 2, S. 7–18

[WG08] WONG, Peter ; GIBBONS, Jeremy: A process semantics for BPMN. In: *Formal Methods and Software Engineering* (2008), S. 355–374

[Zac87] ZACHMAN, John A.: A framework for information systems architecture. In: *IBM systems journal* 26 (1987), Nr. 3, S. 276–292

[Zac96] ZACHMAN, John A.: Enterprise architecture: The issue of the century. In: *Database Programming and Design Magazine* (1996)

[Zac05] ZACHARIAS, Roger: Serviceorientierung: Der OO-König ist tot, es lebe der SOA-König. In: *OBJEKTspektrum* 12 (2005), Nr. 2, S. 43–52

[Zac08] ZACHMAN, John A.: *About The Zachman Framework*. http://www.zachman.com/about-the-zachman-framework, 1 2008. – Heruntergeladen am 22. Juni 2012

[ZLY05] ZHANG, Zhuopeng ; LIU, Ruimin ; YANG, Hongji: Service identification and packaging in service oriented reengineering. In: *Proceedings of the 17th international conference on software engineering and knowledge engineering, Taipei, Taiwan*, 2005, S. 14–16