Sichten auf Geschäftsprozesse mit besonderer Betrachtung von Compliance

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

Vorgelegt von

David Schumm

aus Heilbronn (Deutschland)

Hauptberichter:	Prof. Dr. Frank Leymann
Mitberichter:	Prof. Dr. Schahram Dustdar
Tag der mündlichen Prüfung:	09.02.2015

Institut für Architektur von Anwendungssystemen
der Universität Stuttgart

2015

Inhaltsverzeichnis

Inhaltsverzeichnis III

Abkürzungsverzeichnis IX

Zusammenfassung XI

Abstract XIII

Teil I Grundlagen 15

1 Einleitung 17

1.1 Problembereich und Motivation 17

1.2 Forschungsschwerpunkte und Beiträge 19

1.3 Aufbau der Arbeit 24

2 Hintergrund und verwandte Arbeiten 25

2.1 Grundlagen zu Geschäftsprozessen 25

2.1.1 Beschreibung und Automatisierung von Geschäftsprozessen 25

2.1.2 Geschäftsprozesstechnologie 27

2.1.3 Lebenszyklus von Geschäftsprozessen 30

2.1.4 Ein graphbasiertes Metamodell für Prozessskizzen 31

2.1.5 Komplexität von Geschäftsprozessen 34

2.2 Stand der Technik von Sichten 37

2.2.1 Sichten in der Softwareentwicklung 37

2.2.2 Sichten auf Geschäftsprozesse 41

2.2.3 Sprachübergreifende Sichten auf Geschäftsprozesse 45

2.3 Stand der Technik von Compliance 47

2.3.1 Abgrenzung und Terminologie 47

2.3.2 Technische Beschreibung von Compliance-Anforderungen 49

2.3.3 Prüfung von Geschäftsprozessen auf Regelkonformität 50

2.3.4 Wiederverwendung im Bereich Geschäftsprozesse 52
2.4 Anforderungsanalyse und Positionierung 55
2.4.1 Geschäftsprozesse in der Praxis 56
2.4.2 Sichten auf Geschäftsprozesse 56
2.4.3 Compliance von Geschäftsprozessen 57
2.4.4 Querbezüge von Sichten und Compliance 58
2.4.5 Zusammenfassung der Zielstellung 59

Teil II Sichten auf Geschäftsprozesse 61

3 Entwurf von Sichten auf Geschäftsprozesse 63
3.1 Motivation des musterbasierten Sichtenentwurfs 63
3.2 Konzeption des musterbasierten Sichtenentwurfs 65
3.3 Beschreibungsformen im musterbasierten Sichtenentwurf 70
3.3.1 Beschreibungsform für Sichtenentwurfsmuster 70
3.3.2 Beschreibungsform für Sichtenentwürfe 72
3.4 Musterkatalog für den Sichtenentwurf 72
3.4.1 Muster zur Strukturveränderung 75
3.4.2 Muster zur Informationsanreicherung 88
3.4.3 Muster zur Darstellungsanpassung 92
3.4.4 Muster von Vorgaben zur Durchführung von Transformationen 105
3.4.5 Musterbeziehungen 110
3.4.6 Vollständigkeit des Musterkatalogs 115
3.5 Klassifikation von Sichtenentwürfen 115
3.5.1 Merkmale von Modelltransformationen 115
3.5.2 Merkmalsausprägungen 116
3.5.3 Klassifikation 117
3.6 Musterbasierter Sichtenentwurf 119
3.6.1 Richtlinien zum Sichtenentwurf 119

3.6.2 Vom Grobentwurf zum Feinentwurf 121
3.6.3 Komplexitätsreduzierende Konzepte im Sichtenentwurf 126
3.7 Informationsdesign im Lebenszyklus von Geschäftsprozessen 128
4 Beispielhafte Sichtenentwürfe für den Prozesslebenszyklus 131
4.1 Vielseitig verwendbare Sichten 131
4.1.1 Entfernen von Details 131
4.1.2 Zusammenfassen bekannter Sachverhalte 133
4.1.3 Hervorhebung bestimmter Aspekte 135
4.1.4 Informationsschutz 137
4.2 Sichten für die Prozessmodellierung 138
4.2.1 Variierende Granularität 138
4.2.2 Ausführungskonfiguration 141
4.3 Sichten für das Monitoring und die Analyse von Prozessen 142
4.3.1 Vereinfachte Prozessinstanz 142
4.3.2 Einfache Zeitanalyse 143
4.4 Übersicht der Verwendung der Sichtenentwurfsmuster 144
5 Sprachübergreifende Sichten 146
5.1 Konzeption des sprachübergreifenden Sichtenentwurfs 147
5.1.1 Terminologie sprachübergreifender Sichten 147
5.1.2 Hilfselemente für den sprachübergreifenden Sichtenentwurf 148
5.1.3 Rahmenbedingungen sprachübergreifender Sichten 150
5.2 Prozessübersetzende Sichten 151
5.2.1 Prozess vereinfachen und anschließend übersetzen 151
5.2.2 Sichten auf übersetzte Prozesse 152
5.3 Sprachemulierende Sichten 153
5.3.1 Ein Metamodell für Chevron-Diagramme 154
5.3.2 Sprachemulation von Prozessmodellen durch Chevron 157

5.3.3 Sprachemulation von Prozessinstanzen durch Chevron 160
5.4 Instanzprojizierende Sichten 162
5.4.1 Prinzip der Instanzprojektion 162
5.4.2 Muster zur Zustandsübertragung 165

Teil III Compliance von Geschäftsprozessen 169

6 Management von Compliance in Geschäftsprozessen 171
6.1 Ganzheitliche Herangehensweise 171
6.2 Erweiterungen des Lebenszyklus von Geschäftsprozessen 174
6.3 Konzeption von rr-Fragmenten 176
6.4 Beispiele von rr-Fragmenten 179
6.4.1 Revisionssichere Dokumentation 180
6.4.2 Einbindung eines Compliance-Beauftragten 180
7 Sichtenentwürfe für Compliance 182
7.1 Sichten für rr-Fragmente 182
7.1.1 Ausschneiden eines rr-Fragments 182
7.1.2 Ausblenden von rr-Fragmenten 184
7.1.3 Hervorheben von rr-Fragmenten 185
7.2 Sichten für Compliance-Prüfungen 187
7.2.1 Nachvollziehbarkeit einer Sicht 187
7.2.2 Analyse von Verantwortlichkeiten 189
7.3 Übersicht der Verwendung der Sichtenentwurfsmuster 191
8 Integrative Techniken 193
8.1 Integration von rr-Fragmenten in einen Geschäftsprozess 193
8.1.1 Vorgehensweise 194
8.1.2 Beispiel 196
8.1.3 Diskussion 198

8.2 Prüfung der Integration durch Verifikationstechniken 198
8.2.1 Vorgehensweise 198
8.2.2 Beispiel 202
8.2.3 Diskussion 205

Teil IV Technische Realisierbarkeit 206

9 Abbildung wesentlicher Konzepte auf die Prozesssprache BPEL 208
9.1 Dienstbasierte Realisierung von Sichtenentwürfen 208
9.1.1 Bereitstellung von Transformationsfunktionen als Dienst 209
9.1.2 Beispielhafte Transformationsdienste 211
9.1.3 Von der Mustersequenz zur Dienstkomposition 215
9.2 Zustandsprojektion von BPEL zu Chevron 218
9.2.1 Konzeptionelles Modell zur Spezifikation einer Zustandsprojektion 219
9.2.2 Vorgehensweise zur Durchführung einer Zustandsprojektion 220
9.2.3 Spezifikation von Zustandsübertragungsregeln 221
9.2.4 Realisierung von Chevron mit XML und SVG 223
9.3 Erweiterungen von BPEL für rr-Fragmente 224
10 Architektur und Implementierung einer unterstützenden Infrastruktur 228
10.1 Architekturüberblick 229
10.2 Dienstkomposition zur Realisierung von Sichten 230
10.2.1 Architektur 230
10.2.2 Implementierung 231
10.3 Bibliothek für Prozessfragmente 233
10.3.1 Architektur 233
10.3.2 Implementierung 235
10.4 Modellierungsumgebung für Prozesse 236
10.4.1 Architektur 236

10.4.2 Implementierung 238
10.5 Monitoring-Plattform 242
10.5.1 Architektur 242
10.5.2 Implementierung 244

Teil V Abschluss 247

11 Validierung 249
11.1 Validierung des musterbasierten Sichtenentwurfs 249
11.2 Validierung des Konzepts der rr-Fragmente 255
12 Ergebnisse 257
12.1 Zusammenfassung 257
12.2 Erkenntnisse 258
13 Ausblick 260
14 Anhang 262
14.1 Literaturverzeichnis 262
14.2 Kompaktdarstellung des Musterkatalogs 285
14.3 Hilfselemente 288
14.4 Vorgehensweise zur Erläuterung des musterbasierten Sichtenentwurfs 289
14.5 Abbildungsverzeichnis 291
14.6 Listingverzeichnis 302
14.7 Tabellenverzeichnis 303
Danksagung 304
Ehrenwörtliche Erklärung 305

Abkürzungsverzeichnis

Abkürzung	Bezeichnung
ACID	Atomicity, Consistency, Isolation, Durability
ADL	Architecture Description Language
AJAX	Asynchronous JavaScript and XML
AOP	Aspektorientierte Programmierung
API	Application Programming Interface
ARIS	Architektur integrierter Informationssysteme
BAM	Business Activity Monitoring
BoD	Binding of Duties
BPEL	Business Process Execution Language
BPM	Business Process Management
BPMN	Business Process Model and Notation
BPMS	Business Process Management System
BPSL	Business Property Specification Language
CEP	Complex Event Processing
CLA	Compliance Level Agreement
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
CTL	Computation Tree Logic
DBMS	Database Management System
DSL	Domain-Specific Language
EA	Enterprise Architecture
EAI	Enterprise Application Integration
ECA	Event Condition Action
EPK	Ereignisgesteuerte Prozesskette
GPL	General Purpose Language
GUI	Graphical User Interface
ISO	International Organization for Standardization
JSF	JavaServer Faces
JSP	JavaServer Pages
KCI	Key Compliance Indicator
KPI	Key Performance Indicator

LTL	Lineare Temporale Logik
MDSD	Model-driven Software Development
MEP	Message Exchange Pattern
MoM	Message-oriented Middleware
OASIS	Organization for the Advancement of Structured Information Standards
OMG	Object Management Group
PEP	Parametrierbare EAI Patterns
PIP	Partner Interface Process
QoS	Quality of Service
RDBMS	Relational Database Management System
rr-Fragment	regelrealisierendes Prozessfragment
SESE	Single-Entry-Single-Exit
SLA	Service Level Agreement
SOA	Serviceorientierte Architektur
SoD	Segregation of Duties
SQL	Structured Query Language
SVG	Scalable Vector Graphics
UE	Usability Engineering
UML	Unified Modeling Language
UML AD	Unified Modeling Language Activity Diagram
UUID	Universally Unique Identifier
WAP	Workflow Activity Pattern
WAR	Web Archive
WfMS	Workflow Management System
WSDL	Web Services Description Language
W3C	World Wide Web Consortium
YAWL	Yet Another Workflow Language

Zusammenfassung

Steigende Komplexität ist eine Begleiterscheinung der industrialisierten und globalisierten Welt, die alle Unternehmensbereiche vor große Herausforderungen stellt: Eine drastisch zunehmende Menge heterogener Anforderungen aus verschiedenen Quellen wirkt auf vielgestaltige Art in die ohnehin komplexe Unternehmenslandschaft und auf das Innenleben der Unternehmen ein. Geschäftsprozesse, als Abbild oder auch als Vorbild für geplantes Handeln, sollen die an sie gestellten Anforderungen und Regeln auf nachvollziehbare Weise erfüllen. Zugleich sollen sie die Wirtschaftlichkeit erhalten und die Wettbewerbsfähigkeit durch mehr Effizienz stärken. Geschäftsprozesse sind in diesem Spannungsfeld einer steigenden Komplexität unterworfen. Wenn man der steigenden Komplexität von Geschäftsprozessen nicht mit komplexitätsreduzierenden Techniken begegnet und sie dadurch besser verstehen und schließlich wieder steuern kann, werden Zusammenhänge möglicherweise nur unzureichend erfasst und womöglich falsche Entscheidungen getroffen.

Sichten auf Prozesse wurden im vergangenen Jahrzehnt von verschiedenen Forschergruppen als Graphtransformationstechnik zur vereinfachenden Darstellung von Prozessen untersucht. Sichtenfunktionen finden sich jedoch nur vereinzelt in Werkzeugen wieder, und es besteht keine gemeinsame Basis an Begrifflichkeiten und zudem kein Konsens, welche Funktionen über das Auslassen und Aggregieren von Modellelementen hinaus zu Prozesssichten gehören. Durch den Mangel an gemeinsamer konzeptioneller Basis stehen diese praxisrelevanten Funktionen nur einem eingeschränkten Nutzerspektrum zur Verfügung.

Mit einem *musterbasierten Ansatz* zum Entwurf von Sichten auf Prozesse wird eine abstrakte Ausgangsbasis geschaffen, die die verschiedenen bestehenden Möglichkeiten von Sichten auf einer einheitlichen konzeptionellen Plattform zugänglich macht und weitgehend unabhängig von Technologien, Werkzeugen und Prozesssprachen bereitstellt. Komplexe Transformationsfunktionen werden abstrahiert und in einer einheitlichen Beschreibungsform als Sichtenentwurfsmuster in einem Musterkatalog bereitgestellt. Darauf aufbauend wird der Stand der Technik um den musterbasierten Sichtenentwurf sowie um den Entwurf sprachübergreifender Sichten erweitert. Der Entwurf sprachübergreifender Sichten macht von Übersetzungstechniken, der Darstellungsanpassung und von Zustandsübertragung Gebrauch.

Mit einer *Sprache für den musterbasierten Sichtenentwurf* (kurz *Mustersprache*) können verschiedene Interessengruppen ihre individuellen Informationsbedarfe durch Anwendung und Komposition der bereitgestellten Muster als Sichtenentwurf formulieren und nach anschließender Implementierung auf Prozesse anwenden, um damit deren Komplexität auf den individuellen Bedarf zu reduzieren. Ein musterbasierter Sichtenentwurf stellt damit den Ausgangspunkt der Spezifikation komplexer Transformationen und Mechanismen zur Prozessvisualisierung dar. Neben dem planerischen Aspekt kann die Mustersprache zur musterbasierten Analyse von Sichtenfunktionen in bestehenden Ansätzen und Werkzeugen verwendet werden. Dieses Konzept erfordert die neuartige Aufgabe des Informationsdesigns im Management von Geschäftsprozessen, die ein Bindeglied zwischen Informationsbedarfen und der technischen Realisierung von Sichten in Werkzeugen schafft.

Der musterbasierte Ansatz grenzt sich davon ab, Geschäftsprozesse an sich zu verschlanken und tatsächlich stattfindende Abläufe zu verändern. Geschäftsprozesse sind trotz individueller Sichten noch immer komplex. Dieser Ansatz verfolgt vielmehr das Ziel, komplexe Modelle von Geschäftsprozessen verständlicher zu machen. Der Schwerpunkt des Ansatzes besteht darin, Prozesse im Kontext verschiedener Interessengruppen mit Informationen zu vernetzen und die Prozesse bedarfsgerecht darzustellen. Nichtsdestotrotz können Prozesssichten ein Werkzeug für Vorhaben zur Verschlankung oder Optimierung von Abläufen darstellen.

Compliance ist zwischenzeitlich zum Sammelbegriff für die Einhaltung von Anforderungen aus verschiedenen Quellen avanciert und dient in dieser Arbeit als zu untersuchende Einflussgröße bei der weiteren Steigerung der Komplexität: Eine zunehmende Zahl an Regularien, unternehmensinterner Richtlinien, wie auch die moralische und ökologische Verantwortung, rufen eine immense Anzahl an Anforderungen hervor, denen die ohnehin komplexen Geschäftsprozesse gerecht werden sollen. Der Stand der Technik bietet überwiegend formale Methoden an, um die geforderte Regelkonformität sicherzustellen. Diese Methoden können eingesetzt werden, um Fehlverhalten in Modellen von Geschäftsprozessen auszumachen. Sie bieten allerdings keine ausreichende Hilfestellung, wie bestehende Geschäftsprozesse umzugestalten sind, damit die darin enthaltenen Aufgaben regelkonform erledigt werden.

Der Ansatz *regelrealisierender Prozessfragmente* (kurz *rr-Fragmente*) erweitert den Stand der Technik mit einer Möglichkeit zur Wiederverwendung von Prozessstrukturen, die einen Geschäftsprozess nachvollziehbar mit regelkonformem Verhalten erweitern. rr-Fragmente bieten somit Lösungsmöglichkeiten dafür an, wie bestimmte Aufgaben im Einklang mit geltenden Regeln erledigt werden können. Es wird eine Verbindung von dem Ansatz der rr-Fragmente mit Techniken zur Prüfung der Regelkonformität hergestellt und er wird im übergeordneten Kontext vom Compliance Management in Geschäftsprozessen als Baustein positioniert. Wiederverwendbare Prozessstrukturen wurden, nach bestem Wissen, zuvor nicht in den Kontext von Compliance gestellt und an die Anforderungen dieses Themenkomplexes angepasst. Aus dem entwickelten Ansatz leiten sich zahlreiche Informationsbedarfe ab, die durch *Prozesssichten für Compliance* abgedeckt werden können. Diese Sichten unterstützen die Handhabung von rr-Fragmenten im Besonderen und befriedigen beispielhafte Informationsbedarfe eines Betriebsprüfers im Allgemeinen. Sichten dienen als Werkzeug im Themenkomplex Compliance, bieten allerdings keine in sich geschlossene Lösung für die Handhabung von rr-Fragmenten. Daher wird der Ansatz der rr-Fragmente und unterstützender Sichten durch integrative Techniken zu einer ganzheitlichen Lösung ausgebaut.

Die Konzepte in der vorliegenden Dissertation werden an Prozessgraphen diskutiert. Zum Nachweis der praktischen Anwendbarkeit werden wesentliche Konzepte grafisch in der Notation BPMN dargestellt, auf die ausführungsnahe Prozesssprache BPEL abgebildet und Architekturen, prototypische Implementierungen und Erweiterungen von Werkzeugen vorgestellt. Der musterbasierte Sichtenentwurf wurde anhand von Workshops mit Vertretern aus Industrie und Forschung validiert. Zur Validierung des Ansatzes der rr-Fragmente und dessen Unterstützung durch spezifische Sichten werden Fallstudien aus dem Forschungsprojekt COMPAS herangezogen.

Abstract

Complexity is an enduring challenge in the industrialized and globalized world, a challenge that all parts of companies are faced with. The number and variety of requirements from different origin is increasing dramatically and all these requirements have a profound and manifold impact on the interior of the companies and their business relationships. Business processes serve the companies as a blueprint of how to act in an effective and efficient manner in order to achieve specific business goals. However, the business processes shall comply with all relevant requirements and rules in a traceable manner and shall, at the same time, preserve or even increase a company's economic strength. As a consequence, complexity of business processes increases further. If the increasing complexity of business processes is not being addressed with complexity reducing techniques, there is danger of a loss of insight and wrong decisions due to insufficient understanding.

In the past decade different groups have investigated *Views on Processes* to obtain a process-simplifying technique based on model transformations. Views are merely used in current business process tools, though. In addition, there is no common terminology and no common sense about which features beyond the omission and aggregation of process model elements belong to this technique. The question is whether and how this technique may serve a broad spectrum of stakeholders to articulate individual information needs.

Through *a pattern-based approach* an abstract conceptual basis is created that distills the existing heterogeneous possibilities of views on processes. The aim is to express, join and connect pieces of knowledge in one expressive platform that is mostly independent of technologies, tools and process languages. This goal is achieved by abstracting complex functions for process graph transformation into so-called process viewing patterns. These basic forms of transformation are easier to communicate and learn as they hide complexity of the functions they represent. Further, the state of the art is advanced by developing a method for propagating the execution state and by providing the possibility to design stakeholder-specific views on processes that cross process languages. For the latter, use of existing process translation techniques and the emulation of process languages by simply adjusting the graphical display is made.

Having a *pattern language* for view design, different stakeholders are enabled to design views that suit their individual information needs by applying and composing the proposed viewing patterns. By implementing and actually applying views the complexity of the respective processes is reduced according to the individual demand. A pattern-based view design represents an initial specification of the complex transformations and visual mechanisms which are required for technically realizing the designed view. Aside from the main purpose of view design, the pattern language can also be used to describe and analyse the viewing functions in existing approaches and tools. The introduced concepts require a new task to be installed in the management of business processes, a task called information design. This new task aligns the information needs on one side with the technical realization of views in tools on the other side.

Process viewing does not aim at actually making business processes leaner or to really change the flow of work which is prescribed in the process models. Despite having simpler views, the business processes are still complex, however, just for particular tasks and roles they are significantly easier to deal with. The scope of the approach is limited to making complex business processes easier to understand, which requires connecting process models with the working context of different stakeholders and to display the result according to the stakeholders' information needs. Nevertheless, Process Views can potentially be used as a starting point for continuous process improvement or process reengineering endeavors.

Compliance is a term to subsume conformance with a set of requirements derived from different origin. Besides, Compliance is becoming a more and more important factor for the increase of complexity, which is investigated in this work. A growing number of regulations, internal policies, as well as social and ecological expectations from society generate a vast number of requirements. These requirements have profound impact on the business processes in order to make them compliant. The state of the art offers different Compliance checking techniques, most of them are based on formal methods and model checking. These techniques can be employed to identify misbehavior in models of business processes. However, they do not provide assistance in specifying what exactly needs to be done in a process in order to perform the work in a compliant manner.

The approach of *Compliance Fragments* advances the state of the art by providing a means to reuse process structures which are known to implement certain requirements in a compliant manner. Thus, Compliance Fragments address Compliance requirements in a consistent and traceable manner. A relation between the Compliance Fragments approach and Compliance checking techniques is established in order to make both of them more complete. Furthermore, the approach is positioned in the overall context of Compliance management in business processes. To the best knowledge, reusable process structures have not been embedded into the Compliance context before. From the developed Compliance management approach several information needs can be derived which are addressed through the aid of *Process Views for Compliance*. These views support the management of Compliance Fragments in particular and they address exemplary information needs of a Compliance auditor in general. The designed views serve as a tool in the Compliance context. However, they do not offer a self-contained solution. Therefore, further techniques are developed to leverage the approach of Compliance Fragments and supporting views.

The concepts are discussed on the level of process graphs. To provide proof for the practical applicability essential concepts are discussed graphically using the process notation BPMN and transferred to the execution language BPEL and architectures and prototypical implementations are presented. The concept of pattern-based view design has been validated in workshops with representatives from industry and research. The validation of the approach of Compliance Fragments and the respective viewing support is based on case studies from the research project COMPAS.

Teil I

Grundlagen

Überblick zu Teil I

In Teil I werden die Grundlagen für die vorliegende Dissertation gelegt.

Kapitel 1 vermittelt dem Leser einen Überblick über den Kontext und den Problembereich dieser Arbeit. Die wichtigsten Forschungsprobleme werden herausgestellt und die wesentlichen Beiträge der vorliegenden Dissertation benannt.

Kapitel 2 präsentiert elementare Hintergrundinformationen zu Geschäftsprozessen, Sichten und Compliance, um den Stand der Technik zu erfassen und eine begriffliche Basis aufzubauen. Anschließend werden die gegenwärtigen Grenzen bestehender Ansätze analysiert und daraus Anforderungen an Lösungskonzepte abgeleitet.

KAPITEL 1

EINLEITUNG

Zu dem Begriff *Geschäftsprozess* existieren verschiedene Interpretationen. Geschäftsprozesse können bspw. als betriebswirtschaftlicher Aspekt von Arbeitsabläufen aufgefasst werden, wie Jablonski et al. [JBS97] erklären. Nach Hammer und Champy [HC93] kann ein Geschäftsprozess auch als Sammlung von Aktivitäten verstanden werden, die eine oder mehrere Eingaben aufnehmen und etwas produzieren, was für einen Kunden von Wert ist. Geschäftsprozesse können außerdem als Vermögenswerte eines Unternehmens verstanden werden, da sie beschreiben, wie ein Unternehmen seine Geschäftsziele erreicht, wie Leymann und Altenhuber [LA94] feststellen. Für ein Unternehmen ist es von strategischer Bedeutung, ein klares *Verständnis* der eigenen Geschäftsprozesse und deren *Regelkonformität* zu erlangen, um die Wettbewerbsfähigkeit aufrecht erhalten und sich an geänderte Rahmenbedingungen anpassen zu können.

1.1 Problembereich und Motivation

Die Effizienz eines Geschäftsprozesses kann in mehreren Dimensionen untersucht werden. Hammer und Champy [HC93] geben bspw. Kosten, Qualität, Geschwindigkeit und angebotene Dienstleistung als kritische Dimensionen an. Auch Flexibilität ist eine häufig genannte Dimension der Effizienz. Eine Möglichkeit, um Verbesserungen in diesen Dimensionen zu erzielen, besteht im Einsatz technischer Mittel zur Automatisierung von Geschäftsprozessen. Workflow Management Systeme (WfMS), auch bekannt als Business Process Management Systeme (BPMS), haben sich als Klasse von Softwaresystemen entwickelt, die speziell diese Form der Automatisierung bietet [LR00]. Diese Technologie erfordert eine formale Beschreibung bzw. Modellierung von Geschäftsprozessen in Prozessmodellen.

Allerdings geht die zunehmende Automatisierung und Vernetzung mit einer steigenden Komplexität in einer Vielzahl von Aspekten einher, die bei der stärkeren Formalisierung in Modellen deutlicher wird, jedoch auch bei nicht automatisierten Geschäftsprozessen besteht. Geschäftsprozessmodelle können heutzutage teils mehrere Hundert Aktivitäten beinhalten [VVL07]. Diese Aktivitäten repräsentieren vornehmlich die Aufgaben von Menschen, den Aufruf von Programmen oder Konstrukte zur Ablaufsteuerung. Neben den Menschen, die tatsächlich in einen Prozess als ausführende Kraft involviert sind, gibt es noch zahlreiche weitere Akteure und Interessengruppen (engl. *stakeholder*), die an den verschiedenen Phasen im Lebenszyklus eines Geschäftsprozesses beteiligt sind. Jeder dieser Akteure bzw. jede

Interessengruppe hat eigene Aufgaben und einen unterschiedlichen Informationsbedarf. Ein Prozesseigner[1] mag die wirtschaftlichen Aspekte wie Kosten, Organisation und Ressourcen verantworten, während sich ein technischer Mitarbeiter mit Aspekten wie Sicherheit, Dienstgüte oder der Auslastung von Rechensystemen befasst. Ebenso spielen ökologische Aspekte wie Energieeffizienz und CO_2-Ausstoß eine immer größere Rolle [Wil07][Gar09].

Ein weiterer Bereich, dem aufgrund von Korruptionsskandalen und daraus resultierenden strafrechtlichen Ermittlungen und Imageschäden eine sehr große Bedeutung beigemessen wird, ist der Bereich *Compliance*, der sich mit der Einhaltung von geltenden Vorschriften befasst [Moo10]. Compliance eines Unternehmens, im deutschen Sprachraum auch unter den Begriffen Regelkonformität oder Regeltreue bekannt, bezeichnet die Einhaltung von Anforderungen, die sich aus Gesetzen, Richtlinien, Verträgen und weiteren Quellen ableiten lassen [DCD+09]. Ein Geschäftsprozess gilt demnach als regelkonform, regeltreu oder auch compliant, wenn darin alle relevanten Anforderungen umgesetzt sind.

Eine Schwierigkeit besteht in der Klärung, welche Anforderungen für ein bestimmtes Unternehmen und einen konkreten Geschäftsprozess gelten. Bspw. gilt das Regelwerk Sarbanes-Oxley Act (SOX) [Con02] für Unternehmen, deren Wertpapiere in den USA gehandelt werden. Das Regelwerk Basel II [Bas06] stellt eine Sammlung von Eigenkapitalvorschriften dar, die bei Banken Anwendung findet. Häufig treffen mehrere Regelwerke auf ein Unternehmen im Allgemeinen und auf dessen Geschäftsprozesse im Besonderen zu. Eine deutsche Bank ist bspw. an das Regelwerk Basel II gebunden. Werden deren Unternehmenswertpapiere an der US-amerikanischen Börse gehandelt, ist sie demnach auch zur Einhaltung der Regeln aus dem Regelwerk SOX verpflichtet. In Abhängigkeit der Geschäftsbeziehungen, die ein Unternehmen unterhält, können zudem kartellrechtliche Regelungen von Bedeutung sein. Weitere Regeln gelten, wenn IT in der Rechnungslegung zum Einsatz kommt. Der Prüfstandard EPS 330 umfasst u.a. Vorgaben zur Prüfung IT-gestützter Geschäftsprozesse, deren Umsetzung von Wirtschaftsprüfern untersucht wird, u.U. unter Hinzuziehen von IT-Experten [IDW01].

Wirtschafts- und Unternehmensprüfer (engl. *auditor*) stellen folglich eine besondere Interessengruppe dar, die es zu unterstützen gilt, da sie über Regelkonformität entscheiden. Zu gesetzlichen Vorschriften kommen u.U. weitere Anforderungen aus unternehmensinternen Richtlinien, Verträgen mit anderen Unternehmen und Industriestandards hinzu. Für einen Kreditvergabeprozess gelten bspw. zusätzlich zu den Vorschriften aus den gesetzlichen Regelwerken ggf. unternehmensinterne Richtlinien zur Risikominimierung. Zudem ist das Thema Sicherheit nur schwer von Compliance zu trennen: Wie das Bundesamt für Sicherheit in der Informationstechnik [BSI09] feststellt, gibt es speziell im Bereich IT-Sicherheit zahlreiche Regelwerke, die sich auf Geschäftsprozesse und deren technische Implementierung beziehen. Das Regelwerk PCI Data Security Standard [PCI10] beschreibt bspw. zahlreiche Anforderungen, die für Unternehmen gelten, die Daten von Kreditkarten

[1] Bei Rollenbezeichnungen wird die männliche Form verwendet, wobei diese für Männer und Frauen gleichermaßen gilt.

verarbeiten. Diese Aufzählung von Quellen von Compliance-Anforderungen führt zu der Einschätzung, dass das Spektrum von Compliance als breit betrachtet werden kann. Die zuvor erwähnten betriebswirtschaftlichen, technischen und ökologischen übergreifenden Aspekte sind je nach Interpretation des Begriffs ebenfalls mit eingeschlossen und tragen zu einer hohen Komplexität und zu einer Vielschichtigkeit von Compliance bei.

Die Anforderungen, die von Fachleuten wie Anwälten, Juristen oder Sicherheitsexperten aus den Regelwerken abgeleitet werden, haben zahlreiche und tiefgreifende Auswirkungen auf die Geschäftsprozesse eines Unternehmens. Die abgeleiteten Anforderungen betreffen sowohl den Prozessaufbau, die verwendeten IT-Systeme als auch die Mitarbeiter und Geschäftspartner eines Unternehmens [SAL+10a]. Die Umsetzung der Anforderungen ist eine zeitintensive Angelegenheit, die sich je nach Komplexität der zutreffenden Regelwerke und Geschäftsprozesse über Monate oder Jahre erstrecken kann [Moo10]. Eine Änderung größerer Regelwerke findet zwar statt, deren Verbindlichkeit setzt allerdings erst mit einer gewissen Zeitverzögerung ein. So wurde bspw. das Regelwerk Basel III [Bas10] Ende des Jahres 2010 veröffentlicht, dessen vollständige Umsetzung ist aber erst Jahre später verpflichtend [Bas10]. Auch wenn unternehmensinterne Richtlinien und Verträge mit anderen Unternehmen einem schnelleren Änderungszyklus unterworfen sind, ist eine unmittelbare Umsetzung neuer oder geänderter Anforderungen nur bedingt möglich. Die Komplexität der Geschäftsprozesse und die Vielzahl der darin enthaltenen Aspekte sind ein Hindernis, das einer zeitnahen Reaktion auf neue Anforderungen entgegensteht.

1.2 Forschungsschwerpunkte und Beiträge

Im Fokus der vorliegenden Dissertation steht die zunehmende *Komplexität von Geschäftsprozessen*. Es sollen methodische und technologische Ansätze entwickelt werden, mit denen diese Komplexität aufgaben-, rollen- und interessengruppenspezifisch handhabbar gemacht werden kann. Die Thematik *Compliance* mit ihren vielschichtigen Anforderungen an Geschäftsprozesse dient als beispielhaftes Anwendungsgebiet dieser Ansätze und wird daher eingehender betrachtet. Den Mittelpunkt der Forschung bilden *Sichten auf Geschäftsprozesse*, die einen Ausschnitt eines Geschäftsprozesses bedarfsgerecht darstellen. Mit Sichten auf Geschäftsprozesse, oder kurz mit Prozesssichten, können Sachverhalte zusammengefasst, für eine bestimmte Aufgabe oder für einen bestimmten Akteur unwichtige Informationen ausgeblendet, relevante Informationsquellen außerhalb des Prozessmodells eingebunden und das resultierende Prozessmodell entsprechend unterschiedlicher Anforderungen dargestellt werden [SAL+10b]. Eine Prozesssicht formt einen bereits bestehenden Prozess vorübergehend um, damit die für eine Interessengruppe und Aufgabenstellung wesentlichen Informationen enthalten sind und weniger relevante Informationen in den Hintergrund treten.

Eine beispielhafte Prozesssicht und deren schrittweise Verfeinerung, die Schumm et al. [SCF+11] diskutiert haben, ist in Abbildung 1.1 skizziert. Ausgehend von einem noch unveränderten Modell eines Geschäftsprozesses werden in jedem Schritt verschiedene Transformationen durchgeführt: In Schritt A werden organisatorische Verteilungsinformationen in das Prozessmodell eingebunden und durch Swimlanes dargestellt; in Schritt B werden

zusätzlich Ausführungsinformationen hinzugefügt; in Schritt C werden die als erledigt gekennzeichneten Aktivitäten ausgeblendet.

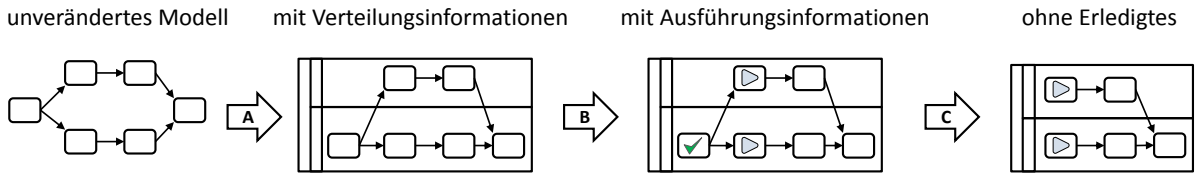


Abbildung 1.1: Beispielhafte Prozesssicht und deren schrittweise Verfeinerung, vgl. [SCF+11].

In der Ausarbeitung eines Ansatzes zur Komplexitätsreduzierung von Geschäftsprozessen auf Basis von Prozesssichten wurden mehrere Problemstellungen identifiziert:

- Es besteht ein Mangel an einer strukturierten Beschreibungsform, mit der eine Sicht auf einen Prozess entworfen und spezifiziert werden kann.
- Zum Erhalt und der Verbesserung der Wettbewerbsfähigkeit spielen das Monitoring und die damit verbundene Analyse der Ausführung von Prozessen eine entscheidende Rolle. Unter diesem Gesichtspunkt stellt sich die Frage, ob und inwieweit verschiedene Akteure beim Monitoring durch Sichten unterstützt werden können.
- Sichten zur Unterstützung verschiedener Interessengruppen machen von ähnlichen Transformationen Gebrauch. Daraus ergibt sich die Fragestellung, wie Bestandteile von Sichten wiederverwendbar gemacht werden können, um so die Effizienz des Sichtenentwurfs und deren Realisierung zu steigern.

In der Betrachtung von Compliance als beispielhaftem Anwendungsgebiet der entwickelten Ansätze zu Sichten auf Geschäftsprozesse haben sich weiterführende Problemstellungen offenbart. Ein Problem besteht darin, dass Geschäftsprozesse häufig nur in Teilen automatisiert sind und dazu zahlreiche unterschiedliche Technologien zum Einsatz kommen. Ein Ansatz für Compliance in Geschäftsprozessen sollte diesem Umstand Rechnung tragen und außerdem den gesamten Prozesslebenszyklus berücksichtigen. Auch im Compliance Management stellt sich das Problem, wie durch Wiederverwendung von bewährten Lösungen eine Effizienzsteigerung erzielt werden kann.

Beiträge. Nachfolgend werden die wesentlichen Beiträge der vorliegenden Dissertation vorgestellt, welche die genannten Problemstellungen berücksichtigen. Die Beiträge 1-4 sind weitgehend unabhängig von einer konkreten Prozesssprache oder Technologie, da in diesen Beiträgen von einem abstrakten, graphbasierten Metamodell ausgegangen wird. In den Beiträgen 5 und 6 wird die praktische Umsetzbarkeit der vorgestellten Beiträge anhand von Architekturen und deren Implementierung für die Business Process Execution Language (BPEL) [OAS07] demonstriert. Zur Unterstützung des Verständnisses werden prozessstrukturbezogene Konzepte und umfassende Ablaufbeschreibungen in der Business Process Model and Notation (BPMN) [OMG11b] veranschaulicht. Die Nutzung der zusätzlichen Prozesssprache BPMN erlaubt die Vermittlung von Konzepten in einer standardisierten Darstellung. Technische Aspekte treten bei diesen Darstellungen in den Hintergrund.

Die wissenschaftlichen Beiträge sind in die nachfolgenden Kernbereiche eingeordnet.

Beitrag 1: Eine Methode zum Entwurf von Sichten auf Basis einer Mustersprache

Ein grundlegendes Problem im Management von Geschäftsprozessen ist deren steigende Komplexität. Eine Sicht auf einen Geschäftsprozess soll komplexe Zusammenhänge einfacher und verständlicher machen. In diesem Beitrag wird eine Methode vorgestellt, mit dem Sichten auf Geschäftsprozesse auf Basis einer Mustersprache entworfen werden können. Diese Methode macht von verschiedenen Transformationsfunktionen Gebrauch, die zu Mustern abstrahiert und als Musterkatalog bereitgestellt sind. Die Transformationen haben unterschiedliche Effekte auf einen Geschäftsprozess. Diese Effekte lassen sich untergliedern in reduzierende und erweiternde Strukturveränderung, Informationsanreicherung, Darstellungsanpassung und transformationsbeeinflussende Vorgaben.

Beitrag 2: Eine Methode zum Monitoring von Geschäftsprozessen auf Basis sprachübergreifender Sichten

Geschäftsprozesse werden in der fachlichen Kommunikation mitunter auf einem vergleichsweise hohen Abstraktionsgrad betrachtet, während ein ausspezifiziertes und technisch verfeinertes Modell der Geschäftsprozesse für die automatisierte Ausführung verwendet wird. Ein Problem an diesem Umstand ist, dass der Zustand einer Instanz eines Geschäftsprozesses nur für das vollständige Ausführungsmodell bekannt ist, jedoch nicht für das abstraktere Ausgangsmodell. In diesem Beitrag werden Sichten auf Geschäftsprozesse als Mittel zur abstrakten Darstellung von Prozessinstanzen entworfen. Diese Sichten stellen eine Abstraktion des vollständigen, zustandsbehafteten Modells dar. Der Ausführungszustand einer Prozessinstanz kann, wie in diesem Beitrag gezeigt wird, auf ein abstraktes Modell projiziert werden, wobei sich die Prozesssprache des Ausführungsmodells und des abstrakten Modells unterscheiden können.

Beitrag 3: Eine Methode zum Management von Compliance in Geschäftsprozessen

In diesem Beitrag werden Aspekte von Compliance und deren Bedeutung für Unternehmen diskutiert. Die Diskussion von Compliance umfasst grundlegende Begriffsklärungen, eine Analyse der Bedeutung von Compliance für die Geschäftsprozesse in Unternehmen sowie eine Betrachtung technischer als auch nicht technischer Mechanismen, die im Stand der Technik zur Gewährleistung von Compliance in Unternehmen eingesetzt werden. Auf Basis dieser Diskussion wird eine Vorgehensweise zum Management von Compliance in Geschäftsprozessen vorgeschlagen. Die Methode berücksichtigt, dass Geschäftsprozesse in einem Unternehmen nicht notwendigerweise mit informationstechnischer Unterstützung realisiert sind.

Beitrag 4: Konzept und Techniken zur Nutzung regelrealisierender Prozessfragmente

Um Compliance in einem Geschäftsprozess sicher zu stellen, sind verschiedene Maßnahmen der Prüfung, Reaktion und Aktion erforderlich. Eine Maßnahme der Aktion besteht in der Anpassung der Ablauflogik von Geschäftsprozessen. In diesem Beitrag werden regelrealisierende Prozessfragmente (rr-Fragmente) konzipiert, um diese Maßnahme zu unterstützen. Ein rr-Fragment ist eine wiederverwendbare aber unvollständige Prozessstruktur, welche bestimmte Anforderungen durch Aktivitäten, Kontrollfluss und Datenfluss realisiert. Auf Basis dieses Konzepts werden Techniken zur Nutzung von rr-Fragmenten entworfen. Diese Techniken umfassen Sichten auf Prozesse für das Ausschneiden, das Ausblenden sowie das Hervorheben von Prozessfragmenten sowie weitere Techniken, die nicht mit Sichten realisiert werden können. Letzteres beinhaltet die Integration von rr-Fragmenten in einen bestehenden Prozessgraphen und die anschließende Prüfung der Regelkonformität.

Beitrag 5: Konzept und Techniken zur Komposition von Sichten auf Basis von Diensten

Ein weiterer Beitrag dieser Arbeit besteht in der dienstbasierten Komposition von Sichten auf Geschäftsprozesse. Sichten, die auf diese Weise erzeugt werden, können bspw. zur Unterstützung von Analyseaufgaben eingesetzt werden. Neben der dazu erforderlichen Vorgehensweise und der dienstbasierten Bereitstellung von Transformationsfunktionen werden Anwendungsszenarien von Prozesssichten für den Prozesslebenszyklus im Allgemeinen sowie für den Anwendungsbereich Compliance im Besonderen betrachtet. Sichten auf Geschäftsprozesse haben bestimme charakteristische Eigenschaften. Als Teil dieses Beitrags werden diese Eigenschaften analysiert und ein Klassifikationsschema erstellt. In dieser Arbeit erstellte Sichtenentwürfe werden nach diesem Schema klassifiziert.

Beitrag 6: Architekturen und Implementierung

Um die praktische Umsetzbarkeit der vorgestellten Beiträge zu demonstrieren wurden Architekturen zur Prozessmodellierung, -darstellung und für das Monitoring entworfen. Diese Architekturen umfassen die Unterstützung von Sichten auf Geschäftsprozesse im Hinblick auf Compliance als auch die zur Verwendung und Verwaltung von rr-Fragmenten erforderlichen Komponenten und Erweiterungen. Die Architekturen sind prototypisch in der Prozesssprache BPEL umgesetzt. Die Umgebung beinhaltet ein grafisches Werkzeug zur Prozessmodellierung, eine Bibliothek zur Verwaltung von Prozessen und Prozessfragmenten, ein dienstbasiertes Framework zur Erstellung von Sichten auf Prozesse sowie ein Framework zum sichtenbasierten Monitoring von Prozessen. Für die Realisierung sprachübergreifender Sichten werden die für diesen Ansatz erforderlichen Artefakte vorgestellt.

Ein Teil der im Rahmen der vorliegenden Dissertation implementierten Software wurde unter Open Source veröffentlicht. Vgl. dazu die Diplom- und Studienarbeiten von Streule [Str09], Latuske [Lat10] und Krein [Kre11]. Die von Streule [Str09] implementierte Unterstützung der Opaque Activities aus dem Themengebiet *Abstract BPEL* hat erfolgreich zu dem Projekt Eclipse BPEL Designer [Ecl11] beigetragen. Ferner bestanden Pläne, einen Teil der von Latuske [Lat10] entwickelten Software in ein Produkt des Herstellers WSO2 [WSO12] als Komponente zur sichtenbasierten Darstellung laufender BPEL-Prozesse zu integrieren.

Eine Zusammenfassung der identifizierten Forschungsprobleme und entsprechender Beiträge der vorliegenden Dissertation ist in Tabelle 1.1 aufgeführt.

Tabelle 1.1: Zusammenfassung der identifizierten Forschungsprobleme und entsprechender Beiträge.

Forschungsproblem	Beitrag der Dissertation
Wie kann man Sichten auf Geschäftsprozesse entwerfen?	1.) Eine Methode zum Entwurf von Sichten auf Basis einer Mustersprache
Wie kann man verschiedene Interessengruppen beim Monitoring von Geschäftsprozessen durch Sichten unterstützen?	2.) Eine Methode zum Monitoring von Geschäftsprozessen auf Basis sprachübergreifender Sichten
Wie kann man Compliance über den Prozesslebenszyklus sicherstellen?	3.) Eine Methode zum Management von Compliance in Geschäftsprozessen
Wie kann man Prozesswissen über Regelkonformität wiederverwenden?	4.) Konzept und Techniken zur Nutzung regelrealisierender Prozessfragmente
Wie kann man Prozesssichten flexibel wiederverwendbar machen?	5.) Konzept und Techniken zur Komposition von Sichten auf Basis von Diensten
Wie lassen sich die aufgeführten Ansätze realisieren?	6.) Architekturen und Implementierung

1.3 Aufbau der Arbeit

Die vorliegende Dissertation gliedert sich in fünf Teile.

In Teil I werden die *Grundlagen der Arbeit* geschaffen. In Kapitel 2 werden die Themengebiete Geschäftsprozesse, (Prozess-)Sichten und Compliance umfassend vorgestellt, welche den begrifflichen Rahmen und die wissenschaftliche Fundierung der vorliegenden Arbeit bilden. Auf dieser Basis werden Möglichkeiten und Grenzen analysiert und Anforderungen formuliert.

In Teil II wird der musterbasierte Ansatz von *Sichten auf Geschäftsprozesse* entwickelt. Kapitel 3 stellt den Ansatz zum Entwurf von Sichten auf Basis eines Musterkatalogs und ein Klassifikationsschema für Sichtenentwürfe vor. Außerdem werden organisatorische Aspekte im Hinblick auf den Prozesslebenszyklus betrachtet. Die vorgestellte Entwurfsmethode wird in Kapitel 4 dazu eingesetzt, beispielhafte Sichten für verschiedene Interessengruppen und Phasen des Prozesslebenszyklus zu entwerfen. Anschließend werden in Kapitel 5 sprachübergreifende Sichten entworfen, die die Darstellung eines Prozesses in einer anderen Prozesssprache ermöglichen. Dieses Kapitel vertieft den Bereich des Monitorings von Prozessen.

In Teil III wird die Thematik *Compliance* diskutiert und damit in Verbindung stehende Interessengruppen und Informationsbedarfe beleuchtet. In Kapitel 6 wird ein ganzheitlicher Ansatz zum Management von Compliance in Geschäftsprozessen entworfen, der verschiedene Reifegrade der Automatisierung berücksichtigt. Regelrealisierende Prozessfragmente (kurz rr-Fragmente) werden als essenzieller Bestandteil des Ansatzes vorgestellt. In Kapitel 7 werden Prozesssichten mit Bezug auf Compliance entworfen, um das Management von Compliance zu vereinfachen. In Kapitel 8 werden weitere Techniken vorgestellt, die zur effizienten Handhabung von rr-Fragmenten benötigt werden.

In Teil IV wird die *technische Realisierbarkeit* der entwickelten Ansätze diskutiert. In Kapitel 9 wird zunächst eine Abbildung wesentlicher Konzepte auf die Prozesssprache BPEL vorgenommen, einem verbreiteten Industriestandard zur Prozessautomatisierung auf Basis Serviceorientierter Architekturen (SOA). In Kapitel 10 werden Architekturen entworfen, die die Realisierung der auf BPEL abgebildeten Konzepte in IT-Systemen konkretisieren. Dieses Kapitel beinhaltet außerdem Beschreibungen zur vorgenommenen Implementierung.

In Teil V wird die Arbeit abgeschlossen. Kapitel 11 stellt die vorgenommene *Validierung* des musterbasierten Sichtenentwurfs und der rr-Fragmente dar. Kapitel 12 fasst die in dieser Arbeit erzielten Ergebnisse zusammen und stellt die *Erkenntnisse* der entwickelten Ansätze heraus. Kapitel 13 gibt dem Leser einen Ausblick auf Möglichkeiten zur Anwendung der entwickelten Ansätze in anderen Bereichen und greift weiterführende Problemstellungen auf.

KAPITEL 2

HINTERGRUND UND VERWANDTE ARBEITEN

In diesem Kapitel wird eine Terminologie und konzeptionelle Basis für die vorliegende Dissertation geschaffen. Abschnitt 2.1 beschreibt das Grundverständnis über Geschäftsprozesse. In Abschnitt 2.2 wird der Stand der Technik von Sichten vorgestellt. In Abschnitt 2.3 wird der Stand der Technik von Compliance diskutiert. Auf dieser Basis wird in Abschnitt 2.4 eine Anforderungsanalyse und eine Positionierung der Dissertation vorgenommen.

2.1 Grundlagen zu Geschäftsprozessen

Dieser Abschnitt beginnt mit einer Begriffsklärung unter Berücksichtigung verschiedener Automatisierungsgrade. Nachfolgend wird die technische Infrastruktur skizziert, die in der Prozessautomatisierung gängig ist. Diese Infrastruktur steht in engem Zusammenhang mit den verschiedenen Lebenszyklusphasen, die ein automatisierter Prozess absolviert. Diese Lebenszyklusphasen werden anschließend beschrieben. Darauf folgt die Definition eines graphbasierten Metamodells für Prozessskizzen. Der Abschnitt schließt mit einem Diskurs über das Verständnis von Komplexität von Geschäftsprozessen in der Literatur.

2.1.1 Beschreibung und Automatisierung von Geschäftsprozessen

Ein Geschäftsprozess besteht aus einer Menge von Aktivitäten, die in einer bestimmten Reihenfolge auszuführen sind. Aktivitäten können von Menschen oder Maschinen ausgeführt werden, wobei mit Maschinen meist IT-Systeme gemeint sind. Durch diese Akteure wirken die Organisationsstruktur und die IT-Infrastruktur eines Unternehmens auf einen Geschäftsprozess ein.

Geschäftsprozesse können nach Reifegraden unterschieden werden. Bspw. werden in dem Rahmenwerk COBIT [ITG05] Reifegrade dahingehend abgestuft, ob für einen bestimmten Bereich ein (Standard-)Prozess existiert, ob er eher ad hoc ist, ob er einer wiederholbaren Struktur folgt, ob er dokumentiert ist und gemessen wird oder ob er technisch unterstützt und optimiert ist. Das Reifegradmodell, das in der vorliegenden Dissertation verwendet wird, differenziert hingegen stärker zwischen verschiedenen Entwicklungsstufen der Beschreibung und Automatisierung.

Ein Geschäftsprozess ist nicht unbedingt technisch realisiert, wodurch mehrere Beschreibungsformen und Automatisierungsstufen unterschieden werden können:

1. Unbeschriebene Geschäftsprozesse: Viele Geschäftsprozesse laufen ab, ohne dass eine Dokumentation darüber vorhanden ist, welche Aktivitäten in dem Prozess enthalten sind und welche Rollen und welche IT-Systeme darin involviert sind.
2. Textuell beschriebene Geschäftsprozesse: Eine textuelle Beschreibung stellt eine Vorstufe zur technischen Realisierung eines Geschäftsprozesses dar. Die textuelle Beschreibung ist nicht notwendigerweise vollständig und entspricht nur bedingt dem Arbeitsablauf, der in einer Organisation tatsächlich stattfindet.
3. Modellierte Geschäftsprozesse: Die Beschreibung in Form eines Prozessmodells stellt eine weitere Vorstufe zur technischen Realisierung dar. Ein Prozessmodell ist nicht immer vollständig und i.d.R. nicht ausführbar. Bspw. kann ein Prozessmodell die syntaktischen Vorgaben der Prozesssprache verletzen, in der es modelliert wurde. Ein solches Prozessmodell hat lediglich Diagrammcharakter.
4. Ausführbare Geschäftsprozesse: Damit ein Geschäftsprozess ausführbar ist, muss er in einem Prozessmodell vorliegen und den syntaktischen Vorgaben der verwendeten Prozesssprache entsprechen. Des Weiteren muss für die verwendete Prozesssprache eine Ausführungsumgebung bereitstehen. Ein ausführbarer Geschäftsprozess kann sowohl Aktivitäten beinhalten, die von Menschen durchgeführt werden als auch Aufrufe von IT-Systemen. Man spricht bei den Modellen ausführbarer Geschäftsprozesse auch von Workflows [LR00].
5. Teilweise ausführbare Geschäftsprozesse: Ein Geschäftsprozess kann mehrere Teilstrukturen beinhalten, die technisch als Workflow realisierbar sind. Andere Teilstrukturen eines Geschäftsprozesses können jedoch auch außerhalb der IT-Systeme stattfinden oder spielen sich in Informationssystemen ab, in denen sich der Arbeitsablauf nicht durch ein Prozessmodell steuern lässt.

Geschäftsprozesse können gleichzeitig in verschiedenen Beschreibungsformen und zudem in unterschiedlichen Abstraktionsgraden vorliegen und genutzt werden [HLD+05]. Bspw. kann anfänglich eine einfache textuelle Beschreibung eines Geschäftsprozesses erstellt werden, die nachfolgend durch eine detailliertere Fassung ergänzt wird. Vergleichbar kann ein Geschäftsprozess anfänglich durch ein einfaches Prozessmodell mit wenigen Aktivitäten beschrieben werden, um die wesentlichen Aspekte des Ablaufs darzustellen ohne bspw. auf Ausnahmefälle, Eskalation oder Ausführbarkeit einzugehen. Nachfolgend kann ein verfeinertes Prozessmodell erstellt werden, welches Aktivitäten und deren Abhängigkeiten verfeinert und auf Details wie Ausnahme- und Fehlerbehandlung eingeht und dadurch eine höhere Komplexität als das anfängliche Prozessmodell aufweist. Zur Ausführbarkeit muss ein Prozessmodell u.U. in eine andere Prozesssprache überführt und an die zur Verfügung stehende IT-Infrastruktur angepasst werden.

Im Rahmen der vorliegenden Dissertation werden Ansätze entwickelt, die Geschäftsprozesse der Stufen 3, 4 und 5 betreffen. Das Vorhandensein eines Prozessmodells wird demnach in den Ansätzen stets vorausgesetzt.

2.1.2 Geschäftsprozesstechnologie

Für die technische Realisierung eines Geschäftsprozesses oder Teilen daraus werden häufig unterschiedliche IT-Systeme eingesetzt. Unternehmen setzen neben Standardsoftware auch individuell entwickelte, datenbankbasierte Anwendungen ein, um ihre Geschäftsprozesse technisch zu unterstützen. Unternehmen setzten darüber hinaus Informationssysteme zur technischen Unterstützung ihrer Geschäftsprozesse ein, wie bspw. Systeme für das Enterprise Resource Planning, Customer Relationship Management und Produktdatenmanagement. Allerdings erstrecken sich Geschäftsprozesse häufig über mehrere IT-Systeme und involvieren Menschen aus unterschiedlichen Abteilungen, die wiederum über verschiedene Standorte verteilt sind. Workflow Management Systeme (WfMS) haben sich neben nachrichtenbasierter Middleware (Message-oriented Middleware, MoM) an dieser Stelle als Integrationslösung etabliert.

Ein Workflow beschreibt die technische Realisierung eines Arbeitsablaufs, der Menschen und IT-Systeme verbindet und deren Zusammenarbeit koordiniert. Ein Workflow ist in einer Prozesssprache als ausführbares Prozessmodell beschrieben, welches in einer passenden Ausführungsumgebung verwendet werden kann. Zwei Aspekte dieser Geschäftsprozesstechnologie sind (1) die dazu verwendbaren Prozesssprachen und (2) die technische Infrastruktur, die zur Ausführung von Workflows benötigt wird.

Prozesssprachen. Petri Netze [Pet62], die aus der Erforschung von Automaten hervorgegangen sind, spielen in der Prozesstheorie eine wichtige Rolle. Petri Netze können dazu eingesetzt werden, die Ausführungssemantik von Prozesssprachen formal zu beschreiben, was den Nachweis bestimmter Prozesseigenschaften erleichtert, wie bspw. das Vorhandensein von Deadlocks [Aal99][LVD08].

UML Aktivitätsdiagramm (UML AD) ist ein Diagrammtyp, der von der Object Management Group (OMG) zusammen mit anderen Diagrammtypen zur Modellierung von Softwaresystemen in der Unified Modeling Language (UML) [OMG11a] standardisiert wurde. Im Bereich der Modellierung von Geschäftsprozessen stellt außerdem die Prozesssprache Ereignisgesteuerte Prozessketten (EPK) [KNS92] eine wichtige Errungenschaft dar.

Während bei EPK und UML AD der Fokus auf der Modellierung liegt, zielt die Business Process Execution Language (BPEL) auf die Ausführung und Automatisierung [Wes07]. Für die Prozesssprache BPEL, welche von der Organization for the Advancement of Structured Information Standards (OASIS) standardisiert wurde [OAS07], wurde lediglich ein Beschreibungsformat in der eXtensible Markup Language (XML) definiert, d.h. eine grafische Darstellung wurde nicht standardisiert. Es gibt zwar grafische Werkzeuge zur Modellierung von Prozessen in BPEL, diese verwenden allerdings eine herstellerspezifische Darstellung.

Im Gegensatz dazu ist bei der Prozesssprache Business Process Model and Notation (BPMN) [OMG11b] die grafische Darstellung Umfang der Spezifikation. Der Schwerpunkt von BPMN lag ursprünglich auf der Modellierung, unterdessen wurde die Ausführung jedoch stärker berücksichtigt und diese Sprache entsprechend erweitert. BPMN gewinnt im Bereich der Modellierung von Geschäftsprozessen zunehmend an Bedeutung.

Prozesssprachen unterscheiden sich hinsichtlich ihrer Ausführungssemantik, der angebotenen Sprachelemente, der grafischen Darstellung der Sprachelemente sowie in ihrer Struktur. Die Struktur einer Prozesssprache kann z.B. graphbasiert, regelbasiert oder datengetrieben sein, wobei zur Modellierung von Geschäftsprozessen i.d.R. graphbasierte Sprachen eingesetzt werden [Wes07]. Neben den zuvor aufgezählten Sprachen wurde eine beträchtliche Zahl weiterer Prozesssprachen entwickelt, die jedoch nur eine geringe Verbreitung und Anwendung in der Praxis fanden. Standardisierung spielt bei der Verbreitung einer Prozesssprache im Unternehmensumfeld eine wichtige Rolle. Sie kann einen positiven Effekt auf die Verbreitung einer Sprache haben, da sie eine zu enge Bindung an einen Hersteller verhindert (engl. *vendor lock-in*). Standardisierung ist jedoch keine Erfolgsgarantie.

Mit einer breiten Herstellerunterstützung hat sich die standardisierte Sprache BPEL zu einem de facto Industriestandard für ausführbare Prozesssprachen entwickelt. Zudem erleichtert die Verfügbarkeit von quelloffenen Werkzeugen zur Modellierung und Ausführung von Prozessmodellen in BPEL den Nachweis praktischer Realisierbarkeit wissenschaftlicher Konzepte. Eine Aufstellung und Analyse der zahlreichen Konzepte, die durch praktische Erweiterungen an der Prozesssprache BPEL demonstriert wurden, zeigen z.B. Kopp et al. [KGK+11]. Daher wurde BPEL als Ausgangsbasis für die technische Realisierung gewählt.

Technische Infrastruktur. Im Folgenden wird die konzeptionelle technische Infrastruktur im Stand der Technik zur Automatisierung von Geschäftsprozessen mit ausführbaren Prozessmodellen vorgestellt, entsprechend der Darstellung von Schumm und Leymann [SL10]. Abbildung 2.1 zeigt die wesentlichen Kategorien von Komponenten in dieser Infrastruktur.

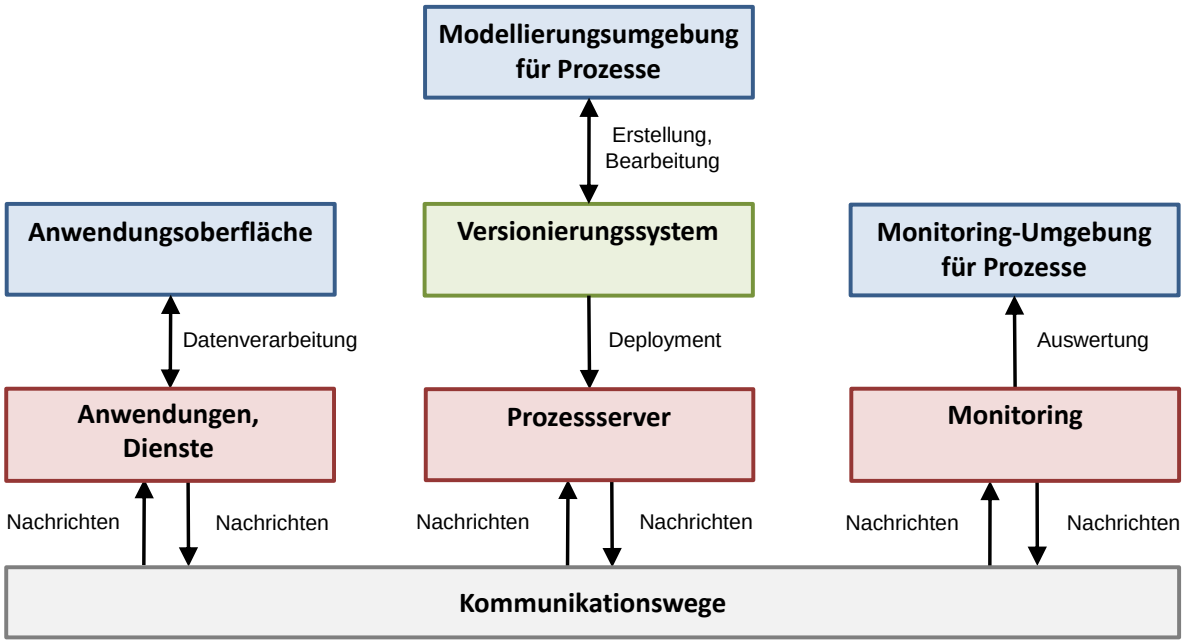


Abbildung 2.1: Infrastruktur zur Automatisierung von Geschäftsprozessen, vgl. [SL10].

In Abbildung 2.1 wird eine *Farbcodierung* verwendet. Diese Farbcodierung wird auch in den Mustersymbolen und Architekturdiagrammen dieser Arbeit verwendet. Die Farbwahl ist von der implizit verwendeten Farbcodierung der Architekturdiagramme in [Lat10] geprägt. Die Semantik der Farbcodierung liest sich wie folgt:

- Blau: wesentlicher Bezug zum Endanwender; Benutzerschicht; Darstellung.
- Rot: Teil- oder vollautomatische Ausführung; Geschäftsschicht; Verarbeitung.
- Grün: Datenhaltung und -verwaltung; Datenschicht; Informationsbereitstellung.
- Grau: Kommunikation von Komponenten.

Im oberen Bereich von Abbildung 2.1 befinden sich die Komponenten, die einen direkten Bezug zu Anwendern herstellen, die in die Planung, in die Durchführung und in das Monitoring von Prozessen involviert sind. In der Modellierungsumgebung für Prozesse werden die Modelle von Prozessen erstellt, d.h. dort werden die Prozesse modelliert. Werkzeuge dieser Kategorie haben meist eine grafische Unterstützung, um die Planung, Erstellung und Analyse von Prozessmodellen zu erleichtern. Prozessmodelle und zugehörige Artefakte wie Schnittstellenbeschreibungen und Datenformate werden in einem Versionierungssystem verwaltet, über das sie auf einem Prozessserver deployed werden können. Nach dem Deployment stehen die Prozesse zur Ausführung bereit.

Die Ausführung eines Prozesses wird durch ein Ereignis gestartet, wie bspw. durch eine eingehende Nachricht, die von einer Anwendung versendet wird. Bei der Ausführung eines Prozesses wird das entsprechend der eingehenden Nachricht benötigte Prozessmodell instanziiert, d.h. der Prozess wird gestartet. Der Prozessserver dient fortan als Koordinator für die Ausführung dieser Prozessinstanz. Bei dieser Koordination werden Anwendungen und Dienste (engl. *services*) angesprochen und dadurch in einen größeren Ablauf integriert, man spricht auch von einer Dienstkomposition oder (Dienst-)Orchestrierung. Die Kommunikation zwischen dem Prozessserver und den Anwendungen und Diensten, die er orchestriert, wird über verschiedene Kommunikationswege und Protokolle realisiert. Die Gesamtheit dieser Kommunikationswege wird in der Literatur Enterprise Service Bus (ESB) genannt [Cha04].

Das (Instanz-)Monitoring verarbeitet die Nachrichten, die über den ESB ausgetauscht werden. Dazu gehören bspw. Events, die vom Prozessserver in Form von Nachrichten nach außen kommuniziert werden. Das Monitoring aggregiert die Informationen und stellt sie verschiedenen Interessengruppen in einer Monitoring-Umgebung für Prozesse bereit. So werden z.B. einem technischen Anwender Dienstgüteeigenschaften gezeigt (engl. *Quality of Service, QoS*), während einem wirtschaftlich ausgerichteten Anwender KPI zur Prozessausführung präsentiert werden. Die Monitoring-Umgebung verwendet eine Datenhaltung in der registrierte Events aufgezeichnet werden (engl. *event log*, *audit trail*). Die Aggregation einzelner Events kann außerdem dazu genutzt werden, den Ausführungszustand von Aktivitäten zu bestimmen. Weitere Sichtweisen der Infrastruktur sind bspw. bei Leymann und Roller [LR00], Weske [Wes07] und der Workflow Management Coalition [WFM95] zu finden.

2.1.3 Lebenszyklus von Geschäftsprozessen

Die verschiedenen Lebenszyklusphasen, die ein automatisierter Geschäftsprozess durchläuft, sind für das Verständnis von Geschäftsprozessen und damit verbundenen Technologien und Ansätzen elementar. Allerdings besteht in der Literatur kein Konsens darüber, welche Phasen ein Geschäftsprozess genau haben sollte und was der Zweck der jeweiligen Phasen ist. Zudem ist nicht ausnahmslos von Phasen die Rede, sondern auch von prozessbezogenen Aufgaben. Bspw. versteht Weske [Wes07] das Process Mining als Teil einer Evaluationsphase, während die Modellierung und Analyse eine gemeinsame Phase bilden. Das Verständnis des Prozesslebenszyklus bei Schumm et al. [SLS10b] sieht separate Aufgaben zur Analyse und Modellierung vor, stellt aber das Process Mining nicht explizit als Aufgabe dar.

Die verschiedenen Vorschläge zu den Lebenszyklusphasen unterscheiden sich in mehreren Details, wobei sich meist, bedingt durch die technische Infrastruktur zur Automatisierung von Geschäftsprozessen, drei elementare Phasen ausmachen lassen, wie bei Schumm et al. [SLS10b] festgehalten wurde: (1) die *Modellierungsphase*, in der ein Prozessmodell erstellt wird, (2) die *Ausführungsphase*, in der das Prozessmodell instanziiert wird und Dienste (bzw. Menschen und Maschinen) die im Prozessmodell festgelegten Arbeitsschritte erledigen sowie (3) die *Monitoringphase*, in der diese Ausführung kontrolliert wird.

In Anbetracht der unterschiedlichen Beschreibungsformen und Automatisierungsgraden von Geschäftsprozessen (vgl. Abschnitt 2.1.1), stellt die technische Realisierung von Prozessen eine sinnvolle Verfeinerung dar: In der *Implementierungsphase* wird ein Prozessmodell verfeinert und in eine ausführbare Prozesssprache überführt, getestet und in eine Ausführungsumgebung deployed. Darüber hinaus kann die auf laufende Prozesse ausgelegte Monitoringphase von der *Analysephase* unterschieden werden, die auf abgeschlossene Prozesse ausgerichtet ist.

Das resultierende Phasenverständnis des Lebenszyklus von Geschäftsprozessen ist in Abbildung 2.2 dargestellt.

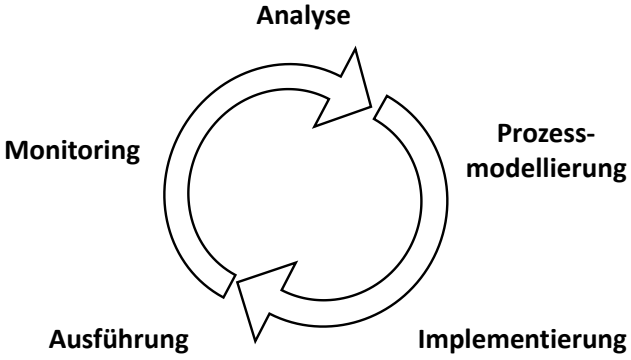


Abbildung 2.2: Phasenverständnis des Lebenszyklus von Geschäftsprozessen, modelliert nach [SLS10b].

2.1.4 Ein graphbasiertes Metamodell für Prozessskizzen

Das Metamodell eines Prozesses beschreibt seine Syntax, d.h. die zur Modellierung verfügbaren Sprachkonstrukte und Kombinationsregeln sowie die Eigenschaften und das Verhalten von Prozessen, die auf dieser Syntax aufbauen, d.h. deren Semantik [LR00].

Im konzeptionellen Teil dieser Arbeit werden grafische Prozessskizzen verwendet, um die Auswirkungen von Transformationen zu veranschaulichen. Diesen Prozessskizzen liegt ein graphbasiertes Metamodell zugrunde, das im Folgenden erläutert wird.

Als Metamodell für Prozessskizzen wird eine Modifikation des von Leymann und Roller [LR00] definierten PM-Graphen verwendet. Im Vergleich zum PM-Graphen sind in dem modifizierten Prozessmetamodell Zyklen im Kontrollflussgraphen erlaubt, welche bei der Anwendung einer Sicht entstehen können. Der Schwerpunkt des Metamodells wird auf die Syntax gelegt, die (Ausführungs-)Semantik ist weitgehend offen gelassen. Mithilfe des Metamodells können Prozessmodelle auf einer abstrakten Ebene skizziert werden. Definition 1 erläutert die Elemente des verwendeten Metamodells.

Definition 1: *Prozessmodell-Graph* (PM-Graph)

Ein Tupel $G = (A, D, B, K, \tau, i, o, \Phi)$ ist ein Prozessmodell-Graph oder kurz PM-Graph: $\Leftrightarrow$

1. A ist eine endliche Menge von Aktivitäten. Jede Aktivität $a \in A$ hat eine endliche Menge von Eigenschaften E, welche die Aktivität beschreiben. E beinhaltet elementare Eigenschaften wie $name \in Namen$ und $typ \in Typen$. Die Menge *Namen* ist die Menge aller zulässigen Namen. Die Menge *Typen* ist die Menge aller zulässigen Typen. *Typen = {invoke, receive, send, ...}*. Entsprechende Funktionen $n: A \rightarrow Namen$ und $t: A \rightarrow Typen$ weisen jeder Aktivität einen Namen und Typ zu.
2. D ist eine endliche Menge von Datenelementen. Jedem Datenelement $d \in D$ ist eine endliche Menge *DOM(d)* zugewiesen. Der Wertebereich *DOM* von Datenelementen d umfasst komplexe Geschäftsobjekte, die sich aus Schlüssel-Wert-Paaren von Standarddatentypen *{string, integer, float, ...}* zusammensetzen.
3. B ist eine endliche Menge an Bedingungen. Jede Bedingung $b \in B$ kann auf einen Wert *{wahr, falsch}* abgebildet werden.
4. $K \subseteq A \times A \times B$ ist eine endliche Menge von Kontrollverbindungen. K ist unifiziert, d.h. zwischen zwei Aktivitäten gibt es maximal eine Kontrollverbindung.
5. $\tau: K \rightarrow B$ weist jeder Kontrollverbindung eine Übergangsbedingung zu.
6. $i: A \rightarrow D$ weist jeder Aktivität ein eingehendes Datenelement zu.
7. $o: A \rightarrow D$ weist jeder Aktivität ein ausgehendes Datenelement zu.
8. $\Phi: A \rightarrow B$ weist jeder Aktivität eine Join-Bedingung zu.

Ein PM-Graph besteht aus einer Menge an Aktivitäten, deren Kontrollabhängigkeit durch gerichtete Kontrollverbindungen festgelegt ist. Datenabhängigkeiten zwischen Aktivitäten sind durch eingehende Datenelemente (lesend) und ausgehende Datenelemente (schreibend) implizit beschrieben. Eine Übergangsbedingung wird ausgewertet, sobald die vorangehende

Aktivität beendet wurde. Sie legt den Wert *{wahr, falsch}* einer Kontrollverbindung fest. Eine Join-Bedingung einer Aktivität bestimmt, ob diese ausgeführt oder übersprungen werden soll. Die Join-Bedingung wird ausgewertet, sobald die Übergangsbedingungen aller eingehenden Kontrollverbindungen ausgewertet sind. Eine Join-Bedingung ist eine Funktion, die die Übergangsbedingungen aller in die Aktivität eingehenden Kontrollverbindungen auf einen Wert *{wahr, falsch}* abbildet. Eine Aktivität wird ausgeführt, wenn seine Join-Bedingung zu *wahr* ausgewertet wurde. Sobald eine Join-Bedingung zu *falsch* ausgewertet wurde, wird eine Eliminierung toter Pfade durchgeführt (engl. *dead path elimination*), um ewige Wartezustände zu vermeiden [LR00]. Bei diesem Vorgang werden die Übergangsbedingungen entlang eines Pfades zu *falsch* evaluiert, bis eine Aktivität erreicht wird, deren Join-Bedingung zu *wahr* ausgewertet werden kann oder bei der noch nicht alle eingehenden Kontrollverbindungen ausgewertet sind.

Ein PM-Graph repräsentiert das Modell eines Prozesses, d.h. Informationen über eine konkrete Instanz dieses Modells sind in dem PM-Graphen nicht enthalten. Der Prozessinstanz-Graph oder kurz PI-Graph erweitert den PM-Graphen um entsprechende Eigenschaften zur Erfassung von Laufzeitinformationen.

Definition 2: *Prozessinstanz-Graph* (PI-Graph)

Ein Tupel $I = (A, D, B, K, \tau, i, o, \Phi, Z_a, \alpha, z)$ ist ein Prozessinstanz-Graph oder kurz PI-Graph: $\Leftrightarrow$

1. $A, D, B, K, \tau, i, o, \Phi$ entsprechen der jeweiligen Definition im PM-Graphen.
2. Z_a ist eine endliche Menge an Zuständen, wobei Z_a = *{Inaktiv, Bereit, Aktiv, Unterbrochen, Erledigt, Fehlerhaft, Übersprungen}*.
3. $\alpha: A \rightarrow Z_a$ weist jeder Aktivität einen Zustand zu.
4. z gibt den Zustand des Prozesses an. $z \in Z_p$, wobei Z_p = *{Bereit, Aktiv, Erledigt, Fehlerhaft}*.

Das Aktivitätszustandsmodell für PI-Graphen ist in Abbildung 2.3 dargestellt. Es ist vom Verständnis des Aktivitätslebenszyklus nach Karastoyanova et al. [KKS+06] geprägt, übernimmt diesen jedoch nur in Teilen. Bspw. wurden Zustandsübergangsbedingungen, der Wartezustand und mehrere Zustandsübergänge ausgelassen und ein Unterbrechungszustand hinzugefügt.

Das Aktivitätszustandsmodell liest sich wie folgt: Initial ist eine Aktivität *Inaktiv*. Nachdem die Vorgängeraktivität beendet wurde (*Erledigt* oder *Fehlerhaft*), wird die Übergangsbedingung auf der Kontrollverbindung ausgewertet. Sind alle eingehenden Kontrollverbindungen ausgewertet, wird die Join-Bedingung der Aktivität ausgewertet. Sollte diese zu *falsch* ausgewertet werden, wird die Aktivität nicht ausgeführt (*Übersprungen*). Andernfalls wird die Aktivität als *Bereit* gekennzeichnet. In diesem Zustand kann sie von einem Anwender zur Bearbeitung angefordert werden oder deren automatische Ausführung erfolgen (z.B. ein Programmaufruf). Sobald die Durchführung der Aktivität begonnen wurde, ist sie *Aktiv*. Die Durchführung kann pausiert werden (*Unterbrochen*) und anschließend wiederaufgenommen werden (*Aktiv*). Nach erfolgreichem Abschluss der Aktivität ist diese *Erledigt*.

Sollten bei der Bearbeitung oder bei der automatischen Ausführung Fehler wie bspw. ein Programmabbruch aufgetreten sein, ist der Zustand der Aktivität *Fehlerhaft*.

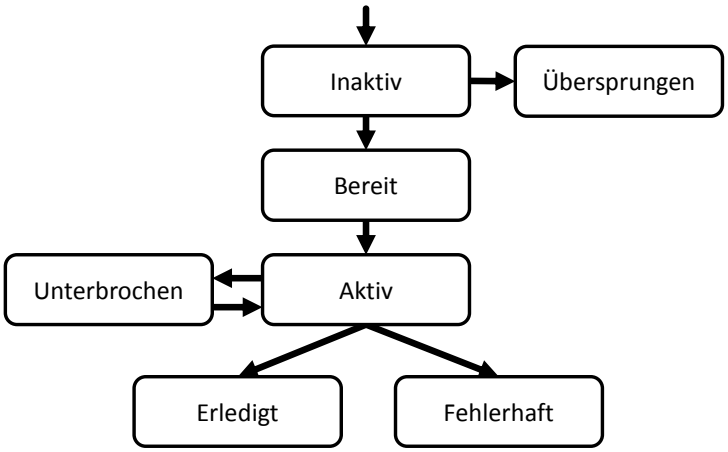


Abbildung 2.3: Aktivitätszustandsmodell von PI-Graphen, vgl. [KKS+06].

Das Zustandsmodell von Prozessinstanzen ist in Abbildung 2.4 dargestellt. Der initiale Gesamtzustand einer Prozessinstanz ist *Bereit*. Sobald die erste Aktivität in den Zustand *Bereit* versetzt wurde, ist die Prozessinstanz *Aktiv*. Wenn keine weiteren Aktivitäten zur Ausführung mehr anstehen ist die Prozessinstanz *Erledigt*. Sind während der Ausführung Fehler aufgetreten ist die Prozessinstanz *Fehlerhaft*.

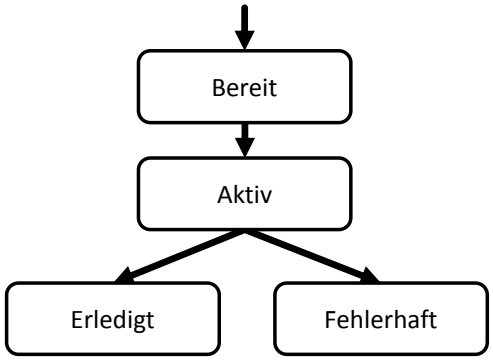


Abbildung 2.4: Prozessinstanzzustandsmodell von PI-Graphen, vgl. [KKS+06].

PM-Graphen und PI-Graphen werden in dieser Arbeit grafisch dargestellt. Die hierfür verwendete Notation wird im Folgenden erläutert. In Abbildung 2.5 sind die Symbole zur Darstellung von PM-Graphen abgebildet, welche sich am Standard BPMN orientieren, vgl. [OMG11b]. Eine Aktivität wird mit einem abgerundeten Rechteck dargestellt; eine Aktivität kann eine Beschriftung tragen; ein Datenelement wird durch ein Blatt Papier mit eingeknickter Ecke repräsentiert; Kontrollverbindungen werden durch einen Pfeil mit ausgefüllter Spitze dargestellt; Bedingungen auf Kontrollverbindungen werden oberhalb des Pfeils platziert; Datenfluss wird mit Pfeilen mit gestrichelter Linie dargestellt.

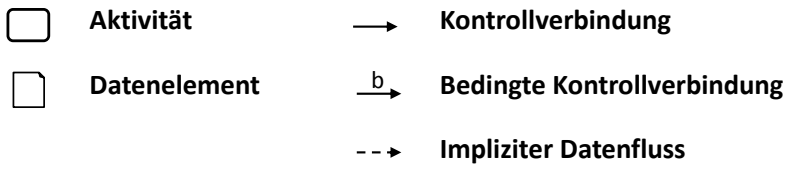


Abbildung 2.5: Grafische Notation zur Darstellung von PM-Graphen, orientiert an BPMN [OMG11b].

Zur grafischen Repräsentation des Zustands von Aktivitäten in PI-Graphen werden die in Abbildung 2.6 gezeigten Aktivitätszustandssymbole verwendet.

Abbildung 2.6: Aktivitätszustandssymbole für die Darstellung von PI-Graphen.

In Abbildung 2.7 wird ein Beispiel eines PI-Graphen gezeigt. Die Symbole zur Darstellung des Ausführungszustands von Aktivitäten werden entweder direkt in die Form der Aktivität gezeichnet oder daran annotiert, um so die Aktivitätsbeschriftung in der Darstellung zu erhalten. Der abgebildete PI-Graph beinhaltet vier Aktivitäten von denen zwei bereits erledigt wurden, eine Aktivität wurde aufgrund der Übergangsbedingung *b* übersprungen und die letzte Aktivität befindet sich in der Ausführung.

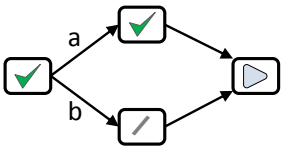


Abbildung 2.7: Beispiel eines PI-Graphen.

Das vorgestellte Metamodell für Prozesse, Prozessinstanzen und deren Darstellung liefert eine Grundlage, welche im weiteren Verlauf der Arbeit erweitert wird, um die entwickelten Ansätze zur Reduzierung der Komplexität von Prozessen skizzieren zu können.

2.1.5 Komplexität von Geschäftsprozessen

Eine mögliche Herangehensweise, um den Begriff der Komplexität auf dem Gebiet von Geschäftsprozessen einzugrenzen, besteht in der Aufstellung einer Formel zur Berechnung der Komplexität, welche strukturelle Eigenschaften eines Geschäftsprozesses einbezieht. Zu diesen Eigenschaften gehören in erster Linie, ausgehend von der Definition des PM-Graphen, die Anzahl der in einem Prozess *P* enthaltenen Datenelemente *D*, Aktivitäten *A*, Kontrollverbindungen *K* und Bedingungen *B*. Eine Formel zur Bestimmung der Komplexität *O(P)* kann bspw. die Anzahl der jeweiligen Elemente aufsummieren und der jeweiligen Kategorie von Elementen eine Gewichtung verleihen.

Somit ergibt sich eine Formel:

$$O(P) = \alpha \cdot |D| + \beta \cdot |A| + \gamma \cdot |K| + \delta \cdot |B|$$

Diese Formel kann in verschiedener Hinsicht erweitert und verfeinert werden, etwa durch Differenzierung der Komplexität unterschiedlicher Typen von Aktivitäten. Verwandte Arbeiten zur Bestimmung der Komplexität von Prozessmodellen wählen zum Teil ein ähnliches Vorgehen. Gruhn und Laue [GL06] argumentieren, dass die Anzahl von Aktivitäten als Komplexitätsmetrik mit der in der Softwareentwicklung gängigen Metrik der Anzahl an Codezeilen ein möglicher Vergleich sei. Sie messen die Komplexität des Kontrollflusses durch die Anzahl an bedingten Kontrollverbindungen. Weitere Metriken, die schwieriger messbar sind, bestehen etwa (1) in dem Grad an Modularisierung, (2) in dem Enthaltensein von bestimmten Prozessstrukturen oder Konstrukten und (3) in dem kognitiven Aufwand der zum Prozessverständnis erforderlich ist. Letztere Metrik hängt nicht nur mit den strukturellen Eigenschaften eines Geschäftsprozesses, sondern auch mit dessen fachlicher Semantik und seiner Darstellung zusammen. Cardoso et al. [CMN+06] kamen teils zu vergleichbaren Ergebnissen, was den ersten Schluss nahelegt, das zumindest die Anzahl enthaltener Elemente eine elementare Metrik zur Komplexitätsmessung darstellt. Die weiterführenden Metriken stellen eine größere Herausforderung bzgl. ihrer Erfassung und Verwendung dar.

Der Begriff der Komplexität mitsamt den gegebenen Möglichkeiten diese zu reduzieren, wird nachfolgend durch eine Beschreibung und Analyse bestehender Ansätze geschärft.

Klassische Konzepte. Ein Vergleich vom heutigen Verständnis automatisierter Geschäftsprozesse mit der Entwicklung von Software wurde schon früh gezogen: DeRemer und Kron [DK75] vergleichen klassische Programmiersprachen mit Sprachen, die zur Integration einzelner Module dienen, dem Programmieren im Großen. In dieser Arbeit wurde vorgeschlagen, klassische Konzepte der Informatik wie Modularisierung und das Information Hiding auch auf diese Form der Programmierung zu übertragen. Modularisierung ist in der Softwareentwicklung dafür bekannt, sich positiv auf die Verständlichkeit eines Systems auszuwirken [Par72]. Dieser Umstand wurde auch in verschiedenen Arbeiten zur Komplexität von Prozessen festgestellt. Eine weitere bewährte Möglichkeit zur Bewältigung von Komplexität besteht in der Trennung von Zuständigkeiten (engl. *separation of concerns*), um nicht mit allen Aspekten eines Problems gleichzeitig konfrontiert zu sein [Dij82]. Die Trennung von Zuständigkeiten wird bspw. durch Schichten in Anwendungssystemen genutzt, um Darstellung, Geschäftslogik und Datenhaltung voneinander zu trennen (engl. *three-tier architecture*). Die Trennung von Zuständigkeiten ähnelt dem „teile und herrsche“-Prinzip (engl. *divide and conquer*), welches darin besteht, ein Problem in kleinere Teile zu zerlegen, diese isoliert zu lösen und die Teillösungen zu einer Gesamtlösung zusammenzuführen.

Die Modularisierung hat mit dem Konzept *Sub-Prozess* bereits deutlich Einfluss auf verschiedene Prozesssprachen genommen. Unter einem Sub-Prozess versteht man einen Prozess, der von einem anderen Prozess oder Sub-Prozess ausgeführt oder aufgerufen wird und der einen Teil des gesamten Prozesses darstellt [WFM99]. Nach Leymann und Roller [LR00] unterscheidet man verschiedene Formen von Sub-Prozessen, je nach Autonomiegrad, Form

des Aufrufs und Ort der Prozessdefinition und der Prozessausführung. Kopp et al. [KEL+10] beschreiben das gesamte Spektrum der verschiedenen Ausprägungen von Sub-Prozessen. Hier sollte festgehalten werden, dass Sub-Prozesse eine etablierte Methode zur Reduzierung der Komplexität von Geschäftsprozessen darstellen, die sich jedoch dauerhaft auf die Prozessstruktur auswirkt und eine festgelegte Ausführungssemantik erfordert.

Die Trennung von Zuständigkeiten wurde mit der Entstehung von Konzepten zu Sichten zunehmend auch für Geschäftsprozesse erschlossen. Verschiedene Hersteller von Lösungen zur Modellierung und Analyse von Geschäftsprozessen setzen diese Methode auf vielfältige Art und Weise ein. In dem Architekturkonzept ARIS (Architektur integrierter Informationssysteme) [Sof11] werden bspw. die Beschreibungssichten Organisation, Daten, Prozesse, Funktionen und Leistung voneinander getrennt. Auf Sichten wird in Abschnitt 2.2 eingegangen.

Modellierungsrichtlinien. Gruhn und Laue haben angeführt, dass bestimmte Konstrukte und Strukturen für das Verständnis von Prozessen vorteilhaft oder unvorteilhaft sind [GL06]. Kenntnisse über derartige Konstrukte können den Prozessmodellierer darin unterstützen, gut verständliche Prozesse zu modellieren. Analog zur Programmierung können umständliche oder komplizierte Konstrukte ggf. durch einfachere oder elegantere ersetzt werden, wodurch die ursprüngliche Semantik erhalten bleibt, das Prozessmodell aber leichter verständlich wird. Mendling et al. [MRA10] haben eine Reihe an Modellierungsrichtlinien auf Basis empirischer Untersuchungen vorgestellt, mit denen gut verständliche Prozessmodelle erstellt werden können. Eine Richtlinie besteht bspw. darin, Aktivitäten im Englischen durchgängig nach dem Schema Verb-Substantiv zu bezeichnen. Dies entspricht der Konvention Substantiv-Verb im Deutschen [Bun12]. Eine weitere Richtlinie schlägt vor, Prozessmodelle klein zu halten (< 50 Aktivitäten) und Teilstrukturen in Sub-Prozesse auszulagern [MRA10]. Eine weitere bewährte Vorgehensweise zur Vereinfachung von Prozessen und der Modellierung besteht laut Silver [Sil09] darin, den Sprachumfang einer Prozesssprache je nach Nutzergruppe einzuschränken (engl. *language subsets*). Spezielle Sprachelemente, die besonderes Expertenwissen erfordern, können so vor bestimmten Nutzergruppen verborgen werden.

Kognitiver Aufwand. Die Komplexität eines Prozesses ist nicht ausschließlich an den enthaltenen Elementen und deren Verbindungen festzumachen. Auch die Darstellung eines Prozesses spielt für das Verständnis und den dafür erforderlichen kognitiven Aufwand eine wichtige Rolle, wie Moody [Moo09] beschreibt. Nach Moody spielt neben vielen anderen Faktoren nicht nur das Layout eine Rolle, sondern auch nach welchen Gesichtspunkten die grafischen Elemente einer (Prozess-)Sprache oder einer grafischen Notation gestaltet sind. Nach Moody berücksichtigen bestehende Notationen nur selten wahrnehmungstheoretische Erkenntnisse. Wie Petre [Pet95] feststellt, benötigen unterschiedliche Gruppen von Betrachtern unterschiedliche Symbole und Formen; ein Symbol setzt implizit bestimmte Vorkenntnisse beim Betrachter voraus. Die grafische Darstellung stellt folglich einen Aspekt dar, der zum Verständnis eines Geschäftsprozesses beiträgt.

2.2 Stand der Technik von Sichten

In diesem Abschnitt wird der Stand der Technik von Sichten vorgestellt. Zuerst wird der Begriff von Sichten in der Softwareentwicklung untersucht. Danach werden Sichten auf Prozesse (kurz Prozesssichten) betrachtet, wobei zwischen herkömmlichen und sprachübergreifenden Sichten unterschieden wird.

2.2.1 Sichten in der Softwareentwicklung

Das Konzept von Sichten hat in verschiedenen Bereichen der Softwareentwicklung dazu beigetragen, mit zunehmender Komplexität effizient umgehen zu können. Besonders in der Datenbanktechnik und in der Softwarearchitektur wurden Ansätze zu Sichten entwickelt.

Sichten auf Datenbanken. Die ständig steigende Komplexität von Softwaresystemen und die zunehmende Menge an Daten, die sie verarbeiten müssen, haben verschiedene Methoden und Prinzipien zur Komplexitätsreduzierung hervorgebracht. Die Trennung des Programmcodes von den Speicherstrukturen der zu verarbeitenden Daten war ein zentraler Schritt hin zur Entwicklung von Datenverwaltungssystemen, den Database Management Systems (DBMS). Diese Systeme bezwecken die effiziente Speicherung von Daten und bieten spezielle Abfragesprachen wie etwa die Structured Query Language (SQL) an.

Sichten auf Datenbanken waren und sind noch immer ein wichtiges Konzept, um die Komplexität aus Anwendersicht zu reduzieren und eine hohe Effizienz trotz großen Umfangs gespeicherter Datenmengen zu gewährleisten. Eine Sicht bietet eine für einen bestimmten Zweck oder eine Anwendergruppe zugeschnittene Perspektive auf die in einer Datenbank enthaltenen Daten. Zudem kann eine Sicht Daten in einer anderen Struktur zeigen als sie tatsächlich in der Datenbank verwendet wird. Eine Eigenschaft einer Sicht auf eine Datenbank ist, dass sie Informationen verbirgt, die in einem bestimmten Kontext nicht von Bedeutung sind. Die folgende Darstellung der Grundlagen zu Sichten auf Datenbanken basiert auf dem Werk Foundations of Databases von Abiteboul et al. [AHV95]. Abbildung 2.8 gibt einen Überblick über die grundlegenden Aspekte dieser Sichten.

Sichten werden i.d.R. durch eine Reihe von Anfragen definiert, welche an die logische Ebene einer Datenbank gestellt werden. Die logische Ebene repräsentiert eine abstrakte, auf den Menschen ausgerichtete Datenrepräsentation, wohingegen die physische Ebene der effizienten Datenspeicherung dient. Die Anfrageoperatoren zur Definition einer Sicht erlauben bspw. die Selektion bestimmter Daten, die Aggregation von Daten, die Manipulation und auch eine Kombination. Eine Sicht kann außerdem als Zwischenergebnis zur weiteren Verfeinerung verwendet werden.

Man unterscheidet zwischen materialisierten und virtuellen Sichten. Bei einer materialisierten Sicht wird eine physische Kopie der in der Sicht vorhandenen Daten angelegt. Eine materialisierte Sicht muss nach Änderungen von ihr zugrunde liegenden Daten teilweise oder vollständig neu berechnet werden. Eine Schwierigkeit besteht hier zu bestimmen, ob sich eine Änderung an grundlegenden Daten auf eine Sicht auswirkt und wenn ja, ob nur ein Teil einer Sicht neu berechnet werden muss, oder ob eine vollständige Neuberechnung erforderlich ist.

Bei der Verarbeitung großer Datenmengen stellt die inkrementelle Neuberechnung eine bessere Performance in Aussicht als eine vollständige Neuberechnung. Virtuelle Sichten hingegen werden auf Anforderung erstellt und erfordern daher keine Instandhaltungsmaßnahmen.

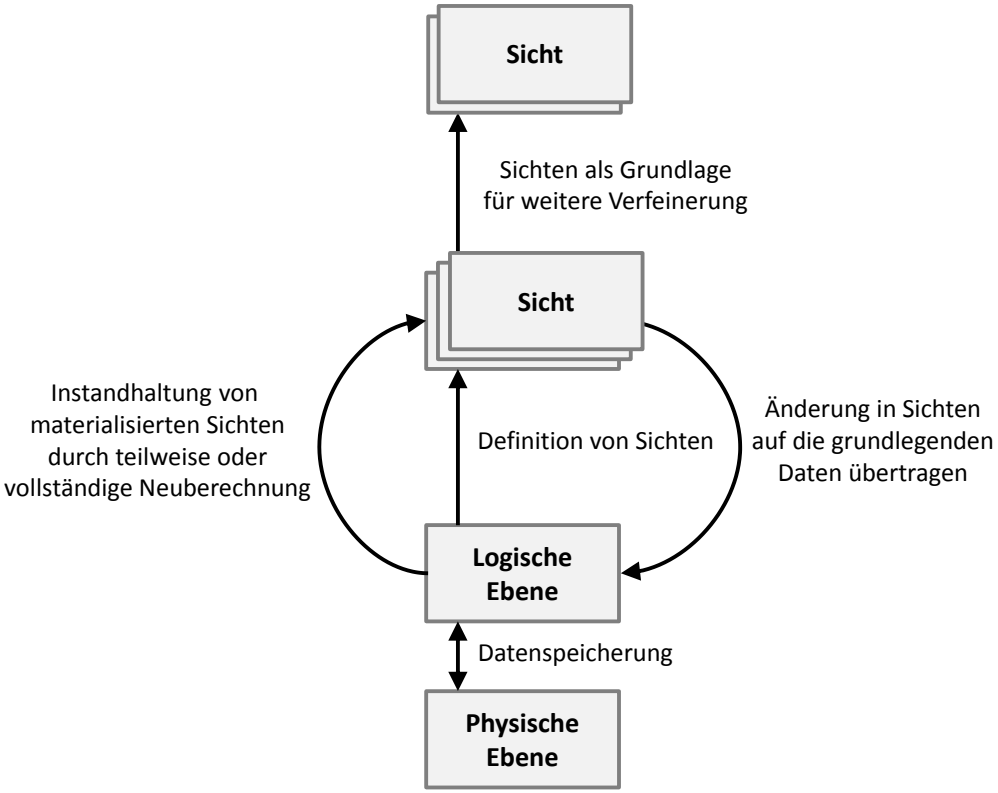


Abbildung 2.8: Grundlagen von Sichten auf Datenbanken, modelliert nach [AHV95].

Sichten sind, wie Abiteboul et al. beschreiben, von Natur aus unvollständig. Gerade diese Eigenschaft ist eine Stärke zur Komplexitätsbewältigung, allerdings bewirkt sie Probleme was die Übertragung von in Sichten vorgenommenen Änderungen auf die ihr zugrunde liegenden Daten angeht. Dieser Umstand ist als *View Update Problem* bekannt. Wenn Änderungen in einer Sicht vorgenommen werden, ist deren Übertragung auf zugrunde liegende Daten u.U. nicht eindeutig, d.h. die Änderungsoperation ist im Hinblick auf die vollständige Datenbasis unterspezifiziert. Eine Möglichkeit, um Mehrdeutigkeiten zu berücksichtigen, aber dennoch Änderungen in einer Sicht zu erlauben, besteht in der Entwicklung von Algorithmen zur Auflösung von Mehrdeutigkeiten (engl. *update translations*), welche die Semantik der Datenbank und die Semantik der Sicht einbeziehen. Constraints auf Sichten können verwendet werden um festzulegen, welche Änderungen zulässig sind. Auch können Sichten nur für Lesezwecke erstellt werden (engl. *read-only*). Abiteboul et al. beschreiben möglicherweise auftretende Seiteneffekte als eine weitere Schwierigkeit bei in Sichten vorgenommenen Änderungen. Eine in einer Sicht vorgenommene Löschoperation kann bspw. unerwartete Änderungen wie das Löschen weiterer, in der Sicht aber ausgeblendeter Elemente bewirken. Je komplexer eine Sicht aufgebaut ist, desto schwieriger ist es, eine Manipulation der grundlegenden Daten in der Sicht zu ermöglichen.

Sichten in der Softwarearchitektur. Die in der Softwareentwicklung etablierten Techniken zum Umgang mit Komplexität wie Modularisierung, Information Hiding und die Trennung von Zuständigkeiten (vgl. Abschnitt 2.1.5) sind auch in der Architektur von Software zu finden. In UML [OMG11a] wurden mehrere Diagrammtypen spezifiziert, die verschiedene Angelegenheiten in der Softwarespezifikation betreffen. In UML gibt es einen Diagrammtyp zur Modellierung von Anwendungsfällen, Klassenstrukturen, Zustandsautomaten, der Verteilung von Komponenten als auch einen Diagrammtyp zur Beschreibung von Prozessen (UML AD). Man kann ein Softwaresystem durch mehrere Diagramme verschiedenen Typs beschreiben, die jeweils eine bestimmte Angelegenheit, Perspektive oder auch Sicht repräsentieren. Ein Teil der Sichten zeigt strukturelle Zusammenhänge einer Software wie z.B. Klassen, ein anderer Teil zeigt dynamische Aspekte wie bspw. Abläufe. Im Gegensatz zu Sichten auf Datenbanken werden diese Arten von Sichten jedoch nicht auf einer einzigen, grundlegenden Informationsbasis durch eine Anfragesprache automatisch erzeugt, sondern jede Sicht wird individuell erstellt und manuell gewartet. Querbezüge zwischen unterschiedlichen Sichten werden manuell oder teilautomatisch verwaltet. Die Erhaltung der Konsistenz zwischen Sichten ist i.d.R. eine vom Modellierer zu erfüllende Aufgabe.

ACME [GMW97] ist ein Beispiel einer reinen Architekturbeschreibungssprache (engl. *Architecture Description Language, ADL*). Mit ACME können Diagramme modelliert werden, die aus Komponenten bestehen, deren Ports mit Konnektoren zu einem System verbunden werden. Durch die Verwendung von Vorlagen soll die Komplexität der Modellierung neuer Systeme verringert werden. Obgleich ACME bereits auf einem hohen Abstraktionsgrad ansetzt, bietet die Sprache neben der Modularisierung keine weiteren Techniken, um in der Sprache beschriebene Modelle dynamisch zu abstrahieren. Dieser Umstand findet sich auch in anderen ADL wieder. Die von Shaw et al. [SDK+95] vorgestellte Notation sieht bspw. als Abstraktionsmechanismus auf bereits erstellte Modelle vor, Komponenten in Paketen zusammenzufassen. Weitere Formen der Abstraktion müssen von dem Architekten manuell vorgenommen werden.

Wenn eine Sprache zur Beschreibung von Softwarearchitekturen keine oder nur geringfügige Mittel zur Abstraktion bietet, ist es letztlich der Modellierer oder der Architekt eines Systems, der selbst Abstraktionen vornimmt und auf diese Weise Sichten von unterschiedlichem Detailgrad erstellt. Selic [Sel11] hat eine Reihe von in der Praxis bewährten Vorgehensweisen beschrieben, die von Architekten häufig zur Abstraktion von Softwaremodellen eingesetzt werden (engl. *abstraction patterns*). *Black Box* ist wohl das bekannteste Muster. Es beschreibt die Abstraktion einer komplexen Struktur von verbundenen Komponenten zu einer einzigen Komponente, die von den Details der darin enthaltenen Elemente abstrahiert. Eine *Black Line* abstrahiert eine Struktur zusammenhängender Komponenten in ein Verbindungselement, z.B. um einen Kommunikationskanal zu repräsentieren. *Summary Message* bezeichnet die Aggregation mehrerer Nachrichten. *Summary State* fasst mehrere feingranulare Zustände zu einem abstrakteren Zustand zusammen. *Summary Activity* aggregiert Aktivitäten, Kontrollverbindungen und Datenverbindungen zu einer komplexen Aktivität.

Der Ansatz, Abstraktionskonzepte in einfach verständlichen Mustern zu beschreiben und so für verschiedene Anwendungsfelder zu erschließen, gewinnt generell an Interesse. De Lara et

al. [LGC12] diskutieren die Aggregation und das Entfernen von Strukturen als allgemeine Techniken zur Modellabstraktion im Kontext modellgetriebener Entwicklung. Sie verfolgen den Ansatz, die Abstraktionstechniken über verschiedene Sprachen hinweg wiederverwendbar zu machen. Dieser Ansatz wird ebenso in der vorliegenden Dissertation verfolgt, jedoch mit dem Fokus auf Prozesssprachen, vgl. dazu Schumm et al. [SLS10a].

Sichten in der Unternehmensarchitektur. In der Architektur, Entwicklung und Vernetzung von Unternehmensanwendungen müssen die Anforderungen verschiedener Interessengruppen berücksichtigt werden. In diesem Zusammenhang werden sogenannte Viewpoints dazu verwendet, um verschiedene Sichtweisen auf einen Problembereich zu erfassen [DS96]. In einem Viewpoint können Anwendungen im übergeordneten Unternehmenskontext betrachtet werden, somit ist die Betrachtung eines Ausschnitts der Unternehmensinfrastruktur (Enterprise Architecture, EA) möglich. Die verschiedenen Interessengruppen können bspw. an einer technischen, einer prozessualen oder auch einer strategischen Perspektive interessiert sein [BEL+07]. Der Ansatz von Viewpoints setzt das Prinzip der Trennung von Zuständigkeiten ein, um die hohe Komplexität von Unternehmensinfrastrukturen auf die für eine bestimmte Aufgabe oder Interessengruppe erforderliche Menge an Informationen zu reduzieren. Buckl et al. [BEL+08] bringen EA-Viewpoints außerdem in Verbindung mit dem Ansatz, bewährte Lösungen in Form von Mustern zu beschreiben und in einem Musterkatalog zusammenzustellen. Buckl et al. haben Wert darauf gelegt, die Beschreibung implementierungsunabhängig und allgemein zu halten, um die beschriebene Lösung leichter in anderen, aber ähnlichen Sachverhalten wiederverwenden zu können.

Werkzeuggestützte Abstraktion in der Modellierung. Das im Bereich der Modellierung mit UML gängige Werkzeug Enterprise Architect 10 [Spa13] bietet mit Diagrammfiltern eine Möglichkeit, Modellelemente auszublenden (engl. *hide*), zu verblassen (engl. *fade*) oder auszugrauen (engl. *gray scale*). Durch Anwendung eines oder mehrerer Filter kann die Komplexität eines Diagramms temporär reduziert werden. Die Selektion der zu filternden Modellelemente kann über Vergleichsoperatoren gesteuert werden, welche auf die Standardeigenschaften von Elementen anwendbar sind (engl. *properties*). Knoten und Kanten eines Modells können separat angesprochen werden. Transitive Verbindungen zwischen Knoten werden beim Ausblenden von Knoten eines Verbindungspfades nicht dargestellt. Filter können unabhängig vom Modell definiert und so flexibel wiederverwendet werden. Bspw. kann ein Filter definiert werden, mit dem alle Elemente ausgeblendet werden, die für die Eigenschaft *Status* den Wert *nicht freigegeben* tragen. Eine Sicht kann temporär als Diagrammkopie erzeugt werden. Änderungen an Elementeigenschaften, die in einer Sicht vorgenommen werden können auf das Ursprungselement übertragen werden, d.h. es besteht eine eingeschränkte Manipulierbarkeit der Daten in einer Sicht.

Benutzersicht. Der Entwurf der Benutzerschnittstelle einer Anwendung stellt einen weiteren Ansatzpunkt zur Reduzierung der Komplexität dar. Nielsen [Nie93] beschreibt verschiedene Richtlinien und Empfehlungen für den ergonomischen Entwurf grafischer Benutzerschnittstellen (engl. *Graphical User Interface, GUI*). Er charakterisiert zehn Richtlinien als wesentlich. Die folgenden Fünf, frei übersetzt aus dem Englischen, lassen sich durchaus auf den Bereich der Komplexität von Geschäftsprozessen übertragen:

1. Einfache und natürliche Dialoge: Dialoge sollten keine Informationen enthalten, die nicht relevant sind oder nur selten benötigt werden. Jede zusätzliche Informationseinheit steht in Konkurrenz zu den wirklich relevanten Informationen und verringert deren relative Sichtbarkeit, vgl. [Nie93].
2. Sprich die Sprache des Benutzers: Dialoge sollten mit klaren Worten verfasst sein und Formulierungen und Konzepte verwenden, mit denen die Benutzer vertraut sind, vgl. [Nie93].
3. Minimiere die Gedächtnisbelastung: Der Benutzer sollte sich keine Informationen von einem Dialog bis zu einem anderen Dialog merken müssen, vgl. [Nie93].
4. Konsistenz: Dieselbe Information sollte in unterschiedlichen Dialogen auf dieselbe Art und Weise und am selben Ort erscheinen, vgl. [Nie93].
5. Abkürzungen: Mit Abkürzungen und Beschleunigern, die für einen Neuling nicht sichtbar sind, ist ein System sowohl für ungeübte als auch für geübte Nutzer passend, vgl. [Nie93].

Als Erkenntnis kann festgehalten werden, dass verschiedene Sichten auf ein System durch unterschiedliche Notationen und Diagrammarten unterstützt werden können. Außerdem können Diagramme manuell oder werkzeuggestützt unter Einsatz geeigneter Techniken abstrahiert werden. Daten können in erzeugten Sichten ggf. nur eingeschränkt geändert werden.

2.2.2 Sichten auf Geschäftsprozesse

Der Begriff einer Sicht auf einen Prozess wird in der Literatur nicht einheitlich verwendet. Während sich ein Teil der Arbeiten auf die reine Abstraktion der Struktur von Prozessmodellen beschränkt, beziehen andere Arbeiten die grafische Darstellung als Bestandteil einer Sicht mit ein. In beiden Ausrichtungen wurden zahlreiche Prozesssprachen untersucht, um für diese Sprachen Sichten zu definieren und automatisch zu erzeugen. Zur Erzeugung von Sichten sind Ansätze aus dem Gebiet der Modelltransformation relevant. Diese Ansätze können als Rahmenwerk zur Definition und Durchführung von strukturellen Änderungen an Prozessen dienen. Grafische Werkzeuge zur Modellierung und Analyse von Prozessen setzen Sichten im Sinne der Trennung von Zuständigkeiten ein, um die Menge an Informationen und Aspekten, die in einem Prozessmodell enthalten sind, zu strukturieren und dem Anwender einfach zugänglich zu machen. Nachfolgend werden diese verwandten Arbeiten vorgestellt.

Abstraktionstechniken. Abstraktion stellt einen wesentlichen Bestandteil des menschlichen Erkenntnisprozesses dar, vgl. Hirschberger [Hir91]. Nach Hirschberger wird Abstraktion in der Philosophie häufig dadurch umschrieben, dass das Wesentliche hervorgehoben wird und vom Unwesentlichen abgesehen wird. Die Schwierigkeit der Abstraktion liege darin, das Wesentliche und das Unwesentliche als solches zu erkennen. Eine weitere Schwierigkeit bestehe darin, dass das Wichtige bei der Abstraktion bewahrt wird und ein Sachverhalt nicht zu sehr abstrahiert wird, z.B. durch Übergeneralisierung oder Idealisierung.

Diese Erkenntnisse spiegeln sich auch in der Entwicklung von Techniken zur strukturellen Abstraktion von Prozessen wider (engl. *process abstraction*). Häufig stützen sich diese Tech-

niken auf Graphentheorie, wie bspw. die von Sadiq und Orlowska [SO00] vorgestellte Arbeit zur Prozessabstraktion. Wie Abello et al. [AHK06] zeigen, existieren unterdessen verschiedene Techniken zur Graphabstraktion. Abello et al. schlagen die Bildung einer Hierarchie aus Knoten durch den Einsatz einer Sequenz verschiedener Gruppierungsalgorithmen vor. Bspw. wird *Peeling* zur Erkennung und Gruppierung von Teilbäumen im Graphen verwendet; *Biconnected Components* betrifft die Gruppierung von Teilgraphen, die nicht zerfallen wenn eine Kante davon entfernt wird. Wie Abello et al. feststellen, ist die Beschriftung der gebildeten Hierarchien ein wichtiger Aspekt für den zwar mehrere Ansätze existieren, aber für den auch heute noch keine abschließende Lösung gefunden ist. Dieser Aspekt ist auch unter der Bezeichnung *Labeling Problem* bekannt.

Die Änderung, oder auch *Transformation*, die auf einen Prozess im Zuge der Abstraktion angewendet wird, besteht meist aus dem *Entfernen* von Aktivitäten und Datenelementen sowie deren Eigenschaften, Kontrollverbindungen und Datenverbindungen (engl. *omission*, auch: *abstraction*). Alleine durch das Entfernen von Strukturen aus einem Prozess können verschiedene Anwendungsfälle unterstützt werden, bspw. die Vereinfachung von Softwareprozessen durch das Auslassen von bestimmten Prozessstrukturen [DGW94].

Die *Aggregation* von Strukturen ist eine ebenfalls häufig genutzte Technik, um ein detailliertes Prozessmodell in ein abstrakteres Modell zu überführen. Die Anwendung der Aggregation ist eine Form der Modularisierung, die sich von der Modularisierung durch Sub-Prozesse deutlich unterscheidet: Eine Aggregation in einer Sicht ist i.d.R. flüchtig, das Prozessmodell und dessen Darstellung werden nur vorübergehend modifiziert, wohingegen eine Modularisierung durch einen Sub-Prozess eine dauerhafte Anpassung eines Prozesses darstellt. Zudem fasst eine Aggregation mitunter keine in sich abgeschlossenen Prozessstrukturen zusammen, sondern eine von einem Algorithmus als zusammengehörig bestimmte, nicht notwendigerweise zusammenhängende Menge von Prozesselementen. Transformationen zum Entfernen und Aggregieren von Strukturen eines Prozessgraphen wurden in zahlreichen Arbeiten formal beschrieben und können als gegeben betrachtet werden [LS03][Bob08] [EG08][Pol12][Smi12]. An dieser Stelle sollte angemerkt werden, dass die Operationen in den verschiedenen Arbeiten nicht deckungsgleich definiert sind und daher ggf. unterschiedliche Ergebnisse auf dieselbe Eingabe erzeugen würden.

Wie die von La Rosa et al. [RWM+11] beschriebene Untersuchung komplexitätsreduzierender Maßnahmen bei verschiedenen Prozesssprachen zeigt, werden das Entfernen und die Aggregation in den untersuchten Prozesssprachen (UML AD, BPMN, BPEL, YAWL) und Tools (Enterprise Architect 9, Signavio 5, JDeveloper 11) derzeit noch nicht durchgängig unterstützt. Einzig die Modularisierung mit der damit einhergehenden strukturellen und dauerhaften Änderung eines Prozesses wird umfassend unterstützt.

Anwendungsfälle der Abstraktion. Mehrere Ansätze zur Prozessabstraktion bezwecken das Verbergen von Strukturen in einem Geschäftsprozess zum Schutz sensibler Informationen. Chebbi et al. [CDT06] als auch Eshuis und Grefen [EG08] haben Modelltransformationen dazu eingesetzt, sogenannte Public Views auf Prozesse zu erzeugen, die interne Informationen ausblenden und einem Geschäftspartner nur diejenigen Teile eines Prozesses offenbaren,

die für die Zusammenarbeit relevant sind. Die Arbeiten von Chinosi [Chi08] und Reichert et al. [RBR+10] setzen Sichten auf Prozesse als Sicherheitsmechanismus für Prozessinformationen ein (engl. *access control*). Die Abstraktion eines Prozesses mit dem Zweck der Komplexitätsreduzierung stellt ebenso eine häufig untersuchte Fragestellung dar. Nach Polyvyanyy et al. [PSW10] ist die Bestimmung der Relevanz von Elementen eine wesentliche Aufgabe bei der Abstraktion. Abhängig vom gewünschtem Grad an Abstraktion, der bspw. durch einen Relevanz-Grenzwert angegeben werden kann, sollen wenig relevante Elemente aggregiert oder entfernt werden und relevante Elemente erhalten bleiben. Dieser Ansatz erfordert eine Abstufung der Relevanz von Elementen.

Einbeziehen der Darstellung. Das in mehreren Arbeiten erklärte Ziel von Sichten auf Prozesse besteht darin, eine rollen- oder aufgabenspezifische Form der Darstellung eines Prozesses zu erzeugen, die die Betrachtung und Anpassung der wesentlichen Teile unterstützt und die unwesentlichen Teile ausblendet, vgl. die für Prozesssichten wegweisende Arbeit von Bobrik [Bob08]. Wie Rinderle et al. [RBR+06] feststellen, gibt es zahlreiche Anwendungsmöglichkeiten von Prozesssichten in der Modellierung, dem Monitoring und der Analyse. Rinderle et al. stellen auch fest, dass die bloße Abstraktion u.U. ein schwer verständliches Prozessmodell als Resultat hervorbringt, sofern die Darstellung des Prozesses nicht berücksichtigt wird. Eine Modifikation der Darstellung unterstützt verschiedene Möglichkeiten der Analyse von Prozessen. In der Patentanmeldung von Gilbert et al. [GHN+08] wird bspw. die Darstellung eines Prozesses dahingehend angepasst, dass analytische Daten zu Laufzeit oder Kosten grafisch wie eine Heatmap über Aktivitäten und Verbindungen zwischen den Aktivitäten gelegt werden. Durch diese Form der Hervorhebung sind besonders kostenintensive Aktivitäten schnell erkennbar.

Bobrik [Bob08] stellt eine konfigurierbare grafische Darstellung vor, welche die Verwendung von grafischen Vorlagen (engl. *templates*) und Konfigurationsskripten vorsieht, mit denen unterschiedliche Informationen in die Darstellung eingebunden werden können. Bobrik hat bspw. in einer Sicht für Führungskräfte einen Richtungspfeil als grafische Vorlage für aggregierte Aktivitäten verwendet, um eine vereinfachte Darstellung eines Prozessmodells oder einer Prozessinstanz zu erreichen. Dieser Ansatz wird in der vorliegenden Dissertation aufgegriffen und im Rahmen sprachübergreifender Sichten untersucht. Als Technologie für die Vorlagen setzt Bobrik skalierbare Vektorgrafiken ein, die in XML beschrieben sind (SVG). Dies erweitert das Verständnis des Begriffs einer Sicht von der strukturellen Abstraktion um die Darstellung. La Rosa et al. [RHW+11] unterscheiden diese Aspekte in die Modifikation der abstrakten Syntax für strukturelle Änderungen und die Modifikation der konkreten Syntax für Änderungen der Darstellung. Ebenso wie Rinderle et al. [RBR+06] führen La Rosa et al. das Layout als wichtigen Aspekt der Darstellung an. Dieser Argumentation wird auch in der vorliegenden Dissertation gefolgt.

Instanzabstraktion. Naef et al. [NSS01] haben den Einsatz von sogenannten Schatten-Prozessen vorgeschlagen. Ein Schattenprozess kann als Public View auf einen detaillierten, internen Prozess interpretiert werden. Der Zustand des Schatten-Prozesses wird aus dem Zustand des Originalprozesses abgeleitet. Zur Mühlen [Müh04] stellt weiterführende Nachforschungen zum Monitoring von Prozessen an und beschreibt verschiedene Ebenen an denen

Monitoring ansetzen kann: auf Prozesskettenebene, auf Prozessebene, auf Segmentebene und auf Aktivitätsebene. Für einen höheren Abstraktionsgrad werden die Zustände von Segmenten ggf. zu einem Geschäftszustand (engl. *business state*) aggregiert, d.h. es wird eine abstrakte Sicht auf eine Prozessinstanz gebildet. Dieser Ansatz wird auch von Bobrik [Bob08] verfolgt: Mit Instanzinformationen angereicherte Prozesse werden durch Entfernen und Aggregieren von Aktivitäten und Datenelementen abstrahiert. Bobrik diskutiert in diesem Kontext sogenannte Zustandskonsistenz, die beschreibt ob der Zustand einer abstrahierten Prozessinstanz nach bestimmten Regeln konsistent ist. Auch Guo et al. [GHN+09] beschreiben eine Methode und eine Architektur zur Erzeugung einer abstrakten Sicht auf eine Prozessinstanz. Die Schritte der Methode bestehen in der Vereinfachung eines Prozesses durch Entfernen von Aktivitäten bestimmten Typs und in der anschließenden Anreicherung des abstrahierten Prozessmodells mit Instanzinformationen über den Ausführungszustand von Aktivitäten.

Änderungen in abstrahierten Prozessen. Ein Problem bei der Manipulation von abstrahierten Prozessen besteht in der Übertragung der Änderungen von einer Sicht auf den originalen, vollständigen Prozess. Analog zum View Update bei Sichten auf Datenbanken (vgl. Abschnitt 2.2.1) besteht ein Problem in der mangelnden Eindeutigkeit bestimmter Änderungen. Soll bspw. eine neue Aktivität zwischen zwei, in einer Sicht unmittelbar verbundenen Aktivitäten, eingefügt werden, ergeben sich im originalen Prozess ggf. mehrere Möglichkeiten für eine Platzierung der Aktivität. Derartige Mehrdeutigkeiten müssen nutzergesteuert oder automatisch aufgelöst werden. Kolb et al. [KKR12] stellen dem Nutzer Platzierungsstrategien zur Auswahl, mit denen die absolute Position einer eingefügten Aktivität direkt nach dem Vorgänger oder direkt vor dem Nachfolger der Einfügeposition in der Sicht bestimmt wird. Für die erlaubten Änderungen in einer Sicht stehen in diesem Ansatz bestimmte Parameter zur Verfügung, um die Übertragung der Änderung auf den vollständigen Prozess zu konfigurieren.

Untersuchte Prozesssprachen. Ansätze zu Sichten wurden für zahlreiche konkrete Prozesssprachen vorgeschlagen. Zum Beispiel wenden Zhao et al. [ZLS+09] abstrahierende Sichten auf Prozesse an, die in BPEL beschrieben sind. BPEL erlaubt die Verwendung von graphbasierten Strukturen als auch von Blockstrukturen, wie sie in der Programmierung gängig sind [KMW+09]. Die Arbeit von Zhao et al. beschränkt sich auf die Abstraktion der Blockstrukturen. Gottschalk et al. [GRA05] übertragen Konzepte zur Sichtenbildung auf die Sprache EPK. Sichten auf UML AD wurden von Eshuis und Grefen [EG08] betrachtet. Die Arbeit von Chinosi [Chi08] ist hingegen an BPMN ausgerichtet. Mehrere weitere Arbeiten sind an BPMN ausgerichtet, wobei lediglich eine Teilmenge des umfangreichen Sprachumfangs berücksichtigt wird. Avrilionis et al. [ACF96] haben Sichten auf Petri Netze definiert, um bestimmte Teile eines Prozesses auszuschneiden und auf diese Weise wiederverwendbar zu machen. Es bleibt an dieser Stelle festzuhalten, dass mehrere Arbeiten zur Erzeugung von Sichten für verschiedene Prozesssprachen existieren, diese Arbeiten jedoch bis auf die Anwendung der allgemein bekannten Abstraktionstechniken des Entfernens und der Aggregation von Prozessstrukturen keine gemeinsame konzeptionelle Basis und keine einheitliche Terminologie verwenden.

Trennung von Zuständigkeiten. Das klassische Konzept der Trennung von Zuständigkeiten findet auch in der modernen Modellierung und Analyse von Geschäftsprozessen Anwendung. Üblicherweise wird die Aufbauorganisation getrennt von der Ablauforganisation verwaltet. Wie Tran et al. [TZD07][TZD11] beschreiben, kann die Trennung von Zuständigkeiten auch für verschiedene Sichten auf eine Prozessstruktur eingesetzt werden. Der Ansatz besteht darin, verschiedene Sichten getrennt voneinander anzuzeigen, getrennt zu verwalten und ein vollständiges Prozessmodell nur auf Anfrage mittels einer Zusammenführung der einzelnen Modelle zu erzeugen. Die von Tran et al. vorgeschlagenen Sichten umfassen (1) den Kontrollfluss (Flow View), (2) die Schnittstellen, die von einem Prozess verwendet und bereitgestellt werden (Collaboration View), (3) die Datenobjekte, die in einem Prozess verwendet werden (Information View) und (4) menschliche Interaktionen (Human View). In grafischen Werkzeugen zur Prozessmodellierung und Analyse wird ebenfalls eine Trennung von Zuständigkeiten realisiert. Diese wird i.d.R. auf der Darstellungsebene realisiert, d.h. es erfolgt keine getrennte Datenhaltung unterschiedlicher Teilmodelle, sondern es gibt verschiedene Sichten auf eine gemeinsame Informationsbasis in Anlehnung an das Entwurfsmuster *Model View Controller* [GHJ95]. Das Werkzeug Eclipse BPEL Designer [Ecl11] stellt bspw. im Hauptbereich der Anwendung einen Prozessgraphen dar. In dieser Darstellung werden lediglich die Aktivitätstypen, deren Name und deren Kontrollverbindungen angezeigt. Details und andere Informationen werden in Registerkarten oder weiteren Sichten zugänglich gemacht, etwa in einer Baumstruktur oder einer Detailansicht.

2.2.3 Sprachübergreifende Sichten auf Geschäftsprozesse

Die formale Beschreibung eines Geschäftsprozesses in einem Prozessmodell legt einen bestimmten Abstraktionsgrad fest und beschreibt die in dem Prozess zu erledigenden Aufgaben in einer bestimmten Granularität. Geschäftsprozesse sind indes komplexe und vielschichtige Vorgänge. Ein aus der Perspektive des Business beschriebenes fachliches Prozessmodell deckt jedoch kaum alle Eventualitäten, Ausnahmen und Sonderbehandlungen ab, sondern beschreibt vielmehr den gewöhnlichen Ablauf. Für eine Prozessautomatisierung sind diese Details erforderlich. Darüber hinaus kann im Zuge der Implementierung eines Prozesses durch die IT (engl. *IT refinement*) eine weitere Prozesssprache zum Einsatz kommen für die eine Ausführungsumgebung zur Verfügung steht. Dadurch entstehen unterschiedliche Abstraktionsebenen, jede mit eigenen Prozessmodellen und -sprachen.

Abstraktionsebenen. Abbildung 2.9 illustriert die Thematik unterschiedlicher Abstraktionsebenen. Ein abstraktes Prozessmodell wird grob spezifiziert (Abbildung 2.9a), anschließend verfeinert und entweder manuell oder unter Einsatz von Techniken modellgetriebener Softwareentwicklung (engl. *Model-driven Software Development, MDSD*) in ein ausführbares Prozessmodell überführt. Nach dem Schritt der Verfeinerung existieren das abstrakte und das ausführbare Modell parallel.

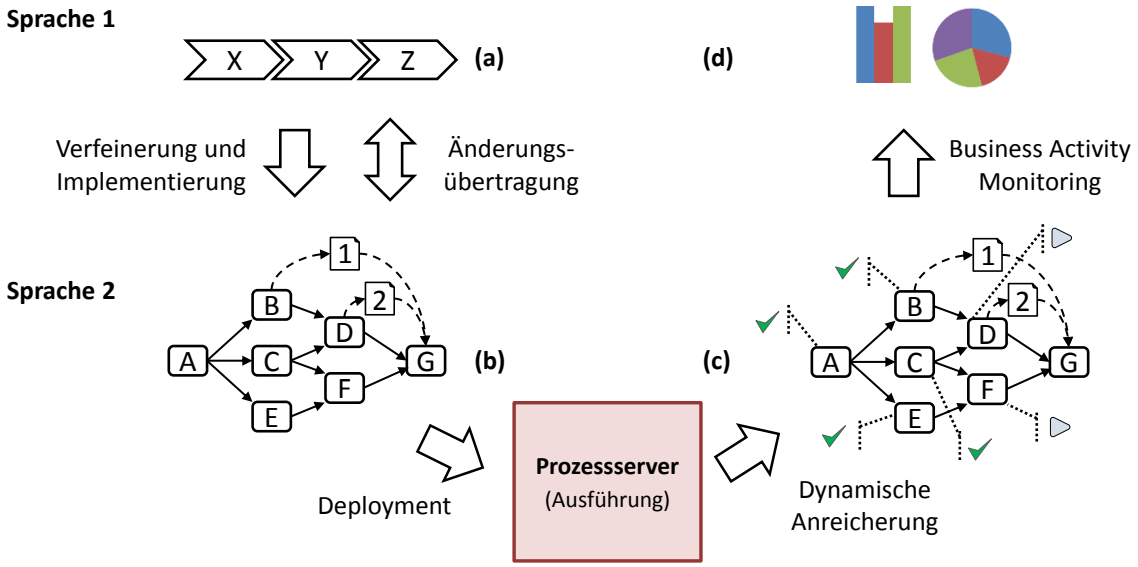


Abbildung 2.9: Abstraktionsebenen im Stand der Technik.

Änderungen am Prozess erfordern dann eine Synchronisation der Modelle (engl. *roundtrip*). Einen Ansatz zur Verwaltung und Übertragung von Änderungen auf Basis einer Änderungswarteschlange haben Weidmann et al. [WAK+11] diskutiert. Die Prozesselemente werden in diesem Synchronisationsansatz aufeinander abgebildet (engl. *mapping*). Auf Basis dieser Abbildung können von einer Änderung betroffene Elemente in anderen Prozessmodellen identifiziert und Änderungen systematisch mithilfe der Änderungswarteschlange verwaltet werden.

Ein ausführbares Prozessmodell (Abbildung 2.9b) kann auf einen Prozessserver deployed und anschließend zur Steuerung und Automatisierung der Ausführung eingesetzt werden. Eine Anreicherung mit Laufzeitinformationen ermöglicht das Monitoring und die Analyse von Instanzen dieses Prozessmodells (Abbildung 2.9c).

Monitoring. Nach dem Stand der Technik ist auf der höheren Abstraktionsebene (Sprache 1) lediglich das Monitoring auf Basis von KPI gängig (Abbildung 2.9d). Mit entsprechenden Business Activity Monitoring Frameworks (BAM) können Individuallösungen realisiert werden, in denen ein nutzerdefiniertes, grafisches Modell an ausgewertete KPI und andere Informationen angebunden wird [ME07]. Es existieren zahlreiche Ansätze, mit denen diese Form des Monitorings realisiert werden kann. Wetzstein et al. [WSL09] zeigen bspw., wie aus abstrakten, plattformunabhängigen Metriken bzgl. der Leistung eventbasierte Monitoring-Modelle für eine BPEL-Ausführungsumgebung generiert werden können. Der Darstellung der abstrakten KPI liegt ein anderes Metamodell zugrunde als dem Prozess aus dem sie abgeleitet werden. Kreisdiagramme, Balkendiagramme und grafische Anzeigen ähnlich zu einem Thermometer sind gängige Elemente, um eine Sicht auf einen Prozess zu schaffen, die den eigentlichen Prozessgraphen jedoch bis zur Unkenntlichkeit abstrahiert. Neben der Darstellung der Leistung eines Prozesses können sogenannte Business Dashboards auch dazu eingesetzt werden, eine Übersicht zur Erfüllung von Compliance-Anforderungen in Prozessen darzustellen [SRC+09]. Ein Compliance Governance Dashboard liefert demnach eine Auswertung kritischer Compliance-Anforderungen (Key Compliance Indicators, KCI).

Übersetzung. Neben dem Monitoring von Prozessen kann die Übersetzung eines Prozessmodells von einer Prozesssprache in eine andere Prozesssprache als sprachübergreifende Sicht auf einen Prozess interpretiert werden, die auf bestimmte Zielgruppen oder Zwecke ausgerichtet ist. In diesen Anwendungsbereich fallen die unterschiedlichen Ansätze und Werkzeuge zur Übersetzung zwischen Prozesssprachen. Mehrere Arbeiten befassen sich mit der Übersetzung zwischen den Sprachen BPMN und BPEL, wie bspw. Ouyang et al. [ODH+08] und im weiteren Sinne Mendling et al. [MLZ06], die Übersetzungsstrategien zwischen block- und graphstrukturierten Prozesssprachen untersuchen. Die Übersetzung von verschiedenen Prozesssprachen zu Petri Netzen ist auch ein wichtiges Thema in der wissenschaftlichen Diskussion, vornehmlich zur Nutzung der auf Petri Netzen basierenden Verifikationstechniken [Aal99]. Allerdings kann bei einer Übersetzung etwas verloren gehen, wenn die Prozesssprachen, zwischen denen übersetzt wird, eine unterschiedliche Ausdruckskraft besitzen. Werden etwa in einem zu übersetzenden Prozess Konstrukte verwendet, die in der Zielsprache nicht unterstützt werden, sind entweder komplizierte Hilfskonstrukte erforderlich oder eine vollumfängliche Übersetzung ist gar nicht erreichbar. Bspw. sind der Übersetzung zwischen BPEL zu EPK an dieser Stelle Grenzen gesetzt.

2.3 Stand der Technik von Compliance

In diesem Abschnitt wird der Stand der Technik von Compliance vorgestellt. Zunächst wird eine Abgrenzung der vorliegenden Dissertation zur juristisch-wirtschaftlichen Perspektive vorgenommen. Anschließend werden drei Strömungen wissenschaftlicher Ansätze vorgestellt, die für Compliance in Geschäftsprozessen als zentral betrachtet werden. Diese Strömungen betreffen die technische Beschreibung von Compliance-Anforderungen, die Prüfung der Regelkonformität und die Wiederverwendung im Bereich Geschäftsprozesse.

2.3.1 Abgrenzung und Terminologie

Im Themenkomplex Compliance kann zwischen einer juristisch-wirtschaftlichen und einer technischen Perspektive unterschieden werden. Moosmayer [Moo10] diskutiert die Erfordernisse, die Notwendigkeit, aber auch den Nutzen von Compliance in großen Unternehmen mit juristisch-wirtschaftlichem Hintergrund. Er geht detailliert auf das Compliance-Programm eines großen Unternehmens ein, das eine Vorbeugung von Verstößen, deren Aufdeckung, angemessene Reaktion sowie die fortlaufende Verbesserung vorsieht. Auch in der Unternehmensberatung spielt Compliance und das damit verbundene Risikomanagement eine wichtige Rolle. Caprasse et al. [CLR08] beschreiben bspw. eine dreistufige Strategie, um ein Compliance-Programm zu unterstützen. Diese Strategie umfasst die Einschätzung und Identifikation von Risiken, nachfolgende Maßnahmen zu deren Entschärfung und entsprechende Beobachtung und Kontrolle der Umsetzung.

So unterschiedlich die Themengebiete sind aus denen Compliance-Anforderungen stammen, so unterschiedlich sind auch die Interessengruppen, die deren Einhaltung einfordern. Was die Einhaltung von Compliance-Anforderungen aus rechtlichen Quellen angeht, sind letztlich Wirtschaftsprüfer diejenigen, die über die Compliance eines Unternehmens und seiner

Prozesse entscheiden. Compliance im Hinblick auf unternehmensinterne Richtlinien wird vornehmlich von der Management-Ebene, der internen Revision oder auch vom Betriebsrat bemessen, wobei eine zunehmende Anzahl von Unternehmen eine eigens auf Compliance ausgerichtete Organisationeinheit als zentrale Zuständigkeit aufbaut, das *Compliance Office* [Moo10]. Diese Organisationeinheit nimmt auch die Funktion der internen Betriebsprüfung hinsichtlich Compliance wahr. Ein Compliance-Prüfer stellt auf Basis der ihm zugänglichen Informationen fest, ob die Anforderungen, die nach seinem Verständnis für einen bestimmten Prozess gelten, seiner Ansicht nach angemessen umgesetzt wurden. Dies erfordert Nachvollziehbarkeit (engl. *traceability*) und Transparenz in der Handhabung der Anforderungen und deren Umsetzung in Prozessen und IT-Systemen.

Compliance-Anforderungen haben wenige charakteristische Merkmale, die sie von anderen Anforderungen unterscheiden. Sie zeichnen sich im Grunde dadurch aus, dass sie auf rechtliche oder interne Regelwerke zurückzuführen sind und einer Interpretation unter juristischem oder wirtschaftlichem Hintergrund unterliegen. Aufgrund der juristischen als auch wirtschaftlichen Risiken eines Compliance-Verstoßes wird Compliance-Anforderungen eine immer größere Bedeutung beigemessen. Eine Herausforderung beim Compliance Management besteht in der stark zunehmenden Komplexität, bewirkt durch die Vernetzung und Interaktion von Unternehmen, Sub-Unternehmen und Dienstleistern sowie durch die steigenden Umfänge an relevanten Regelwerken und deren Gültigkeitsbereichen. Die Zusammenhänge werden schwer überschaubar. Entsprechend wirkt sich die Umsetzung von Compliance-Anforderungen auf den Unternehmensaufbau, Prozessstrukturen, verwendete IT-Systeme als auch auf Geschäftsbeziehungen aus [Moo10].

Die Arbeiten mit juristisch-wirtschaftlicher Perspektive beschreiben nur unzureichend, wie der Übergang von Anforderungen und konkreten Maßnahmen zu automatisierten Prozessen bewerkstelligt werden kann oder soll. Die in der vorliegenden Dissertation entwickelten Ansätze setzen an diesem Punkt an und sollen Unterstützung dafür bieten, bereits identifizierte, widerspruchsfreie Compliance-Anforderungen in Geschäftsprozessen zu realisieren.

Im Kontext von Compliance werden bestimmte Begriffe verwendet, die im Folgenden erläutert werden. Die Begriffsbeschreibungen basieren auf der Terminologie, die im Konsortium des COMPAS-Projekts[2] erstellt wurde, vgl. [COM10d].

- Eine *Compliance-Quelle* (engl. *Compliance source*) stellt den Ursprung einer oder mehrerer Compliance-Anforderungen dar. Beispiele dazu sind Regelwerke wie Basel II [Bas06], unternehmensinterne Richtlinien und Verträge.
- Eine *Compliance-Anforderung* (engl. *Compliance requirement*) ist eine Auflage, die sich durch fachmännische (d.h. i.d.R. juristische) Analyse und Interpretation von relevanten Compliance-Quellen formulieren lässt.
- Ein *Compliance-Risiko* (engl. *Compliance risk*) beschreibt mögliche Folgen der Nichteinhaltung bestimmter Compliance-Anforderungen.

[2] COMPAS – Compliance-driven Models, Languages, and Architectures for Services

- Ein *Compliance-Instrument* (engl. *Compliance control*) bezeichnet die Einflussmöglichkeiten einer Compliance-Anforderung nachzukommen. Dazu gehören u.a. direkte Einflussnahme auf Prozesse, Personen und IT-Systeme als auch Überprüfungen.

2.3.2 Technische Beschreibung von Compliance-Anforderungen

Compliance-Anforderungen betreffen zahlreiche Aspekte von Geschäftsprozessen. Nach einer Studie [HPO08] sind dies im Wesentlichen Kontrollfluss, Datenverwendung und Datenschutz, räumliche Aspekte wie der Ort von beteiligten Personen, Daten und verwendete Ressourcen sowie Sicherheit, Lizensierung, vereinbarte Dienstgüteeeigenschaften (engl. *Service Level Agreement, SLA*) wie auch das Monitoring von Abläufen. An der Vielfalt dieser Aspekte wird deutlich, dass Compliance einen übergreifenden Aspekt (engl. *cross cutting concern*) darstellt, der Bereiche betrifft in denen unterschiedliche technische Beschreibungsformen erforderlich sind.

Compliance-Anforderungen gelten für alle Beschreibungsformen und Automatisierungsstufen von Geschäftsprozessen (vgl. Abschnitt 2.1.1). Folglich bestehen auch in der Beschreibung bestimmter Compliance-Anforderungen, wie bspw. für Aspekte der Sicherheit, mehrere Ebenen der Konkretisierung und Formalisierung. Während eine textuelle Beschreibung von Sicherheitsanforderungen noch viel Raum für Ausgestaltung, Interpretation und Umsetzung bietet, stellt eine technische, d.h. eine maschinenlesbare Beschreibung, eine präzise und eindeutige Form der Beschreibung dar, wie bspw. in Formaten zur Beschreibung von Zugangskontrolleigenschaften in XACML [OAS05].

Namiri und Stojanovic [NS07] unterscheiden etwa zwischen *high level control patterns* und *system level control patterns*, um Compliance-Anforderungen einerseits in der Terminologie von Compliance-Experten und andererseits auf einer konkreten Implementierungsplattform darzustellen. Sie berücksichtigen in ihrem Ansatz, dass die technische Implementierung eines Prozesses verschiedene IT-Systeme umfassen kann. Die Compliance-Instrumente zur Einflussnahme auf die IT-Systeme können daher bspw. in Form von logischen Formeln, in SQL-Ausdrücken oder durch Geschäftsregeln (engl. *business rules*) implementiert werden. Namiri und Stojanovic beschreiben außerdem häufig auftretende Anforderungen wie bspw. das Vorhandensein oder die Abwesenheit von bestimmten Aktivitäten, das 4-Augen-Prinzip, Berichtswesen, Autorisierung und Eskalation.

Sicherheit ist nur einer der Aspekte, für den zahlreiche Formate und Beschreibungsformen entwickelt wurden. In diesem Zusammenhang ist auch von domänenspezifischen Sprachen die Rede (engl. *Domain-specific Language, DSL*). Eine DSL stellt im Gegensatz zu einer Mehrzwecksprache (engl. *General Purpose Language, GPL*) eine für einen bestimmten Bereich ausreichend ausdrucksstarke Sprache dar, um einen Sachverhalt zu beschreiben. Im Umfeld von Compliance sind viele dieser DSL relevant. Mit Bezug auf die zuvor erwähnten Aspekte können an dieser Stelle eine DSL zur Lizensierung [GD11], für Dienstgüteeigenschaften [OZD09] und auch für das Monitoring von Abläufen [MZD10] genannt werden.

2.3.3 Prüfung von Geschäftsprozessen auf Regelkonformität

Techniken zur Prüfung der Regelkonformität von Geschäftsprozessen unterscheiden sich grundsätzlich in Bezug auf ihre Phasenzugehörigkeit im Prozesslebenszyklus. Eine mögliche Kategorisierung der unterschiedlichen Ansätze schlagen Kharbili et al. [KSM+08] vor. Zur Modellierungsphase werden vorbeugende Techniken angeführt (engl. *design time Compliance checking*, *forward Compliance checking* oder auch *Compliance by design*); zur Ausführungs- und Monitoringphase sind erkennende Techniken maßgeblich (engl. *runtime checking* oder *online monitoring*). Neben dem Erkennen eines (potentiellen) Verstoßes gegen Regeln besteht außerdem die Möglichkeit, unmittelbar in einen laufenden Prozess einzugreifen, um Compliance sicherzustellen (engl. *Compliance enforcement*); zur Analysephase wird eine nachträgliche Prüfung auf Regelkonformität betrachtet, d.h. nach dem Abschluss der Ausführung (engl. *after the fact, offline monitoring* oder *backward checking*).

Prüfung zur Modellierungszeit. Der Großteil der akademischen Arbeiten zur Prüfung der Regelkonformität setzt den Fokus auf den Prozessaufbau und den darin definierten Kontrollfluss. Dies umfasst u.a. das Vorhandensein von Aktivitäten und deren Abfolge im Prozess. Vergleichsweise wenige Arbeiten untersuchen die Regelkonformität des Datenflusses in einem Prozess, der beim Schutz sensibler Daten relevant ist [KLR+10][SFG+11]. Der Ansatz in der vorbeugenden Prüfung von Kontroll- und Datenflusseigenschaften besteht prinzipiell darin, an einen Prozess gestellte Compliance-Anforderungen zu formalisieren und deren Erfüllung im Prozess zu verifizieren. Für die Verifikation wird das Prozessmodell in ein anderes Modell übersetzt, welches in einem Model Checking System verarbeitet werden kann. Abouzaid und Mullins [AM08] geben bspw. eine Abbildung der Prozesssprache BPEL in einen Formalismus auf Basis von Pi-Kalkül an, wodurch BPEL-Prozesse mit dem HAL-Toolkit [FGM+03] überprüft werden können.

Wie Ly et al. [LGR+08] erklären, sollte eine formale Sprache für Compliance-Anforderungen ausdrucksstark sein und formale Grundlagen und Möglichkeiten zur Analyse bieten. Zur Formalisierung von Compliance-Anforderungen wird häufig Temporale Logik [Pnu77] verwendet, mit der logische Operatoren, zeitliche Operatoren und Aktivitäten eines Prozesses in Formeln verknüpft werden können. Neben Linearer Temporaler Logik (LTL) werden aber auch andere formale Sprachen verwendet wie Computation Tree Logic (CTL) oder auch Deontische Logik [COM09]. Diese Sprachen haben allerdings gemein, dass Compliance-Anforderungen ohne Weiteres in komplexen Ausdrücken resultieren und deren Formulierung und Verständnis fundierte mathematische Kenntnisse erfordern, was einem flächendeckenden Einsatz entgegensteht [YMH+06]. Darüber hinaus ist die Überprüfung formaler Regeln in Model Checking Systemen u.U. mit hohem Zeitaufwand verbunden, vgl. z.B. die Gegenüberstellung der Komplexität der Regelprüfung in LTL und CTL [Var01]. Es besteht aber die Möglichkeit, dem Zeitaufwand einer Regelprüfung durch vorangehende Vereinfachung des Prozesses mit Prozessabstraktionstechniken zu begegnen [LKR+10].

Vereinfachung formaler Ansätze. Eine Möglichkeit zur Reduzierung der Komplexität von formalisierten Anforderungen liegt in der Abstraktion der Formalisierung sowie in der Wiederverwendung von häufig auftretenden Anforderungen. Türetken et al. [TEH+12] setzen

eine abstrakte Beschreibung häufig verwendeter Formeln als Muster ein. Diese Art der Formelabstraktion wurde bereits von Dwyer et al. [DAC98] vorgeschlagen und von Yu et al. [YMH+06] im Kontext von Compliance untersucht. Türetken et al. bauen diese Grundlagen dahingehend aus, dass Compliance-Anforderungen zunächst formal definiert werden und diese Formalisierung zu einem Muster abstrahiert wird, in dem lediglich prozessbezogene Parameter verändert werden müssen. Auf diese Weise entsteht eine Hochsprache mit wiederverwendbaren Bausteinen, die den darunterliegenden Formalismus verbergen [Elg12].

Mit Regelkonformitätsbereichen (engl. *Compliance scopes*) schlagen Schleicher et al. [SWL+10] einen anderen Ansatz vor, um die Regelkonformitätsprüfung einfacher zu gestalten. Der Grundgedanke besteht darin, dass bestimmte Compliance-Anforderungen nur einen bestimmten Bereich eines Prozesses betreffen und folglich nur dieser Teil geprüft werden muss. Dadurch muss bei einer Regelprüfung wie auch bei einer festgestellten Regelverletzung nicht der gesamte Prozess betrachtet werden, sondern lediglich der betreffende Teilbereich.

Grafische Darstellung von Regeln. Durch die Visualisierung von einem mit formalisierten Compliance-Anforderungen angereicherten Prozess kann ein Modellierer bei der Umsetzung der Anforderungen unterstützt werden, wie Sadiq et al. [SGN07] beschreiben. Ferner kann die Sprache BPMN-Q [ADW08] dazu verwendet werden, eine Anforderung grafisch in Form eines Prozessgraphen zu formulieren, in dem neben Kontrollflussoperatoren auch temporale Operatoren verwendet werden können. Zur Prüfung der Konformität wird eine grafische Anforderung in BPMN-Q in einen logischen Ausdruck in LTL übersetzt, das zu prüfende Prozessmodell vereinfacht und in einen endlichen Automat (engl. *finite state machine*) übersetzt. Anschließend kann der logische Ausdruck in einem Model Checker gegen den endlichen Automat geprüft werden. Ein Prozessmodell kann außerdem mit einer festgestellten Regelverletzung angereichert und diese Information auf die Darstellung eines Prozessmodells übertragen werden, um den Modellierer in der Auflösung des Konflikts zu unterstützen [Awa10]. Ein ähnlicher Ansatz wie in BPMN-Q wird von Liu et al. [LMX07] verfolgt. In dieser Arbeit werden Anforderungen in der grafischen Business Property Specification Language (BPSL) beschrieben, die Prozessmodelle liegen in der Sprache BPEL vor. Die Anforderungen werden ebenfalls in LTL übersetzt. Zur Prüfung wird der BPEL-Prozess zuerst in Pi-Kalkül und anschließend in einen endlichen Automaten übersetzt. Ly et al. [LKR+10] schlagen Regelgraphen als grafischen Formalismus vor (engl. *Compliance rule graphs*). In Regelgraphen können Compliance-Anforderungen als Graphen dargestellt werden, welche die geforderte Abfolge von Aktivitäten beschreiben. Außerdem wird in einem Regelgraphen unterschieden ob bestimmte Aktivitäten auftreten oder ausbleiben sollen.

Der Ansatz von Förster et al. [FES05] diskutiert Muster von UML AD, um Anforderungen an das Qualitätsmanagement nach ISO 9001 abzubilden. Obgleich dieser Ansatz nicht im Themenkomplex Compliance verortet ist, dient er doch dazu, Anforderungen in Form von Prozessstrukturen zu formalisieren und diese gegen einen Prozess zu prüfen. Förster et al. stellen fest, dass eine geprüfte, aber verletzte Anforderung in einem Prozess dennoch enthalten sein kann – nur auf einem anderen Abstraktionsgrad. Diese Problematik stellt sich auch für die Regelkonformitätsprüfung wenn die formalisierte Regel und der zu prüfende

Prozess in unterschiedlichen Abstraktionsgraden vorliegen und so fälschlicherweise ein Regelverstoß festgestellt wird.

Laufzeitüberprüfung. Bestimmte, vornehmlich (lauf-)zeitbezogene Anforderungen wie etwa die Einhaltung einer bestimmten Frist können nur eingeschränkt zur Modellierungszeit geprüft werden. Für das Monitoring der Ausführung stehen zahlreiche Techniken bereit, die auf Events arbeiten, die während der Prozessausführung erzeugt werden. Complex Event Processing (CEP) Systeme wie bspw. ESPER [Esp12] bieten eine SQL-ähnliche Syntax zur technischen Beschreibung zu prüfender Anforderungen. Einfache Events können zu komplexeren aggregiert werden, um damit bestimmte Regelverstöße feststellen zu können [BDL+10].

Sackmann und Kähmer [SK08] haben einen weiteren Ansatz zum Monitoring von Compliance zur Laufzeit von Prozessen entwickelt. Zu einem auszuführenden Prozess wird zusätzlich ein Kontrollprozess gestartet, der während der Laufzeit bestimmte Richtlinien überprüft, die von Compliance-Anforderungen abgeleitet werden. Dadurch können Routinen zur Ausnahmebehandlung hinterlegt werden, bspw. ein Einleiten von Sanktionen.

Wenn der Regelverstoß unmittelbar verhindert oder blockiert werden soll, ist im Englischen auch von *Compliance Enforcement*, d.h. dem Erzwingen von Compliance, die Rede. Bspw. beschreiben Gheorghe et al. [GCS+10], dass ein potentieller Regelverstoß auf Basis von Events entdeckt werden kann und verschiedene Gegenmaßnahmen zur Verfügung stehen:

- Blockieren einer Prozessinstanz
- Blockieren einer Nachricht
- Terminieren oder Außerkraftsetzen einer Prozessinstanz
- Einfügen, Löschen oder Modifizieren von Aktivitäten einer Prozessinstanz
- Modifikation von Variablen einer Prozessinstanz

Methoden zur nachträglichen Prüfung auf Regelkonformität basieren i.d.R. auf der Analyse von Events die während der Ausführung eines Prozesses aufgezeichnet wurden. Das Process Mining liefert an dieser Stelle den Nachweis, ob bestimmtes Verhalten aufgetreten oder ausgeblieben ist [RFA12]. Ein Vorteil des Mining Ansatzes besteht darin, dass Prozesse mit unterschiedlicher Prozesssprache gleichermaßen geprüft werden können, da die Prüfung auf den aufgezeichneten Events basiert. Ramezani et al. [RFA12] modellieren einen ganzen Katalog von Compliance-Anforderungen in Form von Petri Netzen, um auf diese Weise regelkonformes Verhalten zur Prüfung wiederverwenden zu können.

2.3.4 Wiederverwendung im Bereich Geschäftsprozesse

Die Wiederverwendung von formalisierten Compliance-Anforderungen, mit der die Überprüfung der Regelkonformität vereinfacht werden kann, wurde bereits angesprochen. Allerdings besteht dadurch keine ausreichende Hilfestellung dafür, wie ein Geschäftsprozess tatsächlich geändert werden soll, um die angelegten Anforderungen zu erfüllen. Daher wird nachfolgend auf bestehende Ansätze zur Wiederverwendung von Prozessstrukturen eingegangen.

Prozesse. Durch die Verwendung eines Prozesses als Vorlage für weitere Prozessmodelle (engl. *process template*) kann die Bildung von Varianten eines Prozesses unterstützt werden, welche auf einen bestimmten Anwendungsbereich oder eine Organisation zugeschnitten sind. Die explizite Auszeichnung von Freiheitsgraden in einer Prozessvorlage unterstützt einen Modellierer bei der Erstellung von Varianten. Konfigurierbare Prozesse [ADG+05] erlauben die Anpassung eines Referenzmodells auf konkrete Anforderungen. Eine alternative Möglichkeit wird von Schleicher et al. [SAL+09] diskutiert: Das Konzept eines Compliance Templates wird als regelkonforme Prozessvorlage vorgeschlagen, die die Compliance-Anforderungen einer Domäne in regelkonformen Prozessstrukturen umsetzt und eingeschränkte Platzhalter zur individuellen Ausgestaltung vorsieht.

Sub-Prozesse. Sub-Prozesse als abgeschlossene, wiederverwendbare Einheiten eines Geschäftsprozesses wurden im Rahmen der Modularisierung als klassisches Konzept zur Reduzierung der Komplexität bereits erwähnt. Zu dem Konzept eines Sub-Prozesses wurden verschiedene Diskussionen in der Wissenschaft geführt, jedoch nicht im Kontext von Compliance. Bspw. sind *Worklets* [AHE+06] in sich abgeschlossene Sub-Prozesse zur Bearbeitung einer Aufgabe innerhalb eines Prozesses in der Sprache YAWL. Ein Worklet wird in Abhängigkeit vom Kontext einer bestimmten Prozessinstanz ausgewählt und ausgeführt. Das Konzept *Pockets of Flexibility* [SSO01] beschreibt ebenfalls modulare Einheiten, deren Wiederverwendbarkeit durch Flexibilität erhöht werden soll: Eine Pocket of Flexibility besteht aus einer Menge an Workflowfragmenten, d.h. Aktivitäten oder Sub-Prozessen sowie aus einer Build-Aktivität, die zur konkreten Festlegung einer Komposition der Fragmente dient. Sub-Prozesse werden häufig, aber nicht notwendigerweise mit genau einem definierten Anfang und genau einem definierten Ende beschrieben (Single-Entry-Single-Exit, SESE) [KEL+10].

Muster. Message Exchange Patterns (MEP) [W3C03] wurden vom World Wide Web Consortium (W3C) zur Beschreibung der gängigen Muster des Nachrichtenaustauschs zwischen Sendern und Empfängern definiert, wie bspw. *request/response* und *one-way*. Neben den vom W3C beschriebenen MEP haben Barros et al. [BDH05] mit Dienstinteraktionsmustern (engl. *service interaction patterns*) weitere Muster des Nachrichtenaustauschs identifiziert. Das Muster *One-to-many send/receive* beschreibt bspw., dass eine Partei eine Nachricht an mehrere andere Parteien schickt und möglicherweise nicht von allen in einer vorgegebenen Zeit eine Antwort erhält. Ferner wurden *Workflow Patterns* [AHK+03] zur Erfassung und zum Vergleich der Ausdrucksstärke von Prozesssprachen und Geschäftsprozesstechnologien definiert. Workflow Patterns erfassen verbreitete Sprachkonstrukte, wie bspw. sequentielle Ausführung oder die Synchronisation paralleler Pfade. Mithilfe der Workflow Patterns können Prozesssprachen gegenübergestellt werden. Sie können außerdem zur Beschleunigung der Prozessmodellierung eingesetzt werden [GKW08].

Prozessstrukturen. Die Wiederverwendung zusammenhängender Prozessstrukturen wurde mit zahlreichen Begrifflichkeiten und Auslegungen diskutiert. Rolland und Prakash haben bspw. den Ansatz wiederverwendbarer Bausteine vorgeschlagen, den sogenannten Process Chunks [RP93]. In den letzten Jahren wird zunehmend der Begriff *Prozessfragment* oder analog *Workflowfragment* verwendet. Von der Wortherkunft ist ein Fragment ein Bruchstück

eines Ganzen. Übertragen auf Prozesse ist ein Fragment demnach Teil eines Prozesses und im Gegensatz zu einem Sub-Prozess nicht zwingend in sich abgeschlossen. Allerdings wird die Beschaffenheit eines Prozessfragments recht unterschiedlich ausgelegt und vom Konzept eines Sub-Prozesses nicht immer abgegrenzt. Manche Arbeiten gehen bei Fragmenten von wiederverwendbaren SESE-Prozessstrukturen aus, wie z.B. Sadiq et al. [SSO01]. Thom et al. [TRI09] schlagen eine Sammlung von sogenannten Workflow Activity Pattern (WAP) vor, um gängige Fragestellungen der Modellierung in Form wiederverwendbarer modularer Blöcke zu beschreiben. Zu den WAP gehören bspw. Genehmigung, Frage-Antwort, Benachrichtigung und Entscheidung. Thom et al. [TRI09] stellen fest, dass eine SESE-Einschränkung die Implementierung und Modellierung deutlich vereinfacht. Die SESE-Einschränkung ist in anderen Arbeiten gelockert. Bspw. schlagen Eberle et al. [EUL09] vor, Fragmente mit mehreren Ein- und Ausgängen zu erlauben. Nach Eberle et al. werden Eingänge und Ausgänge durch nur teils verbundene Kontrollverbindungen dargestellt (engl. *dangling control connectors*); Platzhalter, sogenannte Regionen (engl. *regions*), dienen in diesen Fragmenten dazu, Freiheitsgrade bei deren Komposition zu kennzeichnen.

Goderis et al. [GSL+05] schlagen außerdem vor, Workflowfragmente mit Parametern zu versehen (Parametrisieren) und diese Parameter beim Einpassen des Fragments in einen Workflow festzulegen (Parametrieren). Nach Goderis et al. können Workflowfragmente auch eine gute Lösung für eine bestimmte Problemstellung in einem wissenschaftlichen Workflow sein, die von anderen Wissenschaftlern wiederverwendet werden kann. Die Ausführungen von Ma [Ma12] sehen überdies vor, auch Routinen zur Behandlung von Ereignissen, Fehlern und Kompensationslogik für langlaufende Transaktionen in einem Prozessfragment unterzubringen. In der Interpretation von Lindert und Deiters [LD99] sind Prozessfragmente hingegen eine Public View auf den vereinbarten Informationsaustausch von Teilnehmern in einer Choreographie. Die thematische Verbundenheit von Prozessfragmenten und Sichten nimmt in der vorliegenden Dissertation an anderer Stelle noch größeren Raum ein und wird daher an dieser Stelle nicht weiter vertieft.

Domänenspezifische Prozessstrukturen. Rupprecht et al. [RFK+00] beschreiben den Ansatz eines domänenspezifischen Prozessbaukastens (engl. *process construction kit*), der prozessuales Wissen in Form modularer und parametrierbarer Prozessfragmente zur Wiederverwendung bereitstellt. Rupprecht et al. [RRF+01] erweitern diesen Ansatz um zehn Aktivitätssymbole zur Darstellung der Aktivitätssemantik. Diese Symbole repräsentieren Tätigkeitsmuster der Domäne Fahrzeugentwicklung: (1) Planung, (2) Entwicklung, Konstruktion, (3) Kommunikation, Besprechung, (4) Recherche, Untersuchung, (5) Simulation, Berechnung, (6) Entscheidung, (7) Erwerb, Einkauf, Akquisition, (8) Herstellung, Produktion, (9) Experiment, Test und (10) allgemeine Aktivität, die als Kategorie für Aktivitäten dient, die nicht in dieses Schema passen.

Mit der PICTURE-Methode[3] wurde eine domänenspezifische Sprache zur Prozessmodellierung auf einem hohen Abstraktionsgrad vorgeschlagen. Durch die Fokussierung auf eine

[3] PICTURE – Process Identification and Clustering for Transparency in Reorganising Public Administrations

bestimmte Domäne ist es nach Becker et al. [BPR07] möglich, alle gängigen Schritte in einem Prozess als wiederverwendbare Bausteine zu definieren. Baacke et al. [BRW07] beschreiben dazu, dass von den Inputs und Outputs konkreter Aktivitäten abstrahiert wurde, um auf Basis der so identifizierbaren Referenzbausteine ein besseres Verständnis und eine einfachere Vergleichbarkeit von Prozessen zu erzielen, die mit der PICTURE Methode modelliert werden. Etwa 30 dieser Referenzbausteine wurden für die Domäne der öffentlichen Verwaltung modelliert. Wie Räcker [Räc09] beschreibt, sind diese Bausteine nach Anwendungsphasen gruppiert, etwa *Inhaltliche Verwaltungsarbeit* und *Informationsbeschaffung und Koordination*. Die abstrakten Bausteine, wie bspw. *Dokument Archivieren* oder *Rückfrage durchführen*, werden bei der Modellierung entsprechend der Semantik des Prozesses konkretisiert. Diese Bausteine entsprechen dem Vokabular und Verständnis der Domäne und können in Form einfacher Sequenzen zu einem Prozess kombiniert werden. Zur Modellierung von PICTURE-Prozessen werden außerdem Schleifen, bedingte Verzweigungen und Sub-Prozesse verwendet [PIC06].

Regeln. Geschäftsregeln (engl. *business rules*) können als feingranulare Einheiten der Wiederverwendung betrachtet werden, wobei auch bei Geschäftsregeln verschiedene Grade an Abstraktion, Formalisierung und Implementierung unterschieden werden [TBR00]. Sie können mit Business Rule Management Systemen verwaltet werden. Man kann sie in ausführbare Geschäftsprozesse integrieren, um diese Prozesse leichter an geänderte Anforderungen anpassen zu können. Bspw. kann die Anforderung, ab einer bestimmten Höhe eines Kreditantrags einen Manager zur Prüfung hinzuzuziehen, als Geschäftsregel implementiert und als Aktivität, Gateway oder Bedingung in einen Prozess integriert werden. Durch eine Anpassung der Regel kann ein Prozess unmittelbar an eine geänderte Anforderung angepasst werden. Außerdem kann eine Geschäftsregel in mehreren Prozessen verwendet werden. Änderungen an einer Regel werden dadurch in allen relevanten Prozessen konsistent umgesetzt.

Wie Casati et al. [CCF+00] beschreiben, können abstrakte Regeln als wiederverwendbare Muster beschrieben werden. Eine Regel besteht aus einem Hauptteil, der als Vorlage (engl. *template*) dient. Eine Regel beinhaltet eine Menge an regelaktivierenden Events, eine Bedingung und mögliche Reaktionen, die ausgeführt werden, wenn die Bedingung erfüllt ist (engl. *Event Condition Action, ECA*). Als Metainformationen zu dem Muster werden ein Name, der Zweck, eine Klassifikation, Schlüsselwörter, verwandte Muster und Anwendungsrichtlinien angegeben. Ein Template Interface unterstützt die Parametrierung des Musters. Konkrete Beispiele (engl. *sample usage*) dienen dazu, die Anwendung des Musters praktisch zu veranschaulichen. Bei der Musterinstanziierung werden die Parameter festgelegt.

2.4 Anforderungsanalyse und Positionierung

In diesem Abschnitt werden die Grenzen des Standes der Technik diskutiert und Anforderungen identifiziert, um die Komplexität von Geschäftsprozessen und Compliance zu reduzieren. Die vorliegende Dissertation wird dadurch gegen den Stand der Technik abgegrenzt und dazu positioniert.

Identifizierte Anforderungen werden mit *A*, Voraussetzungen mit *V* und Feststellungen mit *F* abgekürzt und über eine fortlaufende Nummer referenziert. Am Ende dieses Abschnitts werden alle Anforderungen, Voraussetzungen und Feststellungen zusammengefasst, siehe dazu Tabelle 2.1 ff.

2.4.1 Geschäftsprozesse in der Praxis

Den Erläuterungen in Abschnitt 2.1 zufolge muss bei der Entwicklung von Ansätzen beachtet werden, dass Geschäftsprozesse auf unterschiedlichen Abstraktionsebenen beschrieben (A1) und für diese Ebenen unterschiedliche Prozesssprachen verwendet werden können (A2). Des Weiteren soll berücksichtigt werden, dass ggf. nur Teile eines Geschäftsprozesses durch IT-Systeme unterstützt werden (A3) und nur ein weiterer Teil davon in Workflows umgesetzt ist (A4). In Abgrenzung zu den möglichen Prozessreifegraden (vgl. Abschnitt 2.1.1) wird allerdings vorausgesetzt, dass für einen Geschäftsprozess zumindest ein Prozessmodell in einer graphbasierten Prozesssprache vorliegt (V1).

In den entwickelten Lösungen sollen aktuelle, für den jeweiligen Aspekt maßgebliche Architekturen und Technologien eingesetzt werden (A5). Der Architekturstil *SOA* kann für die Automatisierung von Geschäftsprozessen als maßgeblich betrachtet werden (F1); *Web Services* sind eine aktuelle und weitverbreitete Technologie zur Realisierung einer SOA (F2).

2.4.2 Sichten auf Geschäftsprozesse

Einheitliche Beschreibungsform. Die Ansätze zu Sichten auf Prozesse im Stand der Technik sind heterogen im Hinblick auf die diskutierten Prozesssprachen und Transformationen, die zur Bildung einer Sicht eingesetzt werden. Eine strukturierte Gegenüberstellung von Ansätzen erfordert eine gemeinsame Terminologie und konzeptionelle Basis, welche die wesentlichen Aspekte von Sichten auf Geschäftsprozesse konsolidiert. Mit einer derartigen Basis kann der Erfahrungsschatz bestehender Ansätze einfacher zugänglich gemacht werden und eine Grundlage für weiterführende Forschung bilden. Daher soll eine einheitliche Beschreibungsform entwickelt werden, mit der bewährte Techniken im Bereich Sichten wiederverwendet werden können (A6). Die Frage nach den Eigenheiten und Gemeinsamkeiten von Sichten soll durch eine strukturgebende Klassifikation unterstützt werden (A7).

Praktische Einsetzbarkeit. Die bestehenden wissenschaftlichen Ansätze sind algorithmisch und formal weitreichend, vernachlässigen allerdings die organisatorische Dimension. Zur praktischen Einsetzbarkeit von Sichten im Unternehmenskontext soll die organisatorische Dimension berücksichtigt werden (A8). Im praktischen Einsatz ist davon auszugehen, dass Funktionen zur Erzeugung von Sichten als (Code-)Module entwickelt werden, die in bestehende Prozesswerkzeuge integriert, oder lose gekoppelt als Dienst bereitgestellt werden. Unter dieser Annahme soll betrachtet werden, wie Ansätze zur Erzeugung von Sichten mit aktuellen Technologien modular bereitgestellt und zu komplexen Diensten komponiert werden können (vgl. F1, F2). Durch die Kapselung als Dienst sollen Sichten als Zwischenergebnis für nachfolgende Verfeinerung und Darstellung wiederverwendet werden können (A9). Die praktische Einsetzbarkeit der entwickelten Ansätze soll validiert werden (A10).

Weiterentwicklung. Die Möglichkeiten der grafischen Darstellung von Prozessen finden als komplexitätsreduzierendes Mittel bisher nur eingeschränkt Beachtung, als Ausnahme ist an dieser Stelle die Arbeit von Bobrik [Bob08] zu nennen. Die bestehenden Möglichkeiten grafischer Darstellung sollen in den entwickelten Ansätzen berücksichtigt und zur Weiterentwicklung zugänglich gemacht werden (A11). Neben der Betrachtung von Prozessmodellen sind Sichten auf Prozessinstanzen praktisch relevant, um die Lebenszyklusphase des Monitorings umfassender zu unterstützen. Daher sollen Ansätze zum Monitoring von Prozessen mit Sichten entwickelt werden (A12). Diese Ansätze sollen berücksichtigen, dass Prozesse in mehreren Abstraktionsebenen (vgl. A1) mit unterschiedlichen Prozesssprachen (vgl. A2) vorliegen können.

Einschränkungen. Das bei Sichten auf Datenbanken bestehende View Update Problem wird auch im Bereich von Sichten auf Prozesse untersucht, vgl. bspw. den Ansatz von Kolb et al. [KKR12] zur Auflösung von Mehrdeutigkeiten bei Prozessänderungen in einer Sicht. Die Manipulation eines Prozesses in einer Sicht ist ein komplexes und vielschichtiges Problem, dessen angemessene Betrachtung den Rahmen der vorliegenden Arbeit sprengen würde. Es werden zwar Sichtenentwürfe zur Unterstützung der Prozessmodellierung betrachtet, im weiteren Verlauf wird allerdings auf den Entwurf und die Erzeugung von prozessdarstellenden Sichten fokussiert.

Eine weitere Einschränkung in dieser Arbeit liegt in der Fokussierung auf einzelne Prozesse. Sichten auf Choreographien, d.h. unternehmensübergreifende Prozesse werden nicht betrachtet. Derartige Sichten sind relevant, jedoch wäre eine angemessene Betrachtung von Choreographien unter Berücksichtigung von Compliance zu umfangreich.

2.4.3 Compliance von Geschäftsprozessen

Rahmenbedingungen. Die Umsetzung von Compliance-Anforderungen und die Prüfung von deren Einhaltung sollen in Abstimmung mit der in einem Unternehmen vorhandenen technischen und organisatorischen Infrastruktur erfolgen (A13). Die entwickelten Ansätze setzen für einen Geschäftsprozess, dessen Regelkonformität unterstützt werden soll, das Bestehen einer Menge von widerspruchsfreien Compliance-Anforderungen voraus (V2). Diese Voraussetzung schließt eine Auflösung von Konflikten bereits mit ein, die sich ergeben können, wenn mehrere Regelwerke Anwendung finden.

Die für einen Prozess relevanten Gesetze, Regelungen und Verträge ändern sich im Laufe der Zeit und folglich verändern sich auch die an einen Prozess angelegten Compliance-Anforderungen (F3). Bei der Umsetzung von Compliance-Anforderungen soll die Nachvollziehbarkeit unterstützt werden (A14), um diesem Umstand Rechnung zu tragen.

Wiederverwendung. Bestehende Ansätze zur Prüfung der Regelkonformität weisen zwar mögliche Regelverletzungen in einem Prozessmodell nach, allerdings unterstützen sie den Modellierer kaum bei der eigentlichen Herstellung der Regelkonformität. Daher soll ein Ansatz entwickelt werden, der eine Prüfung der Regelkonformität nach dem Stand der

Technik einschließt, jedoch auch Methoden und Hilfsmittel bereitstellt, um Regelkonformität aktiv herzustellen (A15).

Für die Herstellung der Regelkonformität in Prozessen sollen wiederverwendbare Prozessstrukturen als Vorlage für regelkonformes Verhalten entwickelt werden (A16), sofern bestehende Konzepte der Wiederverwendung für Compliance-Anforderungen nicht ausreichen.

2.4.4 Querbezüge von Sichten und Compliance

Mit Laufzeitinformationen zu Prozessen kann man einen Einblick in deren Ausführung und Regelkonformität gewinnen. Daher sollen in der Betrachtung von Compliance Laufzeitaspekte berücksichtigt werden (A17). Compliance ist ein übergreifender Aspekt, der unterschiedliche Aufgaben und Interessengruppen betrifft. Jede der Interessengruppen hat einen individuellen Bedarf an Informationen und ein individuelles Verständnis, welche Informationen relevant sind. Daher sollen Sichten entwickelt werden, die unterschiedlichen Interessengruppen Möglichkeiten bieten, Informationen auszublenden, hinzuzufügen und individuell darzustellen (A18).

Die Erweiterung eines Prozesses durch regelkonforme Prozessstrukturen kann eine Prozessbelastung bewirken (engl. *process pollution*), da die eingefügten Strukturen mitunter keine wertschöpfende Arbeit darstellen [SLM+10]. Was einen Prozess belastet, liegt allerdings im Auge des Betrachters. Während für einen Betriebsprüfer gerade die Compliance-relevanten Strukturen von Bedeutung sind, versperren sie anderen Interessengruppen die Sicht. Ein ähnliches Problem besteht bei Geschäftsprozessen, die fachliche und informationstechnische Strukturen enthalten, wodurch eine sogenannte *semantische Verschmierung* vorliegt [Bun12]. Daher soll eine Sicht entwickelt werden, die einer Prozessbelastung entgegenwirkt (A19).

Zwischen dem Einsatz der grafischen Darstellung und dem Themenkomplex Compliance gibt es verschiedene Querbezüge. Eine grafische Hervorhebung von regelverletzenden Prozessstrukturen ist bspw. Stand der Technik in grafischen Werkzeugen zur Prozessverifikation. Auch wurde bereits vorgeschlagen, an einen Prozess annotierte Prüfregeln grafisch im Prozess darzustellen [SGN07]. Die Möglichkeiten von Sichten für die Anreicherung eines Prozesses mit Compliance-relevanten Informationen und deren Darstellung im Prozesskontext sind damit jedoch nicht ausgeschöpft. Zur Unterstützung der Nachvollziehbarkeit soll eine Sicht entwickelt werden, die Betriebsprüfern aufzeigt, wie Compliance-Anforderungen in einem Prozess realisiert wurden (A20).

Sichten erfüllen nicht nur den Zweck der Reduzierung der Komplexität, sondern sie haben auch einen direkten Bezug zur Wiederverwendung, weil sie zur Reduzierung eines Prozesses auf eine wiederverwendbare Einheit eingesetzt werden können [ACF96]. Daher soll eine Sicht entwickelt werden, die als Technik zur Wiederverwendung von Prozessstrukturen mit Bezug zu Compliance eingesetzt werden kann (A21).

2.4.5 Zusammenfassung der Zielstellung

In diesem Abschnitt werden die identifizierten Anforderungen, Feststellungen und Voraussetzungen zusammengefasst, welche als Zielführung dieser Arbeit dienen. Anforderungen sind in Tabelle 2.1 aufgelistet. Die Überblicksseiten zu den Teilen der vorliegenden Dissertation bieten Aufschluss, welche Anforderungen in welchem Kapitel berücksichtigt werden.

Tabelle 2.1: Zusammenfassung der Anforderungen an die Lösungsansätze.

	Nr.	Anforderung
Geschäftsprozesse in der Praxis	A1	Lösungsansätze sollen berücksichtigen, dass Geschäftsprozesse auf unterschiedlichen Abstraktionsebenen beschrieben werden können.
	A2	Für unterschiedliche Abstraktionsebenen eines Prozesses sollen unterschiedliche Prozesssprachen verwendet werden können.
	A3	Lösungsansätze sollen berücksichtigen, dass ggf. nur Teile eines Prozesses durch IT-Systeme unterstützt werden.
	A4	Es soll berücksichtigt werden, dass ggf. nur Teile der durch IT-Systeme unterstützten Teile eines Prozesses durch Workflows implementiert sind.
	A5	Lösungsansätze sollen aktuelle, für den jeweiligen Aspekt maßgebliche Architekturen und Technologien einsetzen (vgl. F1, F2).
Sichten auf Geschäftsprozesse	A6	Es soll eine einheitliche Beschreibungsform entwickelt werden, mit der bewährte Techniken im Bereich Sichten wiederverwendet werden können.
	A7	Sichten sollen nach einer strukturgebenden Klassifikation eingeordnet werden.
	A8	Zur praktischen Einsetzbarkeit von Sichten im Unternehmenskontext soll die organisatorische Dimension berücksichtigt werden.
	A9	Sichten sollen als Zwischenergebnis für nachfolgende Verfeinerung und Darstellung wiederverwendet werden können.
	A10	Die praktische Einsetzbarkeit der entwickelten Ansätze soll validiert werden.
	A11	Die bestehenden Möglichkeiten grafischer Darstellung sollen in den entwickelten Ansätzen berücksichtigt und zur Weiterentwicklung zugänglich gemacht werden.
	A12	Es sollen Ansätze zum Monitoring von Prozessen mit Sichten entwickelt werden.
Compliance von Geschäftsprozessen	A13	Lösungsansätze zum Management von Compliance sollen technische und organisatorische Gegebenheiten eines Unternehmens berücksichtigen.
	A14	Bei der Umsetzung von Compliance-Anforderungen soll die Nachvollziehbarkeit unterstützt werden (vgl. F3).
	A15	Es soll ein Ansatz zur Unterstützung von Compliance entwickelt werden, der sowohl die Prüfung von Regelkonformität umfasst, als auch Techniken und Hilfsmittel, um Regelkonformität aktiv herzustellen (vgl. F4).
	A16	Es sollen wiederverwendbare Prozessstrukturen als Vorlage für regelkonformes Verhalten entwickelt werden.

Querbezüge von Sichten und Compliance	A17	In Sichten für Compliance sollen Laufzeitaspekte berücksichtigt werden.
	A18	Es sollen Sichten entwickelt werden, die unterschiedlichen Interessengruppen Möglichkeiten bieten Informationen auszublenden, hinzuzufügen und individuell darzustellen.
	A19	Es soll eine Sicht entwickelt werden, die einer Prozessbelastung entgegenwirkt.
	A20	Es soll eine Sicht entwickelt werden, die zeigt, wie Compliance-Anforderungen in einem Prozess realisiert wurden.
	A21	Es soll eine Sicht entwickelt werden, die als Technik zur Wiederverwendung von Prozessstrukturen mit Bezug zu Compliance eingesetzt werden kann.

SOA als maßgeblicher Architekturstil für automatisierte Geschäftsprozesse (F1) und Web Services als aktuelle SOA-Technologie (F2) bedingen die Wahl eben dieser Kombination für die technische Realisierung (A5). Die weiteren erwähnten Feststellungen sind in Tabelle 2.2 aufgeführt.

Tabelle 2.2: Zusammenfassung der allgemeinen Feststellungen zum Rahmen dieser Arbeit

Nr.	Feststellung
F1	Der derzeit maßgebliche Architekturstil als Grundlage für die Automatisierung von Geschäftsprozessen ist die Serviceorientierte Architektur (SOA).
F2	Web Services sind eine aktuelle Technologie zur Realisierung einer SOA.
F3	Die für einen Prozess relevanten Gesetze, Regelungen und Verträge ändern sich im Laufe der Zeit und folglich verändern sich die an einen Prozess angelegten Compliance-Anforderungen.
F4	Compliance als ganzheitliche Aufgabe besteht nicht allein in der Prüfung von Regelkonformität, sondern auch aus Handlungen, um diese herzustellen.

Die aufgelisteten Anforderungen sind in ihrer Umsetzung eingeschränkt durch gewisse Voraussetzungen, welche in Tabelle 2.3 zusammengefasst sind.

Tabelle 2.3: Zusammenfassung der Voraussetzungen der entwickelten Ansätze.

Nr.	Voraussetzung
V1	Für einen Geschäftsprozess liegt ein Prozessmodell in einer graphbasierten Prozesssprache vor.
V2	Für einen Geschäftsprozess, dessen Regelkonformität unterstützt werden soll, besteht eine Menge von widerspruchsfreien Compliance-Anforderungen.

Teil II

Sichten auf Geschäftsprozesse

Überblick zu Teil II

In Teil II wird der Ansatz des musterbasierten Sichtenentwurfs vorgestellt.

In Kapitel 3 wird der Entwurf von Sichten auf Basis einer Mustersprache beschrieben. Die Mustersprache bringt bewährte Lösungen aus verschiedenen Ansätzen in eine einheitliche Form und macht sie für weitere Forschung und Verwendung leicht zugänglich (A6). Die Entwurfsmethode berücksichtigt eine schrittweise Verfeinerung sowie eine Wiederverwendung von Sichten oder Teilen daraus (A9). Die organisatorische Dimension wird mit einer Erweiterung des Prozesslebenszyklus berücksichtigt (A8). Ferner werden die bestehenden Möglichkeiten grafischer Darstellung in die Mustersprache einbezogen (A11).

In Kapitel 4 werden Prozesssichten auf Basis der Mustersprache und der vorgestellten Entwurfsmethode entworfen und entsprechend ihrer Eigenschaften klassifiziert (A7). Diese beispielhaften Sichten sind an Fragestellungen im Management von Geschäftsprozessen ausgerichtet und bieten unterschiedlichen Interessengruppen die Möglichkeit, Prozesse aufgabenangemessen darzustellen (A18).

In Kapitel 5 werden sprachübergreifende Sichten auf Geschäftsprozesse betrachtet und Sichtenentwürfe auf Basis der Mustersprache entwickelt (A6, A7). Instanzprojizierende Sichten werden als neuartige Methode des Monitorings von Prozessen vorgestellt (A12). Dieser Ansatz berücksichtigt im Besonderen die Anforderung, dass ein Geschäftsprozess auf unterschiedlichen Abstraktionsebenen bestehen kann (A1) und dass für diese Ebenen unterschiedliche Prozesssprachen verwendet werden können (A2).

KAPITEL 3

ENTWURF VON SICHTEN AUF GESCHÄFTSPROZESSE

In diesem Kapitel wird der Ansatz des musterbasierten Entwurfs von Sichten auf Geschäftsprozesse entwickelt. Zunächst wird der Einsatz von Mustern begründet und die Terminologie und eine Beschreibungsform für Sichtenentwurfsmuster und Sichtenentwürfe fixiert. Danach werden die Sichtenentwurfsmuster vorgestellt, die im Stand der Technik identifiziert wurden. Zur Einordnung von Sichtenentwürfen wird eine Klassifikation vorgestellt. Die Gesamtheit der identifizierten Sichtenentwurfsmuster bildet einen Musterkatalog, auf dessen Basis die Methode des musterbasierten Sichtenentwurfs entwickelt wird. Das Kapitel schließt mit einer Diskussion, welche Auswirkungen der Ansatz auf den Prozesslebenszyklus hat.

3.1 Motivation des musterbasierten Sichtenentwurfs

Ein Ansatz auf Basis von Mustern kann aus verschiedenen Gründen als geeignete Herangehensweise für die Problematik des Sichtenentwurfs betrachtet werden. Diese Gründe werden nachfolgend diskutiert.

Muster als bewährtes Mittel zur Wiederverwendung. Ein Muster ist explizit keine neue Erfindung, sondern stellt vielmehr das Resultat einer Identifizierung, Isolierung, Abstraktion und Beschreibung von bereits bestehenden, aber häufig impliziten Ideen oder Techniken dar. Im Stand der Technik wurde die Verwendung von Mustern im Bereich Geschäftsprozesse bereits beleuchtet, doch es gibt zahlreiche weitere Anwendungsfelder, in denen bewährte Ansätze strukturiert als Muster beschrieben und dadurch wiederverwendbar gemacht werden. Ein prominentes Beispiel sind die Architekturmuster für den Städtebau von Alexander et al. [AIS97], die eine Inspiration für viele andere Bereiche dargestellt haben. Musterbasierte Ansätze haben bisher im IT-Bereich die größte Verbreitung gefunden, wie bspw. die Muster für den Softwareentwurf nach Gamma et al. [GHJ95], die Muster zur Integration von Anwendungssystemen nach Hohpe und Woolf [HW03] und die Muster zur Erfassung und zum Vergleich der Ausdrucksstärke von Prozesssprachen von van der Aalst et al. [AHK+03]. Häufig werden Muster zueinander in Beziehung gesetzt und können ggf. zu komplexen Konstrukten kombiniert werden. Die Muster von Hohpe und Woolf lassen sich bspw. zu Integrationsszenarien kombinieren.

Sichten als Hilfsmittel zur Lösung eines Problems. Sichten betreffen eine Problemstellung einer Interessengruppe und sollen den für die Lösung des Problems erforderlichen Bedarf an Informationen decken. Mit Sichten, die auf spezifische Informationsbedarfe zugeschnitten sind, kann die inhärente Komplexität bestehender Prozesse je nach Bedarf individuell und temporär reduziert werden. Die bestehenden Ansätze verwenden formale oder programmiersprachenähnliche Formen zur Beschreibung der für eine Sicht erforderlichen Transformationen. Diese Ebene der Abstraktion hat den Nachteil, dass in diesen Formalismen entworfene Sichten kaum vergleichbar und nur schwer übertragbar sind. Eine rein textuelle Beschreibung ist hingegen i.d.R. zu unpräzise und zu unkonkret, um daraus eine Realisierung ableiten zu können. Muster stellen eine Ebene der Abstraktion dar, die zwischen der formalen und der rein textuellen Beschreibung liegt.

Muster für den Sichtenentwurf. Ein Sichtenentwurfsmuster löst eine für den Entwurf einer Sicht relevante Idee aus ihrem Kontext und beschreibt sie in einer einheitlichen Form. Die Musteridentifikation erfolgt durch eine Analyse von Literatur und Werkzeugen, die im Rahmen dieser Arbeit geleistet wurde, vgl. dazu deren Zusammenfassung in Kapitel 2.2. Durch den entstehenden Katalog an Mustern wird eine Vielfalt an Ideen bereitgestellt, die zu neuen Ansätzen kombiniert werden können. Die Herangehensweise des musterbasierten Sichtenentwurfs wurde erstmals von Schumm et al. [SLS10a] vorgeschlagen. Unterdessen werden auch im Prozesskontext Musterbeschreibungen eingesetzt, um bewährte komplexitätsreduzierende Techniken zu erfassen [RHW+11][RWM+11]. Diese Arbeiten verfolgen allerdings nicht wie der Ansatz der vorliegenden Dissertation das weiterführende Ziel der Entwicklung einer Mustersprache, auf deren Basis Sichten auf Prozesse durch Anwendung und Komposition der bereitgestellten Muster entworfen werden können.

Problemstellung beim musterbasierten Sichtenentwurf. Ein Musterkatalog bietet für sich alleine genommen nur eine eingeschränkte Hilfestellung für den Sichtenentwurf. Es konnten verschiedene Probleme identifiziert werden, die neben der Bereitstellung eines Musterkatalogs betrachtet werden müssen und den Rahmen des Sichtenentwurfs bilden:

1. Wie kann die Anwendung einzelner Muster zu einem Sichtenentwurf komponiert werden? Es bedarf daher einer Methode, mit der Sichten auf Basis der Muster eines Musterkatalogs *entworfen* werden können.
2. Wie kann die voraussichtliche Auswirkung des Einsatzes bestimmter Muster zur Entwurfszeit bestimmt werden? Bspw. beeinträchtigen strukturverändernde Muster ggf. die syntaktische Korrektheit. Es bedarf daher einer *Klassifikation* von Sichtenentwürfen, deren konkrete Ausprägung von den verwendeten Mustern abgeleitet wird.
3. Wie kann ein Sichtenentwurf in eine technische Realisierung überführt werden? Die Abstraktionsebene des Musteransatzes bedingt, dass auch die Sichtenentwürfe abstrakt sind und deren Realisierung auf eine konkrete technische Infrastruktur, eine konkrete Prozesssprache und auf konkrete Prozessmodelle abgestimmt und letztlich *implementiert* werden muss.

3.2 Konzeption des musterbasierten Sichtenentwurfs

Der Ansatz des musterbasierten Entwurfs erweitert den Stand der Technik von Sichten auf Prozesse um die Phase des Entwurfs. Diese Phase wird als Vorstufe zur Implementierung betrachtet, vgl. dazu die in Abbildung 3.1 dargestellte Vorgehensweise. Die Implementierung auf Basis von Diensten ist im musterbasierten Sichtenentwurf keine Notwendigkeit, sondern stellt lediglich eine der eingehender betrachteten Möglichkeiten zur Sichtenimplementierung dar. Für die Implementierung eines Sichtenentwurfs können auch andere Technologien oder Werkzeuge eingesetzt werden, sofern sie die im Sichtenentwurf verwendeten Muster unterstützen. Die Diskussion der technischen Realisierbarkeit in Teil IV berücksichtigt sowohl dienstbasierte als auch alternative Implementierungen von Sichtenentwürfen.

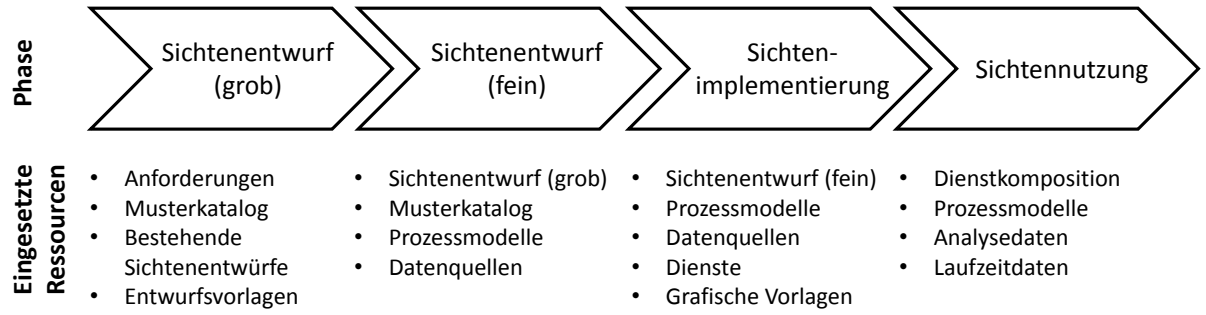


Abbildung 3.1: Vorgehensweise zum musterbasierten Entwurf und der dienstorientierten Implementierung von Sichten auf Geschäftsprozesse.

Vorgehensweise. Die Vorgehensweise zum musterbasierten Entwurf und der dienstorientierten Implementierung von Sichten auf Geschäftsprozesse sieht vier Phasen vor:

1. Sichtenentwurf (grob): Der Grobentwurf einer Sicht erfolgt auf Basis von Sichtenentwurfsmustern oder durch die Wiederverwendung und Anpassung bestehender Sichtenentwürfe oder Entwurfsvorlagen. Sie ist vergleichbar mit der Erstellung eines Lastenhefts zur Spezifikation von Anforderungen an ein Informationssystem.
2. Sichtenentwurf (fein): Der Feinentwurf erfolgt durch eine konkretere Beschreibung durchzuführender Transformationen und anzubindender Datenquellen. Im Feinentwurf werden Anpassungen am bestehenden Entwurf unter Berücksichtigung der zu unterstützenden Prozessmodelle vorgenommen. Diese Phase ist vergleichbar mit der Erstellung eines Pflichtenhefts und dessen Abstimmung mit dem Auftraggeber.
3. Sichtenimplementierung: Eine dienstbasierte Implementierung erfolgt durch Auswahl, Parametrierung, Komposition und Bereitstellung von Diensten. Ferner muss eine Anbindung von Datenquellen oder eine anderweitige Form der Datenbereitstellung erfolgen. Außerdem ist die Auswahl, Beschaffung und Konfiguration von grafischen Vorlagen für die Darstellung erforderlich. Diese Phase ist vergleichbar mit der Umsetzung des in einem Pflichtenheft spezifizierten Informationssystems.
4. Sichtennutzung: Die Nutzung der Implementierung erfolgt durch Auswahl eines in einer Sicht darzustellenden Prozessmodells, der einzubeziehenden Analyse- oder Laufzeitdaten und dem Aufruf der Dienstkomposition mit individuellen Parametern.

Konzeptionelles Modell. In diesem Abschnitt wird die Terminologie eingeführt, mit der die Konzepte rund um den Entwurf von Prozesssichten beschrieben werden. Abbildung 3.2 zeigt das konzeptionelle Modell, das dem musterbasierten Sichtenentwurf zugrunde gelegt wird. Die Darstellung erfolgt in Form eines abstrakten UML Klassendiagramms [OMG11a] mit zusätzlicher Kennzeichnung der Semantik von Beziehungen, wobei ein Pfeil die Leserichtung angibt. Diese Form der Darstellung wird auch für die übrigen konzeptionellen Modelle in der vorliegenden Dissertation verwendet. Das konzeptionelle Modell und die damit verbundenen Begrifflichkeiten basieren auf einer vorangehenden Arbeit zu Sichtenentwurfsmustern, vgl. Schumm et al. [SLS10a]. Die Begriffe des konzeptionellen Modells werden im Folgenden definiert und in den nachfolgenden Abschnitten näher erläutert und vertieft.

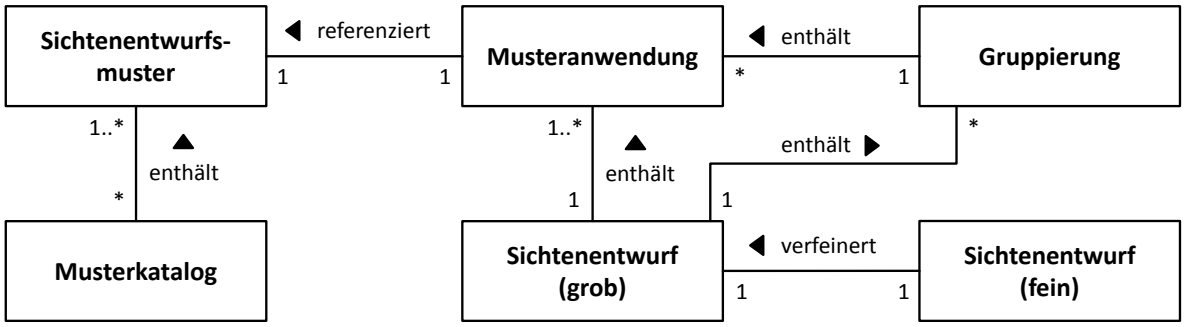


Abbildung 3.2: Konzeptionelles Modell zum musterbasieren Entwurf von Sichten auf Geschäftsprozesse.

Definition 3: Ein *Sichtenentwurfsmuster* ist eine zur Beschreibung von Sichten geeignete Abstraktion von Transformationstechniken in einer einheitlichen Beschreibungsform.

Definition 4: Ein *Musterkatalog* ist eine Zusammenstellung von Mustern.

Definition 5: Eine *Mustersprache* umfasst einen Musterkatalog und die Beziehungen zwischen den darin enthaltenen Mustern.

Definition 6: Eine *Musteranwendung* stellt die konkrete Verwendung eines Musters im Kontext eines spezifischen Problems dar. Die Anwendung eines Sichtenentwurfsmusters beschreibt folglich, in welcher konkreten Ausprägung und auf welchen Anwendungsbereich eines Prozesses die im referenzierten Sichtenentwurfsmuster beschriebene Transformation angewendet werden soll.

Definition 7: Ein *Sichtenentwurf* beschreibt eine spezifische Anwendung und Komposition von Sichtenentwurfsmustern. Ein feiner Sichtenentwurf verfeinert einen groben Sichtenentwurf. Die Reihenfolge der Musteranwendungen in einer Komposition kann sequentiell in Form einer Mustersequenz oder in Form eines zyklenfreien Mustergraphen beschrieben werden, in welchem die Abhängigkeiten zwischen den Musteranwendungen dargestellt sind.

Definition 8: Eine *Gruppierung* fasst ein Segment einer Mustersequenz oder analog einen Teilgraphen eines Mustergraphen zusammen. Sie dient als Mittel der Modularisierung und zur Einschränkung des Gültigkeitsbereichs von Vorgaben an die Durchführung von Musteranwendungen.

Abbildung 3.3 stellt weitere Begrifflichkeiten anhand einer Skizze zu einem Sichtenentwurf beispielhaft dar. Im Vergleich zur Beschreibung des PM-Graphen (vgl. Definition 1) sind die nachfolgenden Begrifflichkeiten abstrakt beschrieben und verzichten bewusst auf eine Formalisierung. Die zahlreichen Möglichkeiten der Ausgestaltung bleiben dadurch erhalten. Bspw. würde eine exakte Transformationsvorschrift einschränken, wie eventuell auftretende Mehrdeutigkeiten beim Entfernen einer Aktivität aufzulösen wären. Auch würde eine formale Definition einer aggregierten Struktur bereits festlegen, ob und ggf. inwiefern sich diese von einer normalen Aktivität unterscheidet. Für die tatsächliche Realisierung einer Sicht für konkrete Prozessmodelle in einer festgelegten Prozesssprache müssen derartige Festlegungen getroffen werden, jedoch nach dem Verständnis der vorliegenden Dissertation nicht für den *Entwurf* einer prozessdarstellenden Sicht.

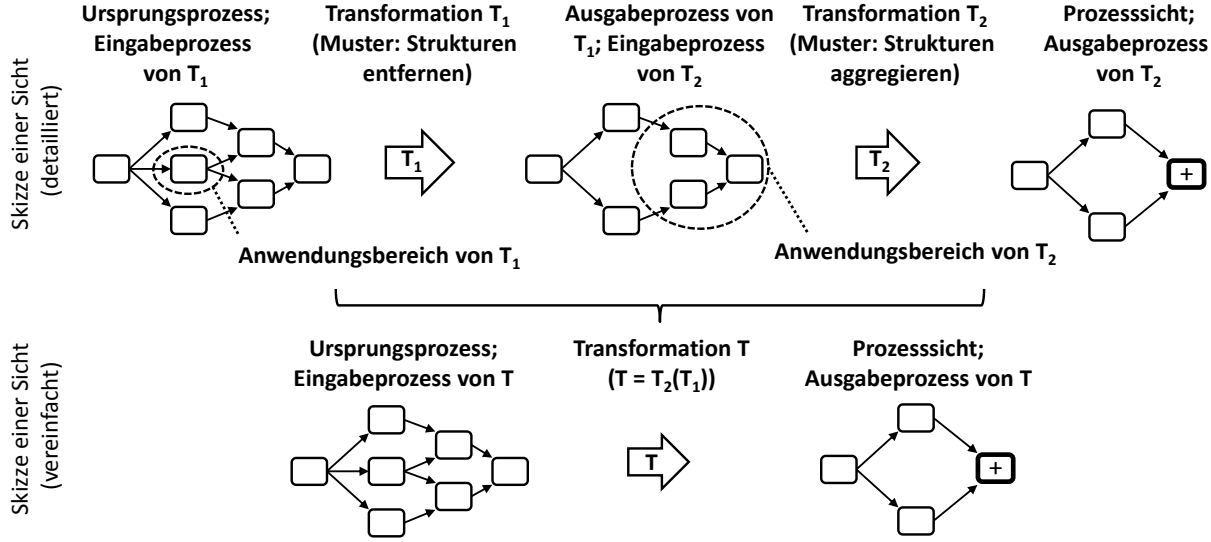


Abbildung 3.3: Begriffe im musterbasieren Entwurf von Sichten auf Geschäftsprozesse.

Eine *detaillierte Skizze* einer Sicht zeigt die vollständige Sequenz der Anwendungen von Transformationen auf einen Prozess. Der Begriff Transformation stellt einen Oberbegriff dar, der die in Mustern erfassten Arten der Transformationen, deren Komposition, als auch spezifische Arten der Transformation einschließt. Jede Transformation innerhalb dieser Sequenz erhält einen Eingabeprozess, den sie transformiert. Sie produziert einen Ausgabeprozess, der das Resultat der Transformation darstellt. Der Eingabeprozess der ersten Transformation wird auch Ursprungsprozess genannt (engl. *base process*, *original process*). Der Ausgabeprozess der letzten Transformation wird auch Prozesssicht oder Sicht genannt (engl. *process view*, *view*). Eine Transformation hat einen Anwendungsbereich, der festlegt, auf welche Strukturen die Transformation angewendet soll.

Eine *vereinfachte Skizze* einer Sicht fasst die einzelnen Transformationen $T_1..T_n$ in einer Verkettung bzw. Hintereinanderausführung zusammen. Die unterschiedlichen Anwendungsbereiche der Transformationen werden in einer vereinfachten Skizze nicht dargestellt.

In Skizzen werden grafische Elemente zur Veranschaulichung der Anwendung von Transformationen eingesetzt. Diese Elemente sind in Abbildung 3.4 dargestellt.

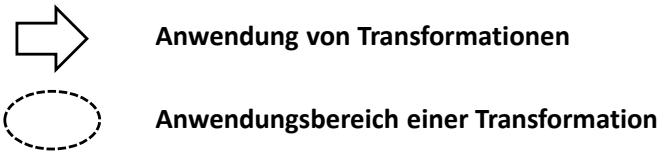


Abbildung 3.4: Elemente zur Veranschaulichung der Anwendung von Transformationen, vgl. [SLS10a].

Zur Skizzierung von Transformationen ist eine *Erweiterung* der grafischen Notation zur Darstellung von Prozessskizzen (siehe Abschnitt 2.1.4) erforderlich, um zwischen ursprünglichen und transformierten Elementen unterscheiden zu können, siehe Abbildung 3.5 (b).

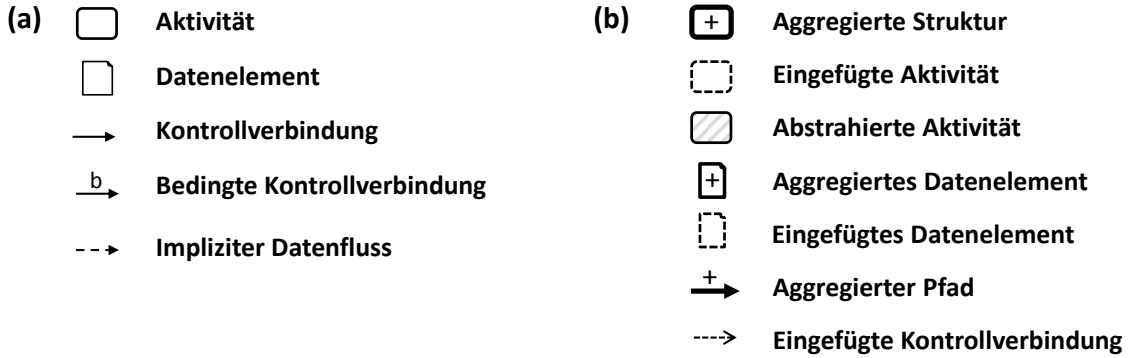


Abbildung 3.5: Elemente der grafischen Notation zur Darstellung von Prozessskizzen (a) und Erweiterungen zur Unterscheidung von ursprünglichen und transformierten Elementen (b), vgl. [SLS10a].

Eine *aggregierte Struktur* ist eine Aggregation nicht notwendigerweise verbundener Aktivitäten und Kontrollverbindungen. Eine *eingefügte Aktivität* ist eine Aktivität, die im Eingabeprozess der Transformation nicht enthalten war. Eine *abstrahierte Aktivität* stellt eine Aktivität dar, deren Eigenschaften durch die Anwendung einer Transformation modifiziert wurden. Ein *aggregiertes Datenelement* ist eine Aggregation von mehreren Datenelementen. Ein *eingefügtes Datenelement* ist ein Datenelement, das in dieser Form im Eingabeprozess nicht enthalten war. Ein eingefügtes Datenelement kann etwa durch die Zerlegung eines Datenelements oder auch durch das Zerlegen von Aktivitäten oder Pfaden entstehen. Ein *aggregierter Pfad* fasst verbundene Aktivitäten, ausschließlich innerhalb des Pfades verwendete Datenelemente und Kontrollverbindungen zu einer komplexen Kontrollverbindung zusammen. Eine *eingefügte Kontrollverbindung* dient der Unterscheidung zwischen bestehenden und neu hinzugekommenen Kontrollverbindungen.

Ein Beispiel der Transformation eines PM-Graphen in einen erweiterten PM-Graphen ist in Abbildung 3.6 dargestellt. Die komposite Transformation *S* …

- … abstrahiert Aktivität *A*,
- aggregiert die Aktivitäten *B*, *C* und *D*,
- aggregiert die Datenelemente *1* und *2*,
- aggregiert den Pfad *A*→*E*→*F* zum aggregierten Pfad *A*→*F* und
- fügt die Aktivität *H* hinzu.

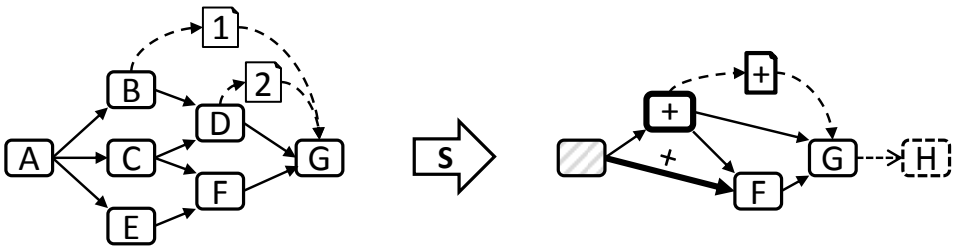


Abbildung 3.6: Beispiel der Transformation eines PM-Graphen in einen erweiterten PM-Graphen.

Zur Skizzierung von Transformationen unter Einbeziehen von Laufzeitinformationen ist eine Erweiterung des eingeführten Metamodells von PI-Graphen erforderlich. Die Menge möglicher Aktivitätszustände eines PI-Graphen wird um die Zustände *Zusammengesetzt* und *Fortgeschritten* erweitert, vgl. Abbildung 3.7 (b). Ein zusammengesetzter Zustand ist eine Menge von Aktivitätszuständen, die entweder durch Anreicherung einer Aktivität mit mehr als einem Aktivitätszustand oder durch die Aggregation mehrerer zustandsbehafteter Aktivitäten entsteht. Ein fortgeschrittener Zustand aggregiert eine Menge von Aktivitätszuständen zu einem teilerledigten Zustand.

Während im Metamodell von PI-Graphen bei Kontrollverbindungen lediglich zwischen *nicht ausgewertet*, *wahr* und *falsch* unterschieden wird, sind bei erweiterten PI-Graphen, hauptsächlich durch die Semantik aggregierter Pfade, weitere Kontrollverbindungszustände möglich, vgl. Abbildung 3.7 (c). Deren Semantik ist analog zur Semantik von aggregierten Strukturen.

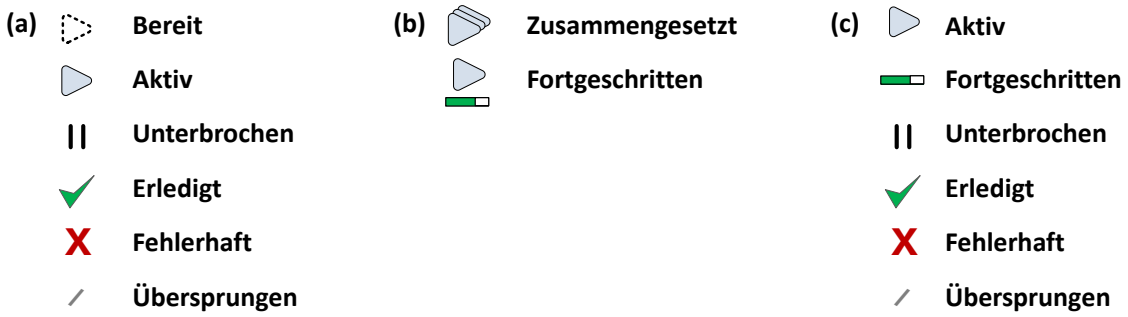


Abbildung 3.7: Aktivitätszustände (a) im PI-Graphen, (b) sichtenbezogene Erweiterungen der Aktivitätszustände und (c) sichtenbezogene Erweiterungen der Kontrollverbindungszustände.

Ein Beispiel der Transformation von einem PM-Graphen über einen PI-Graphen in einen erweiterten PI-Graphen ist in Abbildung 3.8 dargestellt. Die Transformation S_1 reichert die Aktivitäten mit Zuständen an und bildet so den Zustand einer Prozessinstanz auf den PM-Graphen ab. Transformation S_2 verfährt analog zum vorigen Beispiel, bezieht allerdings die nun vorhandenen Laufzeitinformationen mit ein.

Transformation S_2 …

- … abstrahiert Aktivität *A*,
- aggregiert die Aktivitäten *B*, *C* und *D* und weist der aggregierten Struktur den Zustand *Fortgeschritten* zu (alternativ wäre auch ein zusammengesetzter Zustand möglich),

- aggregiert die Datenelemente *1* und *2*,
- aggregiert den Pfad *A→E→F* zum aggregierten Pfad *A→F* und weist ihm den Zustand *Erledigt* zu und
- fügt die Aktivität *H* hinzu.

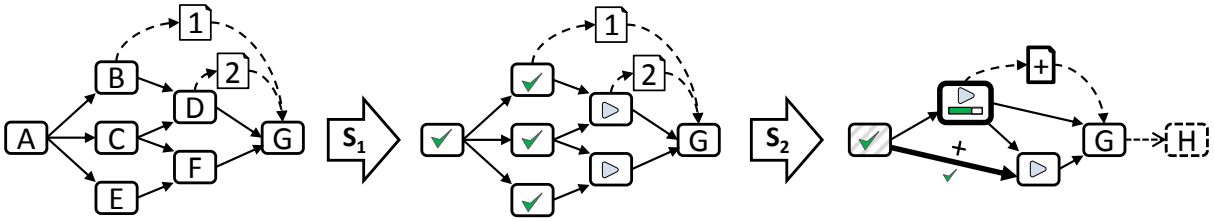


Abbildung 3.8: Beispiel eines erweiterten PI-Graphen.

3.3 Beschreibungsformen im musterbasierten Sichtenentwurf

In diesem Abschnitt wird die gewählte Beschreibungsform für Sichtenentwurfsmuster vorgestellt und von den gängigen Formen der Musterbeschreibung abgegrenzt. Anschließend wird die gewählte Beschreibungsform für Sichtenentwürfe erläutert.

3.3.1 Beschreibungsform für Sichtenentwurfsmuster

Die vorgeschlagene Beschreibungsform von Sichtenentwurfsmustern ist an Beschreibungsformen von Mustern orientiert, die im Softwareentwurf (Gamma et al. [GHJ95]) oder in der Architektur (Alexander et al. [AIS97]) verwendet werden. Sie umfasst verschiedene Bestandteile: *Name*, *Nummer*, *Symbol*, *Problem*, *Beschreibung*, *Skizze*, *Beispiele*, *Schlagwörter*, *Beziehungen*, *Literaturhinweise* und *Kategorie*. Der *Name* unterstützt die sprechende und eindeutige, die *Nummer* die abgekürzte und das *Symbol* die visuelle Identifikation eines Musters. Ein Sichtenentwurfsmuster hat eine wesentliche Fragestellung (*Problem*), zu der es einen Lösungsansatz beschreibt. Die *Beschreibung* steht für eine textuelle Beschreibung der Wirkung der Transformation. Eine *Skizze* zur Veranschaulichung der Anwendung eines Musters wird auf Basis erweiterter PM- oder PI-Graphen angegeben. *Beispiele* erläutern verwandte Arbeiten oder Werkzeuge, in denen ein Muster in einem konkreten Kontext und ggf. auf eine spezifische Prozesssprache wie bspw. BPMN oder BPEL angewendet wird. *Schlagwörter* umfassen Synonyme und themenverwandte Begriffe und dienen zur Unterstützung der Suche nach einem Muster. Die vielfältigen *Beziehungen* zu anderen Mustern werden innerhalb der Beschreibung der Muster erläutert und sind in Abschnitt 3.4.5 tabellarisch zusammengefasst. *Literaturhinweise* finden sich in der *Beschreibung* und bei den *Beispielen*.

Sichtenentwurfsmuster werden in vier *Kategorien* gegliedert. Dazu wird eine Farbcodierung verwendet (siehe Abbildung 3.9), die aus Konsistenzgründen analog zur Semantik der Farbcodierung von Architekturdiagrammen gestaltet ist (vgl. Abschnitt 2.1.2).

Sichtenentwurfsmuster einer bestimmten Kategorie dienen einem bestimmten *Zweck*:

- Die Kategorie *Strukturveränderung* wird rot dargestellt, analog zur teil- oder vollautomatischen Ausführung. Eine Strukturveränderung bezweckt eine Anpassung des Grades an Abstraktion. Reduzierende Strukturveränderungen wirken abstrahierend, erweiternde Strukturveränderungen wirken konkretisierend.
- Die Kategorie *Informationsanreicherung* wird grün dargestellt, analog zur Datenhaltung und -verwaltung. Eine Informationsanreicherung bezweckt eine Erweiterung der für eine Sichtenerzeugung nutzbaren Menge an Informationen.
- Die Kategorie *Darstellungsanpassung* wird blau dargestellt, analog zum wesentlichen Bezug zum Endanwender. Eine Darstellungsanpassung bezweckt die Festlegung oder Modifikation der Art und Weise, wie ein Prozess grafisch dargestellt werden soll.
- Die Kategorie *Vorgabe* wird orange dargestellt; für diese Kategorie besteht keine Analogie zur Farbcodierung in Architekturen. Eine Vorgabe bezweckt die Einschränkung von Transformationen durch Festlegung einzuhaltender Vorgaben.

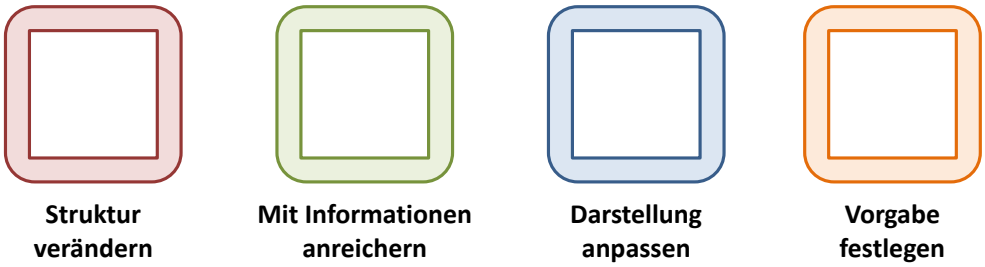


Abbildung 3.9: Unterscheidung der Musterkategorien durch Farbcodierung der Rahmenfarbe der Symbole.

Die Informationsbausteine *Kategorie*, *Name*, *Nummer* und *Symbol* werden, wie in Abbildung 3.10 dargestellt, zusammengesetzt. Die Nummer und der Name des Sichtenentwurfsmusters finden sich als Symbolunterschrift; die Farbcodierung der Kategorie ist in einem äußeren Rahmen untergebracht. Innerhalb des Rahmens wird das musterbeschreibende Symbol platziert. Die in dieser Arbeit verwendeten Symbole wurden selbst erstellt; sie dienen der Veranschaulichung, haben keinen normativen Charakter und erheben nicht den Anspruch, nach kognitiven Gesichtspunkten gestaltet zu sein.

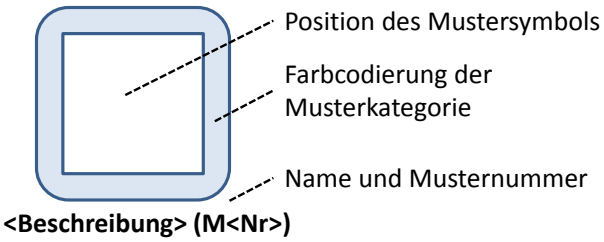


Abbildung 3.10: Informationszusammensetzung in den Symbolen der Sichtenentwurfsmuster.

3.3.2 Beschreibungsform für Sichtenentwürfe

Ein Problem im Kontext von Sichten auf Geschäftsprozesse ist eine bestimmte prozessbezogene und ggf. rollen- oder interessengruppenspezifische Fragestellung oder Aufgabe. Ein solches Problem wird lösbar, indem der zur Bewältigung des Problems erforderliche Bedarf an Informationen gedeckt wird. Die Abdeckung dieses Informationsbedarfs (Problem) wird durch die tatsächliche Realisierung einer Sicht auf einen konkreten Prozess erreicht (Lösung). Die spezifische Anwendung und Komposition von Sichtenentwurfsmustern zu einem Sichtenentwurf beschreibt in abstrakter und technologieunabhängiger Form, wie diese Sicht erzeugt werden kann (Lösungsansatz).

Die gewählte Beschreibungsform eines Sichtenentwurfs umfasst daher die Beschreibung eines *Problems* und einen sowohl textuell als auch auf Basis von angewendeten Sichtenentwurfsmustern beschriebenen *Lösungsansatz*. Zur Veranschaulichung des Lösungsansatzes enthält ein Sichtenentwurf außerdem eine *Skizze* sowie *Beispiele* ähnlicher Ansätze aus Werkzeugen und verwandten Arbeiten. Die *Klassifikation* eines Sichtenentwurfs zeigt die Konsequenzen, die sich aus dem Lösungsansatz ergeben.

Auch diese Beschreibungsform ähnelt der im Softwareentwurf [GHJ95] gängigen Beschreibungsform eines Musters und ein Sichtenentwurf kann durchaus als bewährte Lösung für ein wiederkehrendes Problem verstanden werden. Um allerdings die Begrifflichkeiten voneinander abzugrenzen, wird in der vorliegenden Dissertation nur von Sichtenentwürfen (und nicht von Mustern) gesprochen, während Sichtenentwurfsmuster die darin verwendeten Techniken bezeichnen.

3.4 Musterkatalog für den Sichtenentwurf

In diesem Abschnitt werden die Muster vorgestellt, die im Rahmen der vorliegenden Dissertation identifiziert wurden. Dieser Musterkatalog stellt eine Erweiterung und Verfeinerung der von Schumm et al. [SLS10a] vorgestellten Sichtenentwurfsmuster dar. Abbildung 3.11 zeigt die Kategorisierung der Sichtenentwurfsmuster im Musterkatalog. Die Kategorien *Strukturveränderung* und *Darstellungsanpassung* sind thematisch untergliedert. Eine kompakte Zusammenfassung des Musterkatalogs befindet sich als Nachschlagehilfe im Anhang (Abschnitt 14.2). In Verweisen auf die Muster dieses Katalogs wird eine abgekürzte Schreibweise verwendet, bspw. steht (M1) als Verweis auf das Muster Nr. 1.

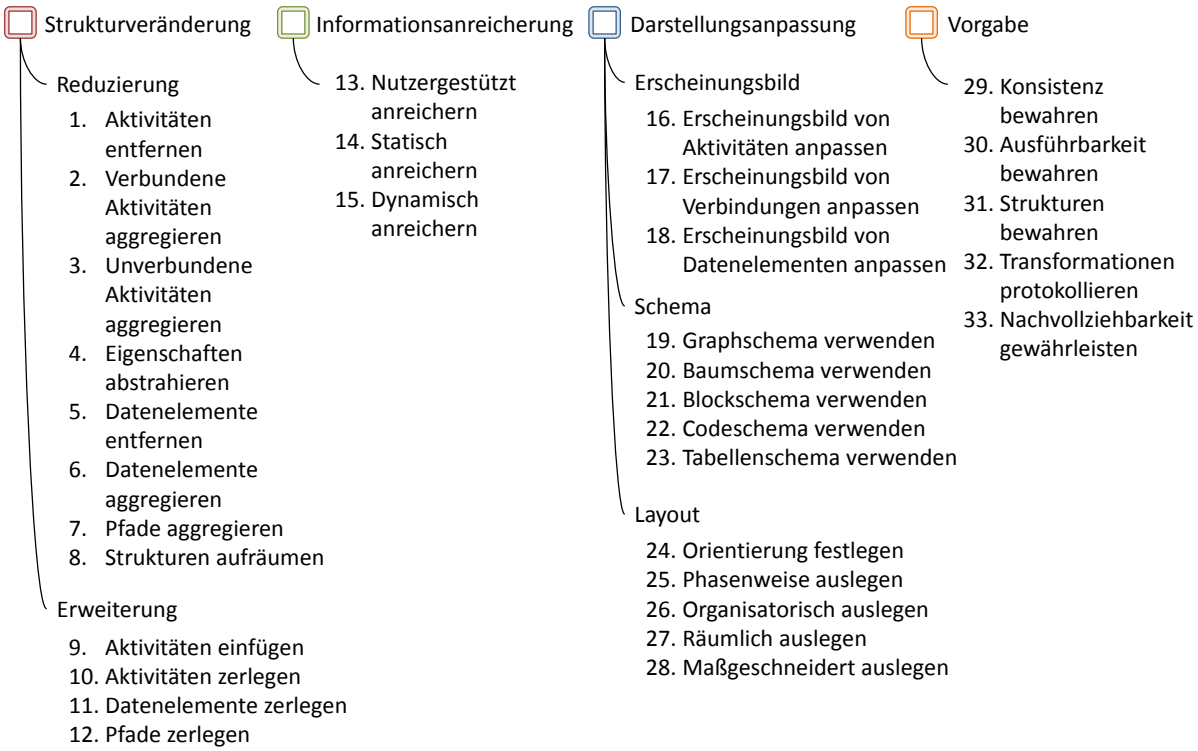


Abbildung 3.11: Kategorisierung und Untergliederung der Sichtenentwurfsmuster im Musterkatalog.

Die Kategorisierung abstrahiert davon, wie Sichtenentwurfsmuster tatsächlich implementiert werden. Bspw. kann ein Entfernen von Datenelementen (M5) unmittelbar auf oder nahe dem physischen Prozessmodell, in der Darstellungsschicht eines Werkzeugs oder in einer Zwischenschicht erfolgen. Die Abgrenzung der Strukturveränderung von der grafischen Darstellung eines Modells (engl. *rendering*) ist daher abhängig von der konkreten Umsetzung.

Abbildung 3.12 bietet eine alternative Übersicht. Sie berücksichtigt die in der vorliegenden Dissertation ausgearbeitete Empfehlung zur Bildung einer Sequenz von Musteranwendungen, angefangen mit Informationsanreicherung und anschließender Strukturveränderung. Abschließend findet eine Festlegung der Darstellung statt. Die Musteranwendungen können zudem mit einzuhaltenden Vorgaben eingeschränkt werden. Die Übersicht ist nach dem Grundverständnis eines Fließbandes gestaltet, auf dem ein Eingabeprozess an mehreren Stationen bearbeitet, d.h. transformiert wird und an dessen Ende die fertige Prozesssicht steht.

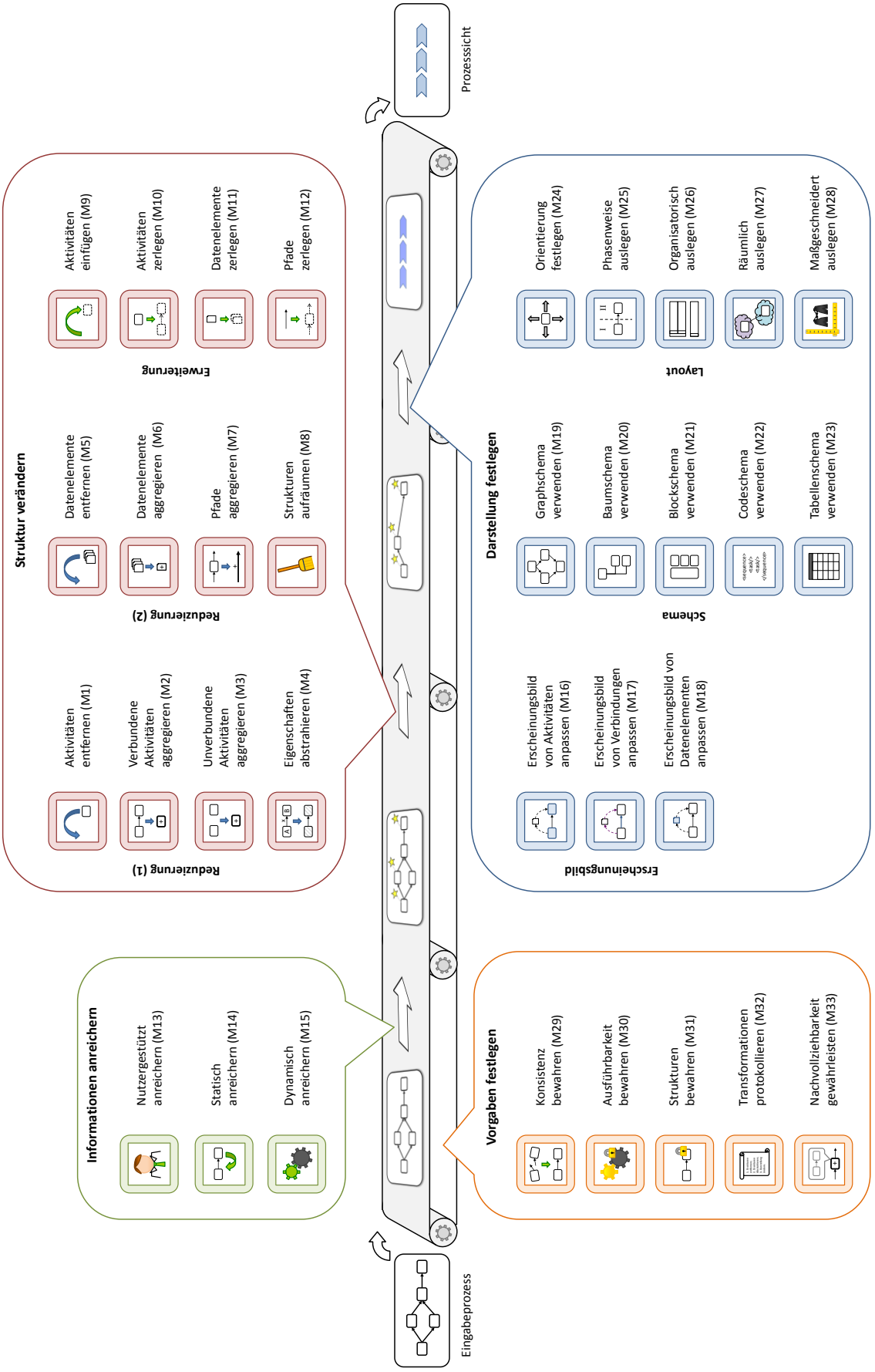


Abbildung 3.12: Übersicht des Musterkatalogs nach dem Verständnis eines Fließbandes, auf dem ein Eingabeprozess an verschiedenen Stationen bearbeitet werden kann und schrittweise eine Prozesssicht angefertigt wird.

3.4.1 Muster zur Strukturveränderung

Die Kategorie Strukturveränderung gliedert sich in die Unterkategorien *Reduzierung* und *Erweiterung*. Durch die Unterkategorie der Erweiterung unterscheidet sich der Begriff einer Sicht vom Begriff der reinen Abstraktion. In Sichten können nach dem Verständnis der vorliegenden Dissertation Informationen nicht nur abstrahiert oder entfernt, sondern auch hinzugefügt werden. Dadurch kann der Grad an Abstraktion in beide Richtungen angepasst werden, Abstraktion und Konkretisierung. Durch die Konkretisierung wird der Informationsgehalt bestimmter Aspekte erhöht und das Prozessmodell stärker an die Realität angenähert.

3.4.1.1 Reduzierung

Muster 1: Aktivitäten entfernen

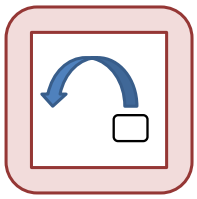


Abbildung 3.13: Symbol zu „Aktivitäten entfernen (M1)“.

Problem: Wie kann die Anzahl an Aktivitäten reduziert werden?

Schlagwörter: Ausblenden; Filtern; Auslassen; Löschen; Verbergen; Verstecken.

Beschreibung: Bei „Aktivitäten entfernen“ werden die Aktivitäten im Anwendungsbereich der Musteranwendung aus dem Prozessgraphen entfernt. Das Entfernen von Aktivitäten kann nahe an der Darstellung erfolgen, was dem Begriff des Ausblendens gleichkommt, oder nahe am physischen Modell des Prozesses geschehen, wodurch die Aktivität dauerhaft aus dem Modell entfernt wird. I.d.R. gilt eine Vorgabe der Konsistenzbewahrung (M29) bei der Entfernung, um sonst auftretende Inkonsistenzen oder den Zerfall eines Prozessgraphen in unverbundene Fragmente zu vermeiden. Abbildung 3.14 skizziert das Entfernen unter der Vorgabe der Konsistenzbewahrung (M29) wodurch transitive Kontrollverbindungen aufrechterhalten werden.

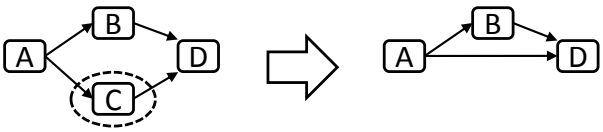


Abbildung 3.14: Skizze zu „Aktivitäten entfernen (M1)“.

Beim Entfernen von Aktivitäten können Mehrdeutigkeiten auftreten, d.h. es gibt mehrere Möglichkeiten, wie die aus der Transformation resultierende Struktur aufgebaut sein kann. Eine Auflösung der Mehrdeutigkeit einer Transformation kann entweder durch eine entsprechende Implementierung automatisch erfolgen oder einen Nutzer in Form von Rückfragen oder Voreinstellungen einbinden.

Durch das Entfernen von Aktivitäten besteht die Möglichkeit, dass im Ausgabeprozess Datenelemente enthalten sind, die von keiner der vorhandenen Aktivitäten verwendet werden. Diese Überreste können durch ein Aufräumen (M8) entfernt werden. Das Aufräumen kann außerdem redundante Kontrollverbindungen entfernen, die beim Entfernen von Aktivitäten entstehen können, vgl. bspw. die Kontrollverbindung $A \rightarrow D$ in Abbildung 3.14.

Beispiele: Abbildung 3.15 zeigt ein abstraktes Beispiel einer Mehrdeutigkeit beim Entfernen einer Aktivität die als Synchronisationspunkt von zwei Pfaden dient. Die Transformationen T_1, T_2 und T_3 stellen verschiedene Möglichkeiten dar wie dieses Muster angewendet werden kann. T_1 löst den Synchronisationspunkt durch eingefügte Kontrollverbindungen auf. Dies führt bei Vorhandensein von bedingten Kontrollverbindungen allerdings zu komplexen Ausdrücken. Zur Reduzierung der Komplexität trägt diese Form der Auflösung daher nur eingeschränkt bei. T_2 entfernt so gut wie möglich alle Informationen über die zu entfernende Aktivität. T_3 ersetzt die zu entfernende Aktivität durch eine leere Aktivität, d.h. durch eine Aktivität vergleichbar mit einem No Operation Ausdruck (NOP).

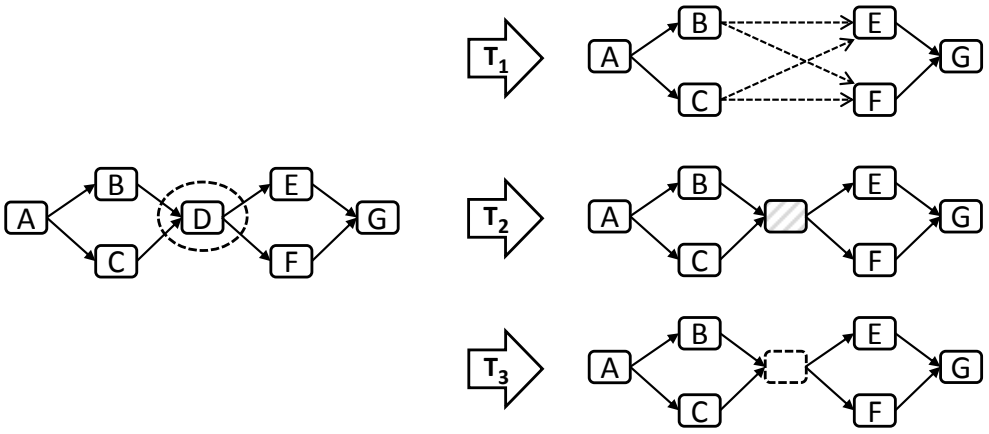


Abbildung 3.15: Beispiel für eine Mehrdeutigkeit beim Entfernen von Aktivitäten.

Das Entfernen von Aktivitäten kann durch verschiedene Aspekte gesteuert werden, welche die Relevanz für eine bestimmte Interessengruppe und Aufgabenstellung widerspiegeln. Der Aktivitätstyp kann als Kriterium eingesetzt werden, um zwischen wichtigen und unwichtigen Aktivitäten zu unterscheiden. In dem Werkzeug [Ora10] zum Monitoring von BPEL-Prozessen, wie auch in der Patentanmeldung [GHN+09] können z.B. Aktivitäten vom Typ `<assign>` ausgeblendet werden. Dies ist ein Aktivitätstyp in BPEL zur Variablenzuweisung.

Neben Informationen, die ohnehin im Prozessmodell enthalten sind, können auch angereicherte Informationen als Filterkriterien eingesetzt werden (M13, M14, M15). Bobrik [Bob08] setzt Laufzeitinformationen über den Aktivitätszustand (M15) dazu ein, übersprungene Aktivitäten zu entfernen (M1).

Muster 2: Verbundene Aktivitäten aggregieren

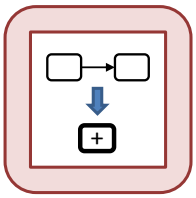


Abbildung 3.16: Symbol zu „Verbundene Aktivitäten aggregieren (M2)“.

Problem: Wie kann die Anzahl an Aktivitäten reduziert werden, ohne Informationen gänzlich zu verlieren?

Schlagwörter: Zusammenfassen; Gruppieren; Clustern; Verdichten.

Beschreibung: Aus mehreren, direkt verbundenen Aktivitäten wird eine aggregierte Struktur gebildet. Die aggregierte Struktur ist nach der Transformation die Quelle aller Kontrollverbindungen, die im Eingabeprozess von den einzelnen zu aggregierenden Aktivitäten ausgehen. Analog ist die aggregierte Struktur nach der Transformation das Ziel aller Kontrollverbindungen, die im Eingabeprozess zu aggregierende Aktivitäten als Ziel haben. Abbildung 3.17 skizziert die Aggregation der Aktivitäten *B* und *C*.

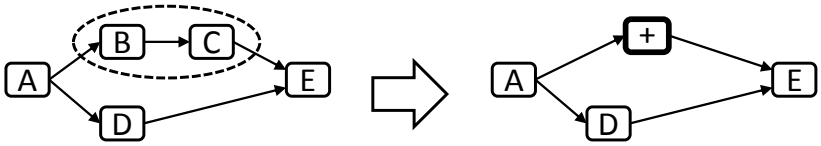


Abbildung 3.17: Skizze zu „Verbundene Aktivitäten aggregieren (M2)“.

Das Aggregieren der Eigenschaften der zusammenzufassenden Elemente ist ein zentraler Aspekt dieser Transformation. Die Bestimmung einer treffenden, d.h. semantisch annähernd äquivalenten, aber abstrakteren Bezeichnung von aggregierten Aktivitäten, ist bisher ungelöst. Zu dieser, auch als Labeling Problem bekannten Fragestellung gibt es bisher nur wenige Forschungsansätze, vgl. bspw. Maqbool und Babri [MB06] oder Leopold et al. [LMR11]. Die derzeit praktikable Lösung für dieses Problem besteht darin, die Bezeichnung von aggregierten Strukturen durch einen Anwender vorgeben zu lassen [Bob08], d.h. es bedarf der Einbindung von Prozesswissen und der Abstraktionsfähigkeit des Menschen (M13).

Quantitative Eigenschaften können hingegen automatisch aggregiert werden. Je nach Kontext und Aufgabenstellung können Eigenschaften summiert werden, ein arithmetisches Mittel oder der Median daraus gebildet werden oder sie können in Listenform aneinandergereiht werden. Bobrik [Bob08] gibt außerdem das Minimum und Maximum, die Anzahl, erster Wert, letzter Wert, Zufallswert, häufigster Wert und seltenster Wert als Möglichkeit zur Bestimmung eines aggregierten Wertes an, wobei die Wahl der Funktion aufgaben- und eigenschaftsspezifisch gewählt wird. Zur Aggregation von Laufzeitinformationen wie bspw. der Aktivitätszustände kann eine Abbildung definiert werden, mit der der Zustand einer aggregierten Struktur auf Basis der Zustände der darin zusammengefassten Aktivitäten bestimmt werden kann. An dieser Stelle zeigt sich außerdem die Notwendigkeit der

Erweiterung des Metamodells von PI-Graphen um die zusätzlichen Zustände *Zusammengesetzt* und *Fortgeschritten*.

Ein Phänomen, das bei der Aggregation von Aktivitäten auftreten kann, ist das Entstehen von Zyklen im Kontrollfluss, wie Abbildung 3.18 skizziert. Die Kontrollverbindungen *A→D* und *D→C* bewirken in einer Aggregation der Aktivitäten *A*, *B* und *C* einen Zyklus zwischen der aggregierten Struktur und der Aktivität *D*. Manche Prozesssprachen erfordern die Freiheit von Kontrollflusszyklen, welche durch diese Transformation verletzt wäre. Für die bloße Darstellung einer Sicht auf einen Prozesses ist dies unproblematisch. Sofern jedoch die syntaktische Korrektheit bzw. die Zyklenfreiheit bewahrt werden soll (M29), kann eine derartige Aggregation nicht durchgeführt werden.

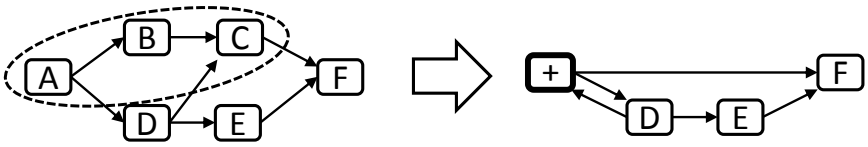


Abbildung 3.18: Beispiel für die Entstehung von Zyklen bei der Aggregation von Aktivitäten.

Beispiele: Vanhatalo et al. [VVL07] nutzen die Aggregation verbundener Aktivitäten mit genau einer eingehenden und genau einer ausgehenden Kontrollverbindung zur Beschleunigung von Kontrollflussanalysen. Die Aggregation von tief verschachtelten Blockstrukturen kann zudem für einen menschlichen Betrachter hilfreich sein, indem es zur Reduzierung der Komplexität beiträgt, vgl. [Lat10][SLL11].

Muster 3: Unverbundene Aktivitäten aggregieren

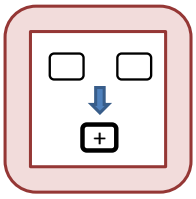


Abbildung 3.19: Symbol zu „Unverbundene Aktivitäten aggregieren (M3)“.

Problem: Wie können thematisch zusammengehörige Aktivitäten zu einer größeren Einheit zusammengefasst werden?

Schlagwörter: Zusammenfassen; Gruppieren; Clustern; Verdichten.

Beschreibung: Mehrere, nicht unmittelbar verbundene Aktivitäten werden aggregiert. In Abgrenzung zur Aggregation verbundener Aktivitäten (M2) kann der Anwendungsbereich der Transformation beliebige Teilgraphen eines Prozessgraphen zu einer komplexen Struktur aggregieren. Wie in Abbildung 3.20 skizziert, können dadurch ungewöhnliche Kontrollstrukturen entstehen. In der Darstellungsform eines Graphen (M19) kann diese Aggregation den Betrachter verwirren, in Form einer Tabelle hingegen (M23) können Aktivitäten gut nach verschiedenen, auch aggregierten Eigenschaften gruppiert und gegliedert werden.

Ohne die Vorgabe der Konsistenzbewahrung (M29) kann die Freiheit von Kontrollflusszyklen nicht gewährleistet werden. Ferner kann die Eigenschaft verletzt werden, dass die Menge von Kontrollverbindungen unifiziert ist, d.h. das zwischen zwei Aktivitäten ggf. mehrere Kontrollverbindungen bestehen.

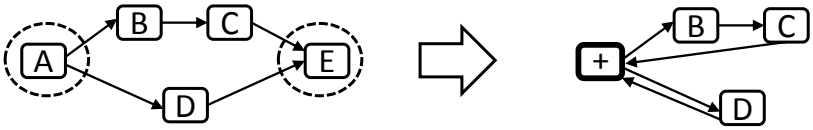


Abbildung 3.20: Skizze zu „Unverbundene Aktivitäten aggregieren (M3)“.

Beispiele: Die Aggregation aller Aktivitäten, die eine bestimmte Eigenschaft gemein haben, aber nicht notwendigerweise direkt verbunden sind, kann bspw. in der Prozessanalyse genutzt werden: Durch Aggregation der Aktivitäten nach der Organisationseinheit können Ausführungszeiten gegenübergestellt werden, sofern der Prozess zuvor mit diesen Laufzeitinformationen angereichert wurde (M15).

Muster 4: Eigenschaften abstrahieren

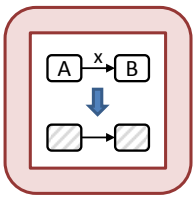


Abbildung 3.21: Symbol zu „Eigenschaften abstrahieren (M4)“.

Problem: Wie kann der Informationsgehalt von Prozesselementen reduziert werden?

Schlagwörter: Abstrahieren; Modifizieren; Ändern; Verbergen; Maskieren; Entfernen.

Beschreibung: Die Anwendung dieses Musters führt Modifikationen an den Eigenschaften von Aktivitäten, Kontrollverbindungen, Datenelementen und aggregierten Strukturen durch. Eine Modifikation kann den Wert einer Eigenschaft ändern, den Wert löschen, eine Eigenschaft gänzlich entfernen, Eigenschaften aggregieren oder hierarchisch gruppieren. Die Abstraktion von Werten kann außerdem zur Vorbereitung weiterer Transformationen verwendet werden.

Abbildung 3.22 skizziert die mehrfache Anwendung des Musters, um:

- die Bedingungen auf den Kontrollverbindungen *A*→*B* und *A*→*E* zu entfernen,
- die Struktur des Datenelementes *1* zu verbergen,
- bestimmte Details von Aktivität *C* zu entfernen und
- alle Details von Aktivität *E* zu entfernen.

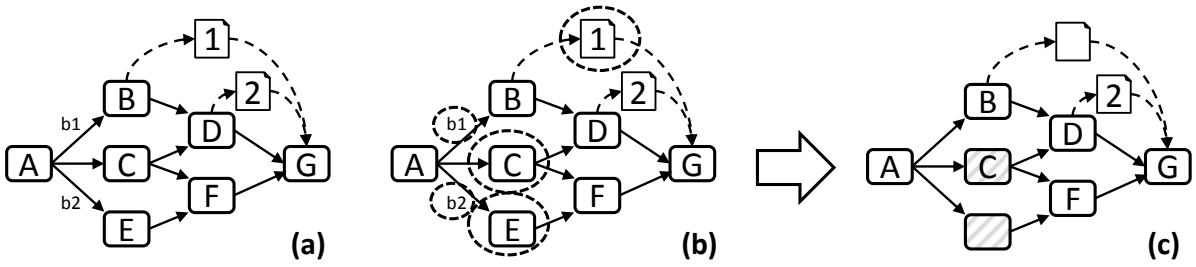


Abbildung 3.22: Skizze zu „Eigenschaften abstrahieren (M4)“ mit Eingabeprozess (a), Veranschaulichung der Anwendungsbereiche der mehrfachen Musteranwendung (b) und dem Ausgabeprozess (c).

Beispiele: Das Ändern des Wertes einer Eigenschaft kann für verschiedene Zwecke genutzt werden. In BPEL kann bspw. der Wert einer Eigenschaft durch den Platzhalter `##opaque` ersetzt werden, um den tatsächlichen Wert dieser Eigenschaft in einer Public View zu verbergen, um eine Zusammenarbeit mit anderen Organisationen zu ermöglichen ohne interne Details zu offenbaren. Zu diesem Zweck können Eigenschaften auch gänzlich entfernt werden.

Die Änderung eines Wertes kann auch dazu genutzt werden, den Abstraktionsgrad zu erhöhen, ohne die Informationen über den Wert gänzlich zu verlieren. Auf diese Weise können Informationen an die Anforderungen und Terminologie verschiedener Interessengruppen angepasst werden. Der Wert einer Eigenschaft *Ausführungsdauer* kann bspw. nach dem folgenden Schema abstrahiert werden:

- 0-30 Minuten: schnell
- 31-60 Minuten: normal
- > 60 Minuten: langsam

La Rosa et al. [RWM+11] beschreiben, dass durch eine Reduzierung des Prozessmetamodells eine Verringerung der Komplexität erreicht werden kann. Elemente eines Prozesses, die nicht im reduzierten Metamodell enthalten sind, werden aus dem Prozess entfernt oder abstrahiert. So können nach La Rosa et al. bspw. spezielle Ereignis- und Gateway-Typen zu allgemeineren Typen abstrahiert werden. Diese Form der Abstraktion kann ggf. durch eine entsprechende Modifikation der Typ-Eigenschaft von Elementen erfolgen.

Fill [Fil11] schlägt einen Ansatz zur Abstraktion der Semantik von Aktivitäten vor. In einer semantischen Annotation (M13) werden Aktivitäten einer Taxonomie zugeordnet, die verschiedene Tätigkeiten nach ihrem Abstraktionsgrad hierarchisch strukturiert. Eine Aktivität kann abstrahiert werden, indem sie der nächsthöheren Ebene der Taxonomie zugeordnet wird. Mithilfe dieser Technik können Prozesselemente mit semantisch ähnlichen, aber nicht wertgleichen Eigenschaften auf einer höheren Abstraktionsebene gemeinsam angesprochen werden. Wie in Abbildung 3.23 veranschaulicht, könnten bspw. Aktivitäten mit den zugeordneten Werten *Verschlüsselung* und *Signierung* aggregiert werden (M2), wenn sie zuvor im Abstraktionsgrad auf die gleiche Semantik *Sicherheit* angehoben wurden (M4).

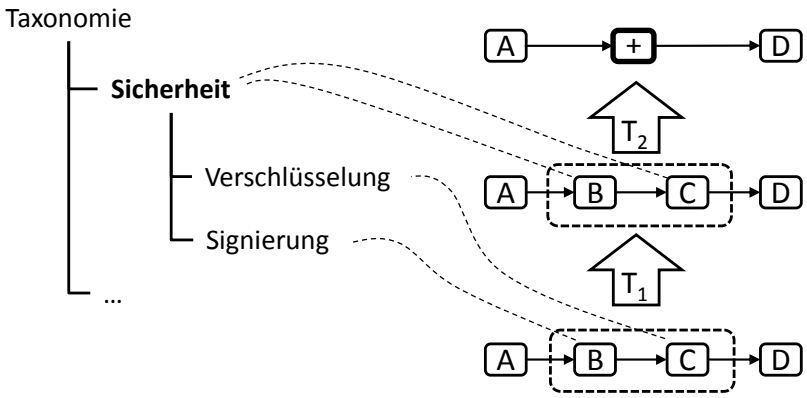


Abbildung 3.23: Feingranulare Abstraktion von Eigenschaften unter Einsatz einer Taxonomiezuordnung (T_1) mit anschließender Aggregation verbundener Aktivitäten mit wertgleichen Eigenschaften (T_2).

Muster 5: Datenelemente entfernen

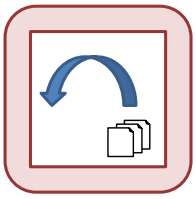


Abbildung 3.24: Symbol zu „Datenelemente entfernen (M5)“.

Problem: Wie kann die Anzahl an Datenelementen reduziert werden?

Schlagwörter: Ausblenden; Filtern; Auslassen; Löschen; Verbergen; Verstecken.

Beschreibung: Das Muster beschreibt das Entfernen von Datenelementen aus einem Prozessgraphen. Sofern die Vorgabe der Konsistenzbewahrung besteht (M29), wird das Lesen und Schreiben von entfernten Datenelementen von den entsprechenden Aktivitäten ebenfalls entfernt. Die Skizze in Abbildung 3.25 veranschaulicht die Entfernung von Datenelement *2* und dem damit verbundenen impliziten Datenfluss.

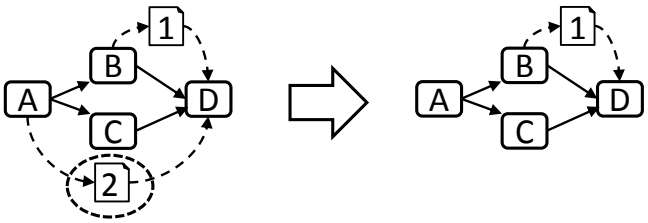


Abbildung 3.25: Skizze zu „Datenelemente entfernen (M5)“.

Beispiele: Das Entfernen von Datenelementen kann zum Schutz von Informationen eingesetzt werden, bspw. um bestimmte Nachrichtenformate oder Datentypen zu verbergen. Bobrik [Bob08] verwendet das Muster im Zuge der Entfernung von Aktivitäten, um nicht mehr verwendete Datenelemente ebenso aus dem Prozessgraphen zu entfernen. Das Werkzeug Stages [Met13] bietet eine Kompaktdarstellung von Prozessen an, in dem alle Datenelemente und der entsprechende Datenfluss ausgeblendet sind.

Muster 6: Datenelemente aggregieren

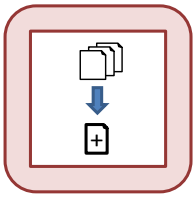


Abbildung 3.26: Symbol zu „Datenelemente aggregieren (M6)“.

Problem: Wie kann die Anzahl an Datenelementen reduziert werden, ohne Informationen gänzlich zu verlieren?

Schlagwörter: Zusammenfassen; Gruppieren; Clustern; Vereinigen; Verdichten.

Beschreibung: Durch die Aggregation von Datenelementen kann ein Prozess vereinfacht werden. Die Aggregation der Eigenschaften von Datenelementen gestaltet sich ähnlich wie bei der Aggregation von Aktivitäten (vgl. M2, M3). Die Bestimmung einer Bezeichnung für aggregierte Datenelemente erfolgt manuell (M13), während weitere Eigenschaften ggf. durch Summen- oder Listenbildung aggregiert werden können.

Die Skizze in Abbildung 3.27 zeigt zwei verschiedene Anwendungsmöglichkeiten dieses Musters. Die Datenelemente *1* und *2* stehen möglicherweise in kausalem Zusammenhang; die Datenelemente *3* und *4* werden zwischen denselben Aktivitäten ausgetauscht. Die Skizze illustriert den Ausgabeprozess, der bei der Aggregation der Datenelemente *1* und *2* sowie *3* und *4* entstehen würde.

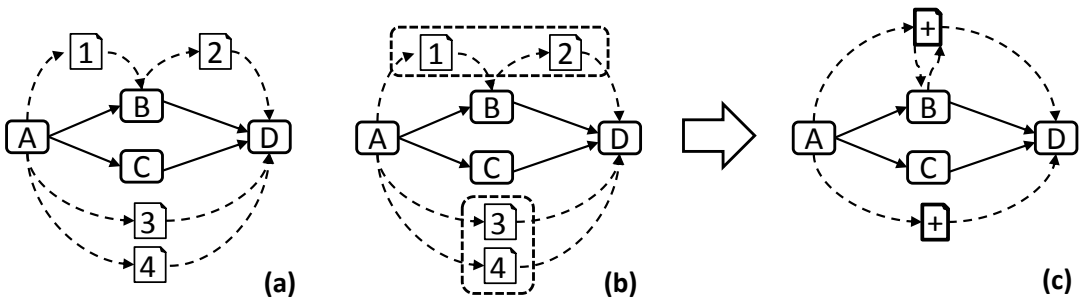


Abbildung 3.27: Skizze zu „Datenelemente aggregieren (M6)“ mit (a) einem Eingabeprozess, (b) zwei Anwendungsbereichen für eine mehrfache Musteranwendung und (c) dem entsprechenden Ausgabeprozess.

Beispiele: Fahrwinkel [Fah95] unterscheidet IT-Informationsobjekte und Papierinformationsobjekte als Typen von Datenelementen, wobei mehrere Datenelemente gleichen Typs zu einer sogenannten Informationsgruppe aggregiert werden können, um den Abstraktionsgrad eines Prozesses zu erhöhen. Wie Bobrik [Bob08] beschreibt, sind komplexe Transformationsfunktionen erforderlich, wenn die Semantik der Datenelemente bei der Aggregation berücksichtigt werden soll.

Selic [Sel11] beschreibt *Summary Message* als Architekturmuster, mit dem mehrere Nachrichten aggregiert werden.

Muster 7: Pfade aggregieren

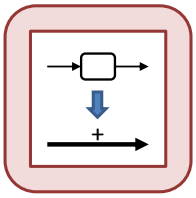


Abbildung 3.28: Symbol zu „Pfade aggregieren (M7)“.

Problem: Wie kann die Anzahl an Aktivitäten reduziert werden, ohne Informationen gänzlich zu verlieren?

Schlagwörter: Zusammenfassen; Vereinfachen.

Beschreibung: Verbundene Aktivitäten werden zu einem aggregierten Pfad zusammengefasst. Dadurch wird von den Tätigkeiten abstrahiert, die auf einem bestimmten Pfad in einem Prozess liegen. Während es bei der Aggregation von Aktivitäten (M2, M3) darum geht, die Information zu erhalten, *dass* etwas getan wird, tritt eben diese Information bei einer Anwendung dieses Musters in den Hintergrund. Die Information *wie* es getan wird, kann durch eine Aggregation von Eigenschaften der Aktivitäten auf dem Pfad in abstrakter Form erhalten werden. Eben diese Pfadinformation macht den Unterschied zum Entfernen von Aktivitäten aus (M1), bei dem diese Information verloren geht. Werden entlang des Pfades Datenelemente verwendet, die keine Bezüge zu Aktivitäten außerhalb des Pfades haben, können diese ebenfalls in den aggregierten Pfad eingeschlossen werden. Bestehen auch außerhalb des Pfades Lese- oder Schreibzugriffe auf Datenelemente, die innerhalb des Pfades verwendet werden, liegt eine Mehrdeutigkeit vor. Diese Mehrdeutigkeit kann bspw. aufgelöst werden, indem diese Datenelemente unverändert im Prozess verbleiben oder entfernt werden.

Die Skizze in Abbildung 3.29 zeigt die Aggregation des Pfades $A \rightarrow .. \rightarrow G$. Die Transformation T_1 aggregiert die im Anwendungsbereich enthaltenen Elemente, ohne eine Veränderung der Position der Aktivitäten *A* und *G* vorzunehmen. Transformation T_2 berücksichtigt die häufig implizite Anforderung, zusätzlich zu einer Strukturveränderung eine Aktualisierung des Layouts und damit eine Neupositionierung der Aktivitäten auf der Zeichenfläche vorzunehmen, wie in Abbildung 3.29 im Graphschema (M19) skizziert.

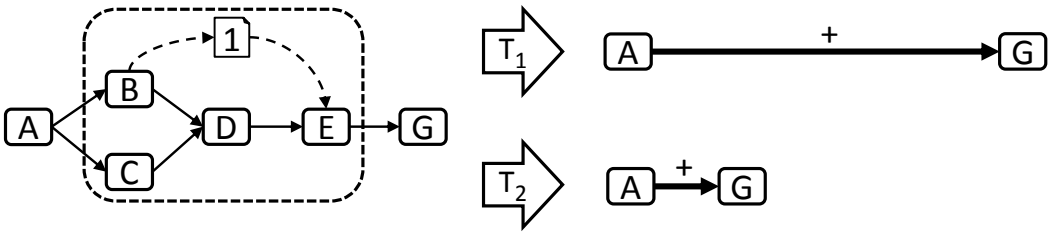


Abbildung 3.29: Skizze zu „Pfade aggregieren (M7)“.

Beispiele: In automatisierten Geschäftsprozessen dienen manche Prozessstrukturen lediglich der Übertragung von Informationen zwischen zwei Systemen, bspw. werden Daten zunächst konvertiert, in eine Nachricht geschrieben und diese wird an ein System verschickt. Eine

derartige Sequenz kann zu einem aggregierten Pfad zusammengefasst werden. Ein solcher Pfad abstrahiert auf diese Weise einen Kommunikationsweg.

Selic [Sel11] beschreibt diese Anwendung im Zusammenhang mit Softwarearchitekturen als Black Line, welche einen komplexen Kommunikationskanal repräsentiert, der durch integrierte Komponenten realisiert ist.

Muster 8: Strukturen aufräumen

Abbildung 3.30: Symbol zu „Strukturen aufräumen (M8)“.

Problem: Wie kann die Komplexität eines Prozess reduziert werden, ohne seinen Informationsgehalt zu reduzieren?

Schlagwörter: Aufräumen; Nachbearbeiten; Überreste entfernen; Überflüssiges entfernen.

Beschreibung: Das Aufräumen (engl. *cleaning*) dient dazu, Elemente zu entfernen, die durch die Anwendung von strukturverändernden Transformationen überflüssig geworden sind. Sie dienen in der Sicht keinem Zweck mehr, beeinträchtigen aber die Konsistenz nicht notwendigerweise (M29). Abhängig davon, wie das Entfernen von Aktivitäten (M1) umgesetzt wird, können Überreste wie ungenutzte Datenelemente im Prozess zurückbleiben. Es können auch durch vorangehende Transformationen Sequenzen von vollständig abstrahierten Aktivitäten entstehen. Das Aufräumen entfernt diese Datenelemente (M5) und überflüssige abstrahierte Aktivitäten (M1) oder aggregiert diese Aktivitäten (M2). Dieses Muster ist nicht überdeckungsfrei und könnte auch als Musterkomposition aufgefasst werden.

Die Skizze in Abbildung 3.31 zeigt einen Eingabeprozess bei dem bereits mehrere Aktivitäten entfernt wurden. Die Datenelemente *1* und *2* sind Überreste aus vorangehenden Transformationen. Der Anwendungsbereich des Aufräumens kann ein bestimmter Teilbereich des Prozesses sein oder sich über den gesamten Prozess erstrecken, vgl. die Skizze (b). Nach der Anwendung des Musters (c) sind die ungenutzten Elemente entfernt.

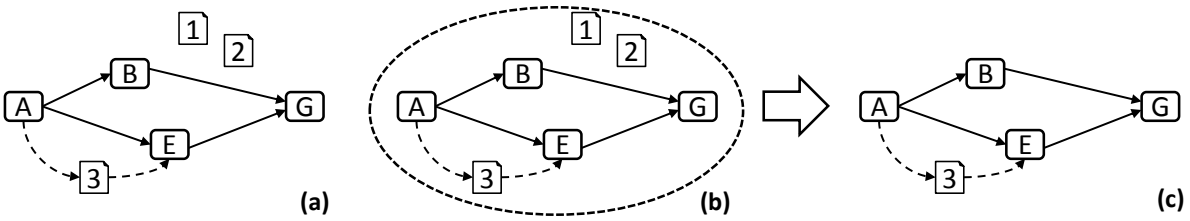


Abbildung 3.31: Skizze zu „Strukturen aufräumen (M8)“.

Beispiele: Kolb et al. [KKR12] setzen das Aufräumen durch vordefinierte Refactoring-Operationen um, in denen unnötige Kontrollverbindungen und Gateways entfernt werden. Diese Operationen werden im Anschluss an Strukturveränderungen aufgerufen. Nach Gruhn et al. [GL09] können überflüssige Gateways aus einem Modell entfernt werden, um das Modell zu vereinfachen ohne dessen Semantik zu verändern.

In einem BPEL-Prozess können bspw. `<variables>`, `<partnerLinks>`, `<extensions>`, `<messagesExchanges>` und `<correlationSets>` nach dem Entfernen von Aktivitäten als Überreste in einem Prozess enthalten sein [SLS10b]. Auch können nicht mehr benötigte Kontrollelemente in einem Prozess nach dem Entfernen von Aktivitäten verbleiben, wie bspw. eine `<sequence>` ohne darin enthaltene Aktivitäten. Durch ein Aufräumen können diese nicht mehr verwendeten Elemente entfernt werden.

3.4.1.2 Erweiterung

Strukturverändernde, erweiternde Muster dienen der Konkretisierung oder der Erweiterung eines Eingabeprozesses. Durch ihre Anwendung bildet ein Prozessmodell die Realität vollständiger, konkreter oder detaillierter ab. Die erweiternden Muster stellen ausdrücklich keine Inverse der reduzierenden Muster dar. Ein Wiedereinblenden ausgeblendeter Strukturen und ein Wiederaufklappen aggregierter Strukturen kann stattdessen durch eine erneute Transformation des Ursprungsprozesses mit abgeschwächten Filter- oder Aggregationskriterien erreicht werden. Das Wiedereinblenden und das Wiederaufklappen sind daher in diesem Katalog nicht als Muster aufgeführt.

Muster 9: Aktivitäten einfügen

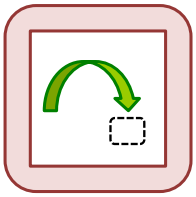


Abbildung 3.32: Symbol zu „Aktivitäten einfügen (M9)“.

Problem: Wie kann ein Prozess um Informationen zu Abläufen erweitert werden, die außerhalb des Prozess stattfinden, aber mit dem Prozess in Verbindung stehen?

Schlagwörter: Einfügen; Hinzufügen; Einweben; Integrieren.

Beschreibung: Das Einfügen von Aktivitäten erweitert einen Eingabeprozess um Aktivitäten, die im Eingabeprozess nicht enthalten sind. Ein Prozessgraph kann dadurch um Aktivitäten erweitert werden, die sich außerhalb des Modells des Prozesses befinden oder ereignen, diesem aber zugeordnet werden können.

Abbildung 3.33 skizziert einen Eingabeprozess, bei dem an zwei Anwendungsbereichen des Musters eine Aktivität eingefügt werden soll: Aktivität *X* nach der Aktivität *A* und Aktivität *Y* nach Aktivität *C*.

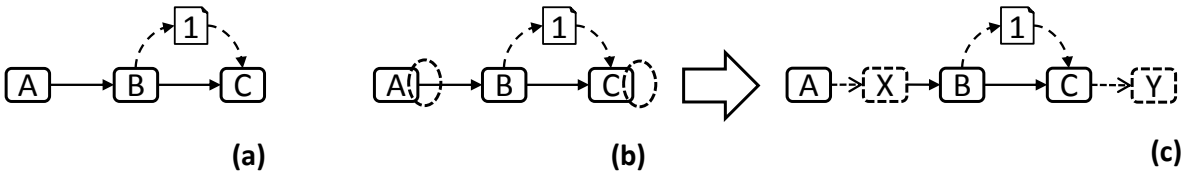


Abbildung 3.33: Skizze zu „Aktivitäten einfügen (M9)“.

Beispiele: Durch die Übertragung von Konzepten der Aspektorientierten Programmierung (AOP) ist ein Einweben zusätzlicher Funktionalität in automatisierte Geschäftsprozesse möglich. Durch Aspect-Weaving-Techniken wie BPEL’n’Aspects [KL09] kann ein Prozess auch während der Laufzeit um zusätzliche Aktivitäten erweitert werden, die im ursprünglichen Prozessmodell nicht enthalten sind. Im Monitoring stellt sich dann jedoch die Frage, welche zusätzliche Funktionalität an welcher Stelle ausgeführt wurde. Das Einfügen von Aktivitäten an den betreffenden Stellen ist daher nur als temporäre Strukturveränderung sinnvoll.

Zur Unterstützung der Prozessanalyse können fiktive Aktivitäten eingefügt werden, um Liegezeiten zwischen Aktivitäten zu repräsentieren. Neben der eher geringen Verbreitung dieses Musters im Bereich der Sichten auf Prozesse wird es verstärkt in der Handhabung und zum Vergleich von Varianten eingesetzt, vgl. z.B. Weber und Reichert [WR08].

Muster 10: Aktivitäten zerlegen

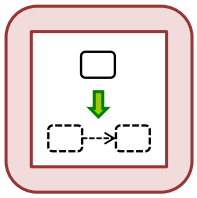


Abbildung 3.34: Symbol zu „Aktivitäten zerlegen (M10)“.

Problem: Wie kann die Komplexität von Aktivitäten reduziert werden, ohne den Informationsgehalt zu reduzieren?

Schlagwörter: Zerlegen; Teilen; Disaggregieren; Aufspalten.

Beschreibung: Das Zerlegen von Aktivitäten bewirkt eine Konkretisierung des Eingabeprozesses. Das Zerlegen kann auf Basis von Informationen erfolgen, die bereits im Prozess enthalten sind oder die von Datenquellen außerhalb des Prozesses bezogen werden (M13, M14, M15). Die Skizze in Abbildung 3.35 veranschaulicht die Zerlegung der Aktivität *B*.

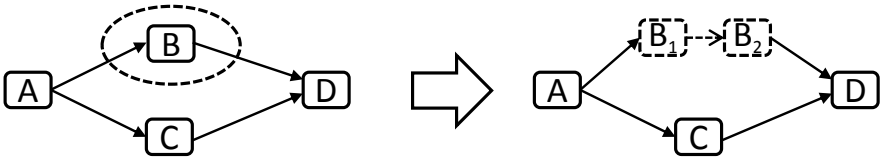


Abbildung 3.35: Skizze zu „Aktivitäten zerlegen (M10)“.

Beispiele: Das Zerlegen von Aktivitäten kann angewendet werden, um die atomare BPEL-Aktivität `<assign>` in die darin enthaltenen `<copy>` Statements zu zerlegen und diese Statements als Aktivitätssequenz darzustellen [SKL+09]. Analog könnte eine atomare Aktivität zum Aufruf eines Dienstes bzw. eines Sub-Prozesses zerlegt werden, sofern prozessuale Informationen über die Funktionsweise des Dienstes verfügbar sind und eingebunden werden können (M15).

Muster 11: Datenelemente zerlegen

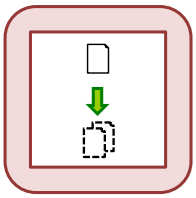


Abbildung 3.36: Symbol zu „Datenelemente zerlegen (M11)“.

Problem: Wie kann die Komplexität von Datenelementen reduziert werden, ohne den Informationsgehalt zu reduzieren?

Schlagwörter: Zerlegen; Teilen; Disaggregieren; Aufspalten.

Beschreibung: Das Zerlegen von Datenelementen ist ein Muster zur Konkretisierung und Verfeinerung eines Prozesses. Wie in Abbildung 3.37 skizziert, kann das atomare Datenelement *1* in zwei Bestandteile zerlegt werden, *1.1* und *1.2*. Für ein Einfügen von nicht im Eingabeprozess enthaltenen Datenelementen wurde kein Anwendungsfall identifiziert und daher kein entsprechendes Muster definiert.

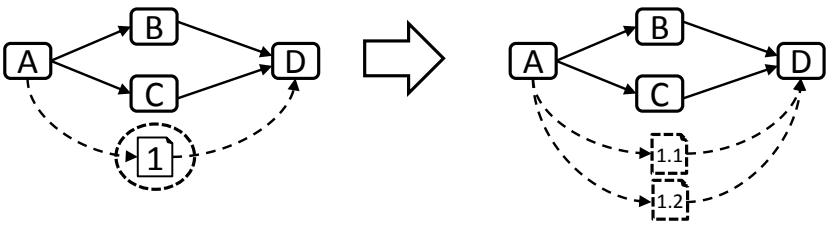


Abbildung 3.37: Skizze zu „Datenelemente zerlegen (M11)“.

Beispiele: Die Datentypen und Nachrichtenformate, die in automatisierten Geschäftsprozessen eingesetzt werden, bestehen mitunter aus mehreren Bestandteilen, in die sie für eine detailliertere Sicht zerlegt werden könnten. XML dient in vielen Prozesssprachen zur Beschreibung von Formaten. Die hierarchische Struktur von XML könnte bei einer Zerlegung von Datenelementen genutzt werden.

Muster 12: Pfade zerlegen

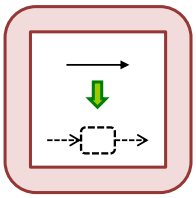


Abbildung 3.38: Symbol zu „Pfade zerlegen (M12)“.

Problem: Wie können zwischen Aktivitäten bestehende Verbindungen konkretisiert werden?

Schlagwörter: Zerlegen; Teilen; Disaggregieren; Aufspalten.

Beschreibung: Sofern Informationen aus anderen Datenquellen vorhanden sind (M13, M14, M15), können diese zur Konkretisierung einer ursprünglich atomaren Kontrollverbindung genutzt werden. Die Skizze in Abbildung 3.39 zeigt die Zerlegung einer atomaren Kontrollverbindung in den Pfad *A→X→Y→B*. Die ursprüngliche atomare Kontrollverbindung kann eine Übergangsbedingung tragen und sollte daher beim Zerlegen erhalten werden. Es besteht allerdings eine aufzulösende Mehrdeutigkeit darin, an welcher Position sich die ursprüngliche Kontrollverbindung im zerlegten Pfad befinden sollte – am Anfang, innerhalb des zerlegten Pfades oder am Ende. Das Zerlegen von Pfaden kann als Gegensatz zur Aggregation von Pfaden (M7) betrachtet werden.

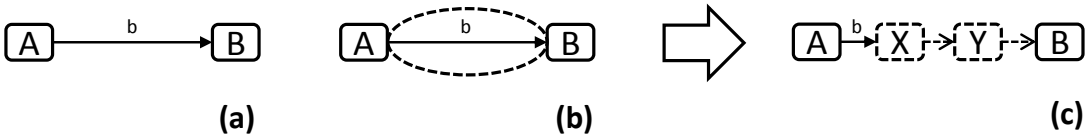


Abbildung 3.39: Skizze zu „Pfade zerlegen (M12)“.

Beispiele: In einem lediglich modellierten, aber nicht ausführbaren Geschäftsprozess (vgl. Abschnitt 2.1.1) besteht i.d.R. keine Garantie, dass die Verbindung zwischen zwei Aktivitäten vollständig spezifiziert ist. Tatsächlich können zahlreiche manuelle Schritte in einer eigentlich direkten Kontrollverbindung verborgen sein. In der Beschreibung und Modellierung von Prozessen wird bewusst oder unbewusst bereits ein gewisser Grad an Abstraktion angesetzt [JFE+11]. Das Zerlegen von Pfaden schafft an dieser Stelle eine Brücke zwischen eher abstrakten Modellen und deren Konkretisierung.

3.4.2 Muster zur Informationsanreicherung

Die Informationen, die ein Prozessmodell bereits enthält, decken nur einen Teil des Bedarfs an Informationen ab, der entlang des Lebenszyklus von Prozessen besteht. Der Zweck einer Informationsanreicherung besteht darin, weitere Informationen mit einem Prozessmodell in Verbindung zu bringen und diese Informationen zur Abstraktion, Konkretisierung und Darstellung zu verwenden.

Muster 13: Nutzergestützt anreichern

Abbildung 3.40: Symbol zu „Nutzergestützt anreichern (M13)“.

Problem: Wie kann ein Nutzer eine Prozesssicht beeinflussen?

Schlagwörter: Annotieren; Markieren; Kennzeichnen; Einschätzen; Bewerten; Klassifizieren.

Beschreibung: Die nutzergestützte Informationsanreicherung bringt Prozesskenntnisse und Erfahrung verschiedener Interessengruppen mit einem Prozess in Verbindung. Diese Anreicherung kann den gesamten Prozess betreffen oder auch nur auf einen bestimmten Bereich angewendet werden. Die angereicherten Informationen können zur Steuerung nachfolgender Transformationen eingesetzt werden. Eine zentrale Anwendung des Musters besteht in der individuellen Kennzeichnung von Strukturen, die für eine bestimmte Interessengruppe relevant sind. Ferner stellt die semantische Annotation von Prozesselementen mit maschinenlesbaren Begriffen und Eigenschaften eine wichtige Ausprägung dieses Musters dar. Es ist erforderlich, diese Informationen bei Änderungen am Prozessmodell entsprechend mitzupflegen und damit aktuell zu halten.

Abbildung 3.41 skizziert die durch einen Nutzer durchgeführte Kennzeichnung der subjektiven Relevanz bestimmter Strukturen. Neben einer binären Relevanzmarkierung wie bspw. mit oder ohne Kennzeichnung als Favorit für wichtige oder unwichtige Elemente bestehen zahlreiche weitere Möglichkeiten, z.B. die Nutzung einer Skala. Durch die einmalige Anwendung dieses Musters und eine Persistierung der angereicherten Informationen können diese in anderen Musteranwendungen genutzt werden, etwa für eine Hervorhebung durch Anpassung des Erscheinungsbildes (M16, M17, M18). Eine Persistierung annotierter Informationen erfolgt im Idealfall getrennt vom Prozessmodell, um dieses nicht mit Informationen zu belasten.

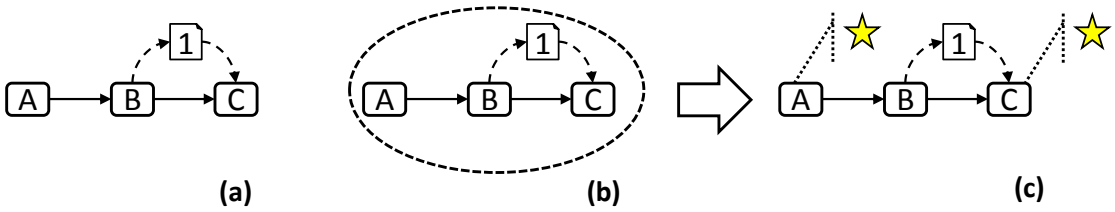


Abbildung 3.41: Skizze zu „Nutzergestützt anreichern (M13)“ für einen Eingabeprozess (a) mit Anwendung auf den gesamten Prozess (b) zur Markierung von Favoriten eines bestimmten Nutzers (c).

Beispiele: Die nutzergestützte Anreicherung kann zur Klassifikation von Geheimhaltungsstufen genutzt werden (z.B. öffentlich, intern, vertraulich). Die Erzeugung einer Public View auf einen Prozess kann diese Information zur Ansteuerung nutzen um Aktivitäten, Eigen-

schaften und Datenelemente zu entfernen (M1, M4, M5), die einem Geschäftspartner nicht gezeigt werden sollen [SLS10a].

Wie in der Patentanmeldung [TRK+08] beschrieben, kann mittels Statistik und Sensoren wie einem Pulsmesser, einem Mikrophon und einer Kamera die emotionale Verfassung eines Nutzers ermittelt werden. Eine Abbildung zwischen dem Blick des Nutzers und dargestellten Prozessstrukturen ermöglicht es, diese Strukturen automatisch mit den ermittelten emotionalen Informationen anzureichern, bspw. im Hinblick auf die empfundene Relevanz.

Streule [Str09] setzt die nutzergestützte Anreicherung in einem Modellierungswerkzeug ein, um Aktivitäten zu kennzeichnen, die von Transformationen ausgenommen werden sollen (Strukturen bewahren, M31). Ähnlich dazu kann im Modellierungswerkzeug Signavio Process Editor [Sig13] zu jedem Element durch eine Auswahl eine Kennzeichnung angegeben werden, in welcher interessengruppenspezifischen Sicht das Element darzustellen oder auszublenden ist (M1, M5). Für eine präzise und eindeutige semantische Annotation ist allerdings eine geeignete Werkzeugunterstützung mit Ontologien empfehlenswert [BDW07].

Muster 14: Statisch anreichern

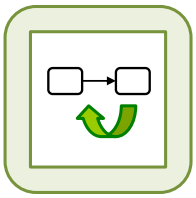


Abbildung 3.42: Symbol zu „Statisch anreichern (M14)“.

Problem: Wie kann der Informationsgehalt eines Prozesses erhöht werden?

Schlagwörter: Prüfen; Erkennen; Analysieren; Ableiten; Bestimmen.

Beschreibung: Die statische Anreicherung repräsentiert Transformationen, mit denen Informationen mit dem Prozess in Verbindung gebracht werden, die über eine statische Analyse des Prozessmodells gewonnen werden können. Die Skizze in Abbildung 3.43 zeigt einen Eingabeprozess (a), der in Bezug auf die Verwendung personenbezogener Daten analysiert und mit den Analyseergebnissen angereichert wurde (b). Diese Analyse kann etwa auf einer Suche nach Schlüsselbegriffen wie „Name“ in den Datenstrukturen basieren.

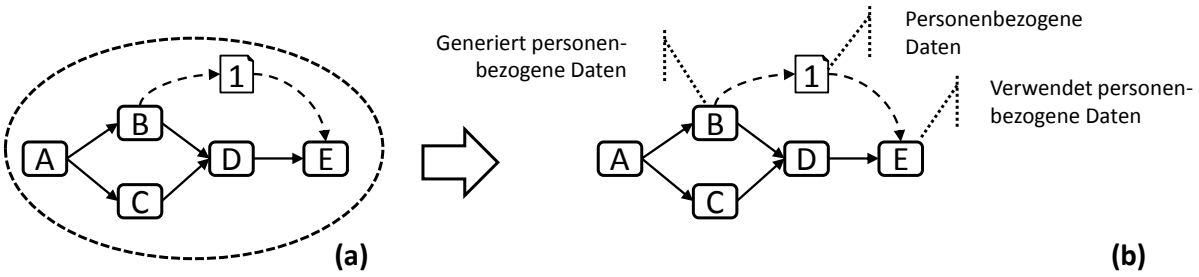


Abbildung 3.43: Skizze zu „Statisch anreichern (M14)“.

Beispiele: In der Forschung zur Prozessmodellabstraktion wurden zahlreiche Techniken diskutiert, um die Relevanz von Aktivitäten durch statische Analyse zu bestimmen. Viele Techniken bestimmen die Relevanz von Aktivitäten anhand bedeutsamer Eigenschaften wie bspw. anhand der Bezeichnung. Eine Aktivitätsbezeichnung kann Verben und Substantive enthalten, die auf die Semantik der Aktivität schließen lassen. Diese Informationen können maschinenlesbar annotiert werden und stehen anschließend für weitere Transformationen bereit. Die Relevanz von Datenelementen kann ebenfalls nach deren Bezeichnung oder anhand anderer Kriterien wie dem Format oder der Zugriffshäufigkeit bestimmt werden. Die annotierte Relevanz kann u.a. zur Abstraktion verwendet werden (M1, M2, M5).

Ferner können Techniken zur Strukturerkennung eingesetzt werden, um bestimmte Prozessfragmente in einem Prozessmodell zu identifizieren [Ma12]. Mit einer entsprechenden Annotation kann diese Information in einem angepassten Erscheinungsbild verwendet werden (M16, M17, M18); etwa zur Hervorhebung einer erkannten Struktur [Pos10b].

Weitere Anwendungsmöglichkeiten bestehen in der Erkennung von Deadlocks sowie in der Bereitstellung von Informationen über festgestellte Verletzungen der Regelkonformität zur Modellierungszeit (engl. *design time Compliance violation*).

Muster 15: Dynamisch anreichern

Abbildung 3.44: Symbol zu „Dynamisch anreichern (M15)“.

Problem: Wie kann der Informationsgehalt eines Prozesses erhöht werden, wenn Laufzeitinformationen zur Verfügung stehen?

Schlagwörter: Laufzeitinformationen hinzufügen; Informationen integrieren.

Beschreibung: Dynamische Anreicherung umfasst Informationen über laufende oder bereits abgeschlossene Ausführungen des Prozesses. Informationen über das dynamische Verhalten eines Prozesses, wie bspw. Aktivitätszustände, Werte von Datenelementen oder der Auswertungszustand von Kontrollverbindungen geben Aufschluss über den aktuellen oder einen vergangenen Zustand einer Prozessinstanz. Durch die Anreicherung dynamischer Informationen kann eine Grundlage für zahlreiche Fragestellungen gelegt werden, die prozessbezogene Informationen im Prozesskontext erfordern.

Die Skizze in Abbildung 3.45 zeigt die Anwendung des Musters, um Informationen über eine bestimmte Prozessinstanz zu einem Prozessgraphen hinzuzufügen. Die Skizzenvariante (b) behält die Aktivitätsbezeichnungen bei und stellt die Aktivitätszustände durch eine Annotation dar; Skizzenvariante (c) zeigt den PI-Graphen in einer vereinfachten Form.

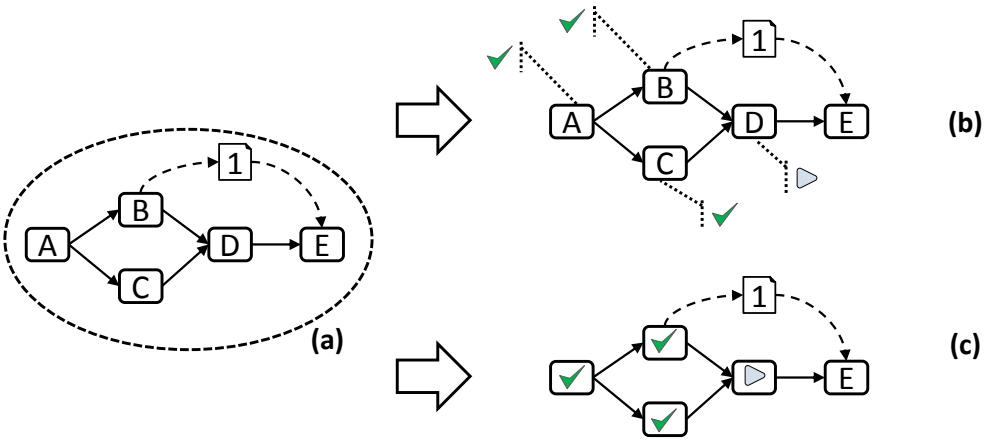


Abbildung 3.45: Skizze zu „Dynamisch anreichern (M15)“ mit einer Anreicherung von Aktivitätszuständen und Werten von Datenelementen.

Beispiele: Neben der Anwendung dieses Muster im Monitoring des Ist-Zustands einer laufenden Prozessinstanz können statistische Laufzeitinformationen, bspw. zur durchschnittlichen Ausführungsdauer von Aktivitäten, für ein vorausschauendes Monitoring oder zur Prozessanalyse genutzt werden. Neben der Nutzung zur Abstraktion können diese angereicherten Informationen auch im Erscheinungsbild von Aktivitäten dargestellt werden (M16).

Die dynamischen Aspekte eines Prozesses beinhalten auch organisatorische Informationen wie den Standort einer Person, die zur Bearbeitung einer bestimmten Aktivität ausgewählt wurde. Aus dem Standort geht hervor, in welchem Land, Kontinent und in welcher Zeitzone sich diese Person befindet. Zur Trennung von Zuständigkeiten enthält das Prozessmodell derartige Informationen bewusst nicht. Für Sichten zur Analyse des Laufzeitverhaltens eines Prozesses sind diese Informationen aber durchaus relevant.

3.4.3 Muster zur Darstellungsanpassung

Darstellungsmuster beschreiben die Art und Weise, wie die Elemente in einem Prozessgraphen dargestellt werden sollen. Darstellungsmuster sind in drei Unterkategorien gegliedert:

- Das *Erscheinungsbild* legt die Darstellung einzelner Aktivitäten, Verbindungen und Datenelemente fest;
- das *Schema* legt die maßgebliche Struktur der Darstellung für den gesamten Prozessgraphen fest;
- das *Layout* spezifiziert Anforderungen an die Darstellung im gewählten Schema.

3.4.3.1 Erscheinungsbild

Muster zur Festlegung und Modifikation des Erscheinungsbildes werden dazu verwendet, um für eine bestimmte Interessengruppe und Aufgabenstellung relevante Informationen auf einen Blick darzustellen. Informationen werden im Erscheinungsbild vornehmlich durch Schrift, geometrische Formen und Symbole mit verschiedenen Farben und Größen codiert.

Muster 16: Erscheinungsbild von Aktivitäten anpassen

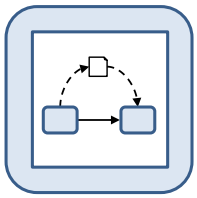


Abbildung 3.46: Symbol zu „Erscheinungsbild von Aktivitäten anpassen (M16)“.

Problem: Wie können die zu Aktivitäten verfügbaren Informationen dargestellt werden?

Schlagwörter: Darstellen; Zeichnen; Codieren; Anzeigen; Konnotieren; Einfärben.

Beschreibung: Das Erscheinungsbild von Aktivitäten ähnelt i.d.R. dem in Abbildung 3.47 gezeigten Stand der Technik. Aktivitäten werden demnach als Rechteck gezeichnet, ggf. hat das Rechteck abgerundeten Ecken. Dieses Rechteck bildet die Zeichenfläche der Aktivität. Der Aktivitätstyp kann mit einem Symbol veranschaulicht werden, wobei dessen Position auf der Zeichenfläche abhängig von der Prozesssprache und dem Werkzeug variiert. Weitere Symbole wie etwa Hinweissymbole werden weniger prominent auf der Zeichenfläche platziert oder können u.U. auch außerhalb der Aktivitätsform dargestellt werden. Die Beschriftung stellt eine für die Aktivität maßgebliche Information dar. Der Aktivitätsname oder eine Kurzbeschreibung wird i.d.R. als Beschriftung verwendet. Weitere Möglichkeiten der Gestaltung bestehen in der Formatierung der Beschriftung, der Anpassung der Aktivitätsgröße sowie in der stilistischen Konfiguration vom Hintergrund und Rahmen.

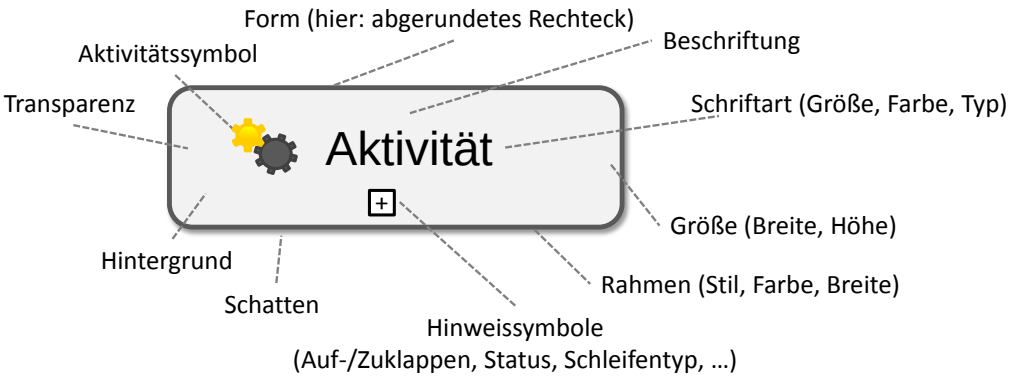


Abbildung 3.47: Stand der Technik in der Darstellung von Aktivitäten, vgl. [SLS10b].

Vorlagen (engl. *templates*) können in unterschiedlichen Detailgraden vordefiniert und für unterschiedliche Aktivitäten und Aktivitätstypen eingesetzt werden [Bob08]. Demzufolge kann für eine aggregierte Struktur (M2, M3) eine normale Vorlage verwendet werden, mit der sie nicht oder ggf. nur durch ein Symbol oder die Beschriftung von gewöhnlichen Aktivitäten unterscheidbar wäre. Andererseits kann die Verwendung einer besonderen Vorlage unmittelbar Aufschluss über die aggregierten Inhalte geben. Abbildung 3.48 skizziert die Verwendung ein- und derselben Vorlage für alle Aktivitäten, wobei das Aktivitätssymbol dem jeweiligen Aktivitätstyp entspricht.

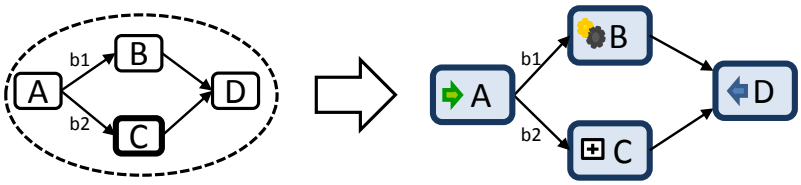


Abbildung 3.48: Skizze zu „Erscheinungsbild von Aktivitäten anpassen (M16)“.

Beispiele: Müller und Rogge-Solti [MR11] setzen die Hintergrundfarbe von Aktivitäten ein, um den Aktivitäten zugeordnete Rollen zu codieren, ohne den Prozess nach dieser Organisationsinformation in Swimlanes auslegen zu müssen (M26). Mehrfachzuordnungen bei komplexen Aktivitäten konnten so ebenfalls dargestellt werden. Dyer et al. [DHL+12] verwenden acht festgelegte Farbcodes, um zwischen verschiedenen Automatisierungsgraden und der Semantik von Aktivitäten zu unterscheiden. Bspw. repräsentieren Aktivitäten in hellem blau Entscheidungen oder Regeln. Sonntag et al. [SGK+11] nutzen die Hintergrundfarbe, um den angereicherten Aktivitätszustand (M15) darzustellen; eine grüne Einfärbung steht dafür, dass eine Aktivität erfolgreich abgeschlossen wurde. Wie auch bei Landkarten, bei denen eine große Anzahl unterschiedlicher Symbole verwendet wird, ist ggf. eine Legende zur Erläuterung der Informationscodierung bei der Prozessdarstellung erforderlich.

Mendling et al. [MRR10] stellen 25 Symbole vor, um die Semantik bestimmter Aktivitäten darzustellen. Jedes Symbol repräsentiert eine von ihnen als gängig festgestellte Klasse von Verben. Wird eines der Verben in einer Aktivitätsbezeichnung verwendet, soll das entsprechende Symbol als Aktivitätssymbol verwendet werden. Rupprecht et al. [RRF+01] haben einen vergleichbaren Ansatz für die Domäne der Fahrzeugentwicklung verfolgt, es werden 10 Symbole zur Darstellung gängiger Tätigkeitsmuster dieser Domäne vorgeschlagen.

Muster 17: Erscheinungsbild von Verbindungen anpassen

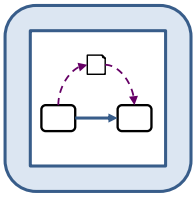


Abbildung 3.49: Symbol zu „Erscheinungsbild von Verbindungen anpassen (M17)“.

Problem: Wie können die zu Verbindungen verfügbaren Informationen dargestellt werden?

Schlagwörter: Darstellen; Zeichnen; Codieren; Anzeigen; Konnotieren; Einfärben.

Beschreibung: Das Erscheinungsbild von Verbindungen kann hinsichtlich verschiedener Merkmale modifiziert werden. Abbildung 3.50 illustriert den Stand der Technik in der Darstellung von Verbindungen. Demnach können für den Pfeilanfang und die Pfeilspitze verschiedene Formen verwendet werden. Eine Verbindung kann mit einer Beschriftung versehen sein. In Abhängigkeit von den Möglichkeiten des Layouts kann sie direkt, geschwungen oder gewinkelt verlaufen. Rahmen, Linienstil, Hinweissymbole und der Hintergrund können zusätzlich zur Informationscodierung herangezogen werden.

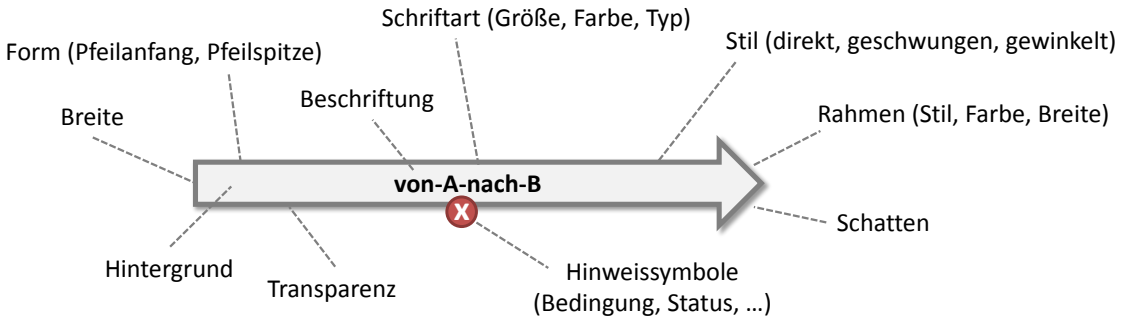


Abbildung 3.50: Stand der Technik in der Darstellung von Verbindungen.

Die Skizze in Abbildung 3.51 zeigt die Darstellung von Kontrollverbindungen in blau, ohne Form am Pfeilanfang und mit geschlossener Pfeilspitze. Datenverbindungen werden in grün mit punktförmigem Pfeilanfang und offener Pfeilspitze dargestellt.

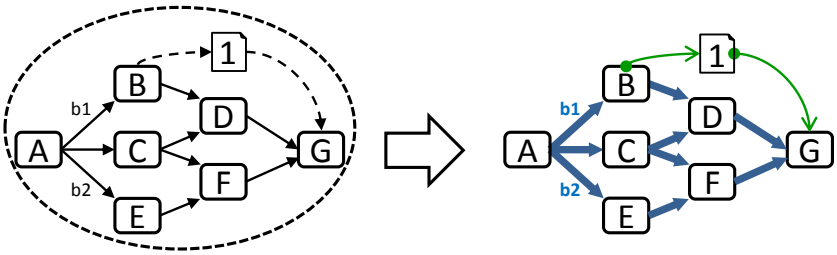


Abbildung 3.51: Skizze zu „Erscheinungsbild von Verbindungen anpassen (M17)“.

Beispiele: Man kann im Erscheinungsbild von Kontrollverbindungen den Auswertungszustand von ihren Übergangsbedingungen darstellen (*nicht ausgewertet*, *wahr*, *falsch*). Dies setzt allerdings die Anreicherung mit Laufzeitinformationen voraus (M15). Sofern mehrere Linienstile und Farben eingesetzt werden, kann ggf. eine Legende zur Erläuterung nötig sein.

Sonntag et al. [SGK+11] haben vorgeschlagen, einen Prozess mit Laufzeitinformationen über das Volumen übertragener Daten anzureichern (M15) und diese Information in Form angepasster Stärke von Datenverbindungen darzustellen.

Wie Awad und Weske [AW10] zeigen, kann eine statische Anreicherung (M14) mit Informationen über die Regelkonformität eines Prozesses verwendet werden, um für Compliance problematische Ausführungspfade farblich zu kennzeichnen.

Muster 18: Erscheinungsbild von Datenelementen anpassen

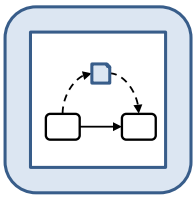


Abbildung 3.52: Symbol zu „Erscheinungsbild von Datenelementen anpassen (M18)“.

Problem: Wie können die zu Datenelementen verfügbaren Informationen dargestellt werden?

Schlagwörter: Darstellen; Zeichnen; Codieren; Anzeigen; Konnotieren; Einfärben.

Beschreibung: Das herkömmliche Erscheinungsbild von Datenelementen in Prozessmodellen besteht aus einem Symbol, das über Datenverbindungen mit Aktivitäten verbunden ist. Eine Beschriftung des Datenelementes kann unter, über, in oder neben dem Symbol erfolgen. Der PM-Graph in Abbildung 3.53 (a) skizziert das herkömmliche Erscheinungsbild. Wie in Abbildung 3.53 (b) skizziert, können je nach Datenelementtyp (z.B. doc, pdf) oder Inhalt (z.B. Bericht, Angebot) unterschiedliche Symbole verwendet werden. Für Symbole mit einer komplexen Informationszusammensetzung empfiehlt sich die Verwendung eines Vorlagenmechanismus (vgl. M16).

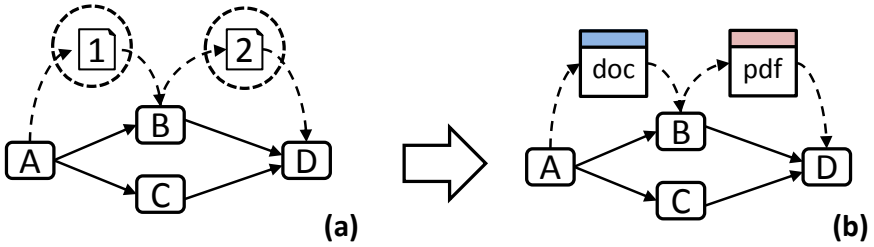


Abbildung 3.53: Skizze zu „Erscheinungsbild von Datenelementen anpassen (M18)“.

Beispiele: Eine Farbcodierung von Datenelementen kann zur Darstellung einer Eigenschaft verwendet werden, die im Prozessmodell bereits vorhanden ist, die zuvor durch einen Nutzer annotiert wurde (M13) oder die durch statische Analyse bestimmt werden kann (M14). Dokumente für die eine gesetzliche Aufbewahrungspflicht gilt und die zuvor entsprechend gekennzeichnet wurden können bspw. rot umrandet werden.

Das Erscheinungsbild von Datenelementen kann auch auf einer Anreicherung mit Laufzeitinformationen aufbauen (M15). Datenelemente können dann z.B. als Tabelle oder Liste von Schlüssel-Wert-Paaren gestaltet werden. Da die Datenelemente in einem Geschäftsprozess u.U. eine umfangreiche Datenmenge enthalten können, ist eine detaillierte Darstellung von Datenelementen im Graphschema (M19) eingeschränkt geeignet. Eine detaillierte Darstellung erfolgt eher für ein ausgewähltes Datenelement dessen Inhalt isoliert dargestellt wird.

3.4.3.2 Schema

Ein Schema gibt die maßgebliche Strukturierung der Darstellung vor. In diesem Musterkatalog werden die Schemata Graph, Baum, Block, Code und Tabelle unterschieden. Diese Schemata unterscheiden sich hinsichtlich ihrer Ausdrucksstärke und schränken so den Informationsgehalt der Prozessdarstellung unterschiedlich stark ein. Ein Prozess einer graphbasierten Prozesssprache kann zwar in Form eines Baumes dargestellt werden. Die durch Kontrollverbindungen festgelegte Ausführungsreihenfolge kann diesem Baum allerdings nicht entnommen werden. Für eine blockstrukturierte Prozesssprache ist die Baumdarstellung hingegen unproblematisch.

Muster 19: Graphschema verwenden

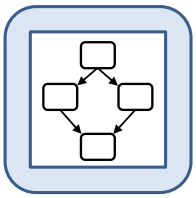


Abbildung 3.54: Symbol zu „Graphschema verwenden (M19)“.

Problem: Wie kann die Darstellung eines Prozesses strukturiert werden?

Schlagwörter: Prozessgraph darstellen; Flussdiagramm darstellen.

Beschreibung: Das Graphschema ist ein weit verbreitetes Schema, um einen Prozess darzustellen und ihn für die grafische Modellierung zugänglich zu machen. Analog zum PM-Graphen stellen Aktivitäten die Knoten des Graphen dar. Das Graphschema kann verschiedene Arten von Kanten beinhalten, etwa für Kontrollverbindungen und Datenverbindungen. Die Skizze in Abbildung 3.55 zeigt die Darstellung einer Menge von ECA Regeln im Graphschema.

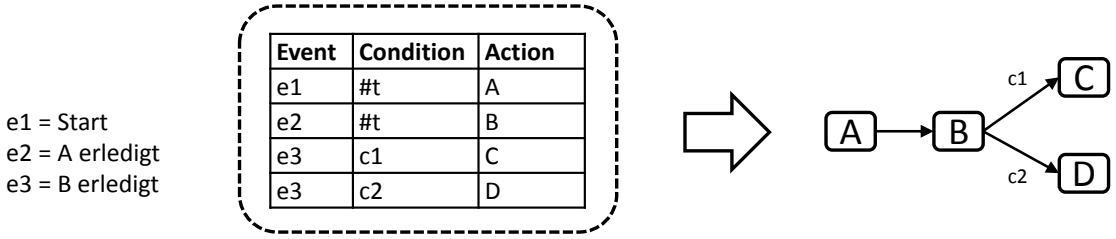


Abbildung 3.55: Skizze zu „Graphschema verwenden (M19)“.

Beispiele: Die heutzutage verbreiteten Modellierungswerkzeuge bieten i.d.R. eine grafische Darstellung der Prozessstruktur in Form eines Graphen an. Häufig zeigt diese Sicht lediglich die Aktivitäten mit einem Aktivitätstypsymbol und Namen sowie den Kontroll- und Datenfluss, der die Aktivitäten verbindet. Details zu den Aktivitäten sind im Graphschema nicht enthalten. Sie werden erst in unmittelbarer Nähe des Mauszeigers (MouseOver) oder durch explizite Selektion einer bestimmten Aktivität dargestellt, wie bspw. in dem Werkzeug Eclipse BPEL Designer [Ecl11] in einem Hinweisfenster (engl. *tooltip*) oder in der Eigenschaftenansicht (engl. *properties view*).

Muster 20: Baumschema verwenden

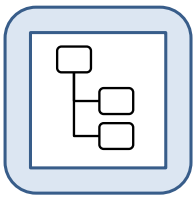


Abbildung 3.56: Symbol zu „Baumschema verwenden (M20)“.

Problem: Wie kann die Darstellung eines Prozesses hierarchisch strukturiert werden?

Schlagwörter: Prozessbaum darstellen; Baumansicht verwenden.

Beschreibung: Die Baumansicht strukturiert die Elemente eines Prozessgraphen in Form eines Baumes. Tief verschachtelte sequentielle Strukturen sind auf diese Weise einfach zugänglich und navigierbar. Allerdings kann dieses Schema den Informationsgehalt von graphbasierten Sprachen nur eingeschränkt darstellten. Die Skizze des PM-Graphen *P* in Abbildung 3.57 zeigt einen problematischen Fall. Nach der Ausführung der Aktivität *A* wird parallel die Ausführung von *B* und *D* angestoßen. Ob als nächste die Aktivität *C* oder *E* ausgeführt werden wird ist dem Kontrollfluss nicht zu entnehmen. In einem Bauschema werden die Aktivitäten daher ggf. als Blätter abgebildet, unter Verzicht auf Informationen über den Kontrollfluss.

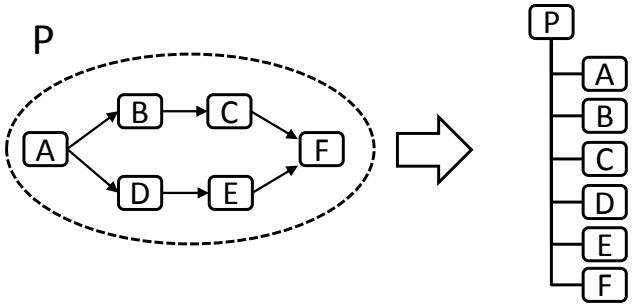


Abbildung 3.57: Skizze zu „Baumschema verwenden (M20)“.

Beispiele: Die Darstellung des Prozessgraphen im Baumschema wird in dem Modellierungswerkzeug Eclipse BPEL Designer [Ecl11] in einer Sicht zum Prozessentwurf eingesetzt (engl. *outline view*). Im Baum werden einfache Aktivitäten als Blätter und strukturierte Aktivitäten, wie bspw. ein While-Block, als Zweige abgebildet. Der Baum enthält außerdem Datenelemente und weitere Elemente entsprechend ihrer Gültigkeitsbereiche (engl. *scopes*). Eine musterbasierte Analyse dieses Werkzeugs wird in Abschnitt 10.4.2 geleistet.

Muster 21: Blockschema verwenden

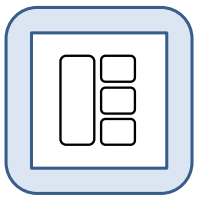


Abbildung 3.58: Symbol zu „Blockschema verwenden (M21)“.

Problem: Wie kann die Darstellung eines Prozesses blockweise strukturiert werden?

Schlagwörter: Blockstruktur darstellen; Strukturiert darstellen.

Beschreibung: In einem Blockschema werden Aktivitäten blockweise aneinandergereiht. Wie in Abbildung 3.59 skizziert, erfordert dieses Schema, dass das Erscheinungsbild von Aktivitäten (M16) angepasst wird, d.h. in die Länge oder Breite gezogen wird. Das Blockschema hat ebenso wie das Baumschema (M20) weniger Ausdruckskraft als das Graphschema (M19).

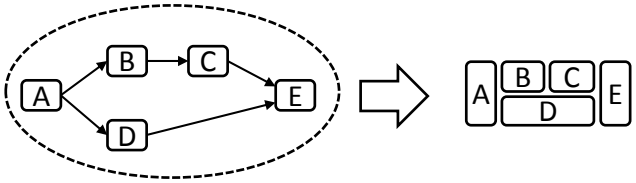


Abbildung 3.59: Skizze zu „Blockschema verwenden (M21)“.

Beispiele: Die Prozesssprache BPEL enthält blockstrukturierte Aktivitäten wie `<sequence>`, `<while>` oder `<if>`. Die Sprache enthält außerdem die Möglichkeit, graphbasierte Strukturen innerhalb eines `<flow>` zu modellieren. Ein Werkzeug zur grafischen Modellierung von BPEL müsste daher sowohl das Graphschema (M19) als auch das Blockschema (M21) unterstützen. Das Werkzeug Eclipse BPEL Designer [Ecl11] verwendet eine Mischform und stellt auch Blockstrukturen im Graphschema (M19) dar. Die im Aufbau von Blöcken implizierten Kontrollverbindungen werden in dem Werkzeug explizit als Kanten dargestellt.

Muster 22: Codeschema verwenden

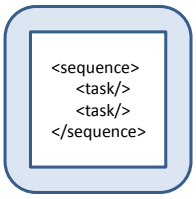


Abbildung 3.60: Symbol zu „Codeschema verwenden (M22)“.

Problem: Wie kann die Darstellung eines Prozesses textuell strukturiert werden?

Schlagwörter: Textuell darstellen; Codeansicht verwenden.

Beschreibung: Der Umgang mit dem Codeschema setzt ein technisches Verständnis beim Nutzer voraus, da es mitunter stark an Programmiersprachen angelehnt ist. Das Codeschema hat große Relevanz in der Automatisierung von Geschäftsprozessen, da im Codeschema alle technischen Details unmittelbar zugänglich sind und letztlich der Code bzgl. der Ausführungssemantik entscheidend ist. Bspw. kann ein Layout, das durch den Modellierer vorgegeben ist (M13, M28), eine bestimmte Abfolge der Fallprüfungen bei einer Verzweigung suggerieren. Bei einer Orientierung des Prozesses von links nach rechts (M24) wird dem Leser etwa suggeriert, dass die oberste Bedingung zuerst geprüft wird, vgl. dazu die hervorgehobenen Bedingungen in Abbildung 3.61 (a). Tatsächlich wird die Bedingung *a* ggf. nicht an erster Stelle geprüft, wie die Reihenfolge im Codeschema illustriert (b).

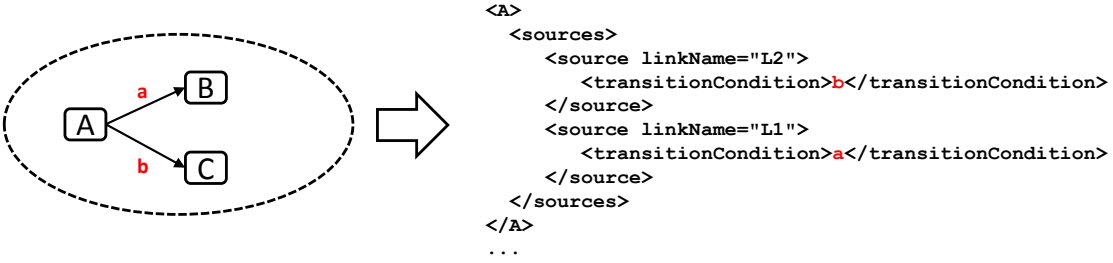


Abbildung 3.61: Skizze zu „Codeschema verwenden (M22)“.

Beispiele: Viele Workflow Management Systeme (WfMS) setzen XML als Austauschformat für Prozessmodelle ein. XML kann in einem Standardtexteditor angezeigt und manipuliert werden. Neben einem Texteditor gibt es zahlreiche freie wie auch kommerzielle Werkzeuge zur Anzeige und Manipulation von XML. Diese Werkzeuge setzen häufig eine Syntaxhervorhebung zur besseren Unterscheidbarkeit von XML-Elementen, XML-Attributen und Werten ein (M16, M17, M18). Des Weiteren können die Ergebnisse einer Syntaxprüfung (M14) durch eine Farbcodierung von fehlerhaften Codepassagen dargestellt werden. Das Codeschema hat mitunter den Nachteil, dass entsprechende Werkzeuge neben einer Einrückung von Code keine weitere Möglichkeit zum Layout bieten (vgl. M24-M28).

Wie Bischof et al. [BKL+09] zeigen, kann ein in XML vorliegender BPEL-Prozess nach einer Übersetzung in der JavaScript-ähnlichen Syntax BPELscript dargestellt werden.

Muster 23: Tabellenschema verwenden

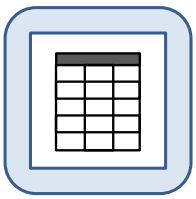
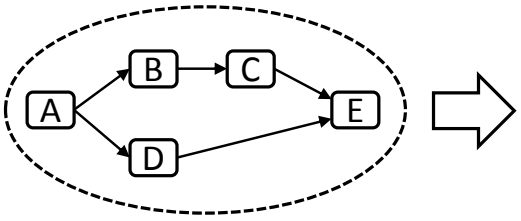


Abbildung 3.62: Symbol zu „Tabellenschema verwenden (M23)“.

Problem: Wie kann die Darstellung eines Prozesses matrixförmig strukturiert werden?

Schlagwörter: Tabellarisch darstellen; Listenansicht verwenden; Raster bilden.

Beschreibung: Das Tabellenschema ist ein mächtiges Schema zur übersichtlichen Darstellung von Prozessinformationen. Zeilen eignen sich zur Darstellung für einzelne Elemente, vornehmlich für Aktivitäten, während Spalten eher für Eigenschaften verwendet werden. Tabellen eignen sich zwar nur bedingt zur Darstellung der Kontrollabhängigkeiten in komplexen Prozessgraphen, können aber durch Sortiermöglichkeiten zur Analyse herangezogen werden. Tabellen können auch als übersichtliche, prozessbezogene Checklisten eingesetzt werden. Sofern erforderlich, können Verzweigungen und Verschachtelungen optisch durch einen Einzug kenntlich gemacht werden. Alternativ ist es möglich, Kontrollverbindungen anhand einer Adjazenzmatrix explizit aufzuführen, wie in Abbildung 3.63 skizziert. Jede Zeile in dieser Tabelle repräsentiert eine Aktivität. Ein *x* zeigt eine Kontrollabhängigkeit von einer Quellaktivität (Zeile) zu einer Zielaktivität (Spaltenkopf) an.



Name	Typ	A	B	C	D	E
A	Service		X		X	
B	Aufgabe			X		
C	Aufgabe					X
D	Aufgabe					X
E	Service					

Abbildung 3.63: Skizze zu „Tabellenschema verwenden (M23)“.

Beispiele: Im Werkzeug Enterprise Architect 10 [Spa13] kann ein Prozessgraph bei Bedarf im Tabellenschema ausgelegt werden (List View), wobei Zeilen einzelne Modellelemente repräsentieren und Spalten Eigenschaften wie Symbol, Name, Typ oder Zustand abbilden.

Das Prozessmodellierungswerkzeug Signavio Process Editor [Sig13] bietet mit der Funktion QuickModel die Möglichkeit, einen Prozess tabellenbasiert aufzunehmen und daraus automatisch einen Prozessgraphen nach BPMN zu erzeugen.

3.4.3.3 Layout

Mit den Mustern zum Layout können Anforderungen an die Darstellung in einem gewählten Schema genauer spezifiziert werden.

Muster 24: Orientierung festlegen

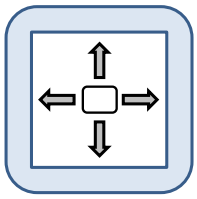


Abbildung 3.64: Symbol zu „Orientierung festlegen (M24)“.

Problem: Wie kann die Leserichtung in die Darstellung einbezogen werden?

Schlagwörter: Horizontal auslegen; Vertikal auslegen; Richtung festlegen.

Beschreibung: Die Orientierung gibt vor, in welcher Richtung ein Prozess in dem gewählten Schema ausgelegt werden soll. Generell ist eine andere Orientierung als von links nach rechts oder von oben nach unten eher unüblich. Die Skizze in Abbildung 3.65 zeigt für einen Prozess im gewählten Blockschema das Layout in der Orientierung von oben nach unten. Der Skizze ist zu entnehmen, dass zwischen der Orientierung und dem Erscheinungsbild der darzustellenden Prozesselemente (M16, M17, M18) eine Abhängigkeit bestehen kann.

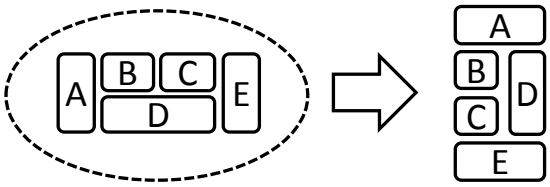


Abbildung 3.65: Skizze zu „Orientierung festlegen (M24)“.

Beispiele: Im Graphschema (M19) ist im technischen Umfeld, wie bspw. bei Flussdiagrammen im Softwareentwurf, eine Orientierung von oben nach unten üblich. Im Kontext von Geschäftsprozessen verläuft das Layout eher von links nach rechts. Analog kann diese Orientierung im Blockschema (M21) verwendet werden. Im Baumschema (M20) hingegen wird ein Prozess i.d.R. nur von oben nach unten ausgelegt. Im Tabellenschema (M23) legt die

Orientierung fest, ob Aktivitäten und Datenelemente als Zeilen oder als Spalten ausgelegt werden sollen. Im Codeschema (M22) ist von oben nach unten gängig.

Bei einer Änderung der Orientierung ist ein Umdenken bei den Betrachtern erforderlich, denn die Aktivitäten befinden sich nach einer Anpassung der Orientierung nicht mehr an gewohnter Stelle.

Muster 25: Phasenweise auslegen

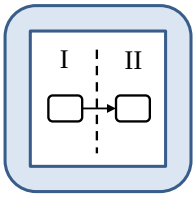


Abbildung 3.66: Symbol zu „Phasenweise auslegen (M25)“.

Problem: Wie kann die Darstellung in verschiedene Bereiche unterteilt werden?

Schlagwörter: Segmentieren; Gliedern; Meilensteine darstellen; Bereichsansicht verwenden.

Beschreibung: Das Prinzip von Phasen ähnelt dem „teile und herrsche“-Prinzip, da es einen Prozess in mehrere Bereiche unterteilt und jeder Bereich gesondert betrachtet werden kann. Ein Layout nach Phasen setzt Informationen über vorhandene Phasen und die Phasenzugehörigkeit der Aktivitäten voraus. Sofern diese Informationen nicht im Prozessmodell enthalten sind, müssen diese angereichert werden (M13, M14, M15). Die Skizze in Abbildung 3.67 zeigt das Layout im Graphschema mit den drei Phasen *I*, *II* und *III*.

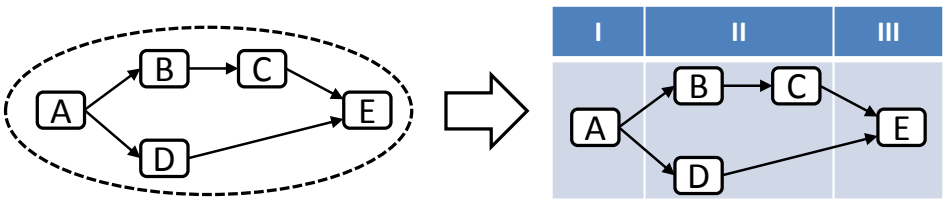


Abbildung 3.67: Skizze zu „Phasenweise auslegen (M25)“.

Beispiele: Das Prinzip von Phasen kann bspw. auf Workflows zur Simulationssteuerung angewendet werden. Sonntag et al. [SGK+11] schlagen die Aufteilung eines solchen Workflows in die Phasen *Pre-Processing*, *Simulation* und *Post-Processing* vor. Die Kennzeichnung der Aktivitäten erfolgt automatisch sofern für die aufgerufenen Dienste eine Phasenzugehörigkeit bekannt ist (M15) oder manuell durch nutzergestützte Anreicherung (M13). Informationen zur Phasenzugehörigkeit können außerdem die Aggregation von Aktivitäten (M2, M3) steuern [Bob08]. Dieser Ansatz wird bspw. in dem Modellierungswerkzeug Enterprise Architect 10 [Spa13] im Tabellenschema (M23) verwendet, in dem Elemente nach der Eigenschaft „Phase“ gruppiert und aggregiert werden können (M2, M3).

Muster 26: Organisatorisch auslegen

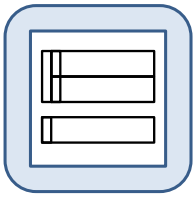


Abbildung 3.68: Symbol zu „Organisatorisch auslegen (M26)“.

Problem: Wie können organisatorische Zusammenhänge dargestellt werden?

Schlagwörter: Organisationseinheiten darstellen; Rollenansicht verwenden.

Beschreibung: Ein Prozessmodell kann organisatorische Informationen bereits enthalten oder diese Informationen können in einem Anreicherungsschritt integriert werden (M14, M15). Die Informationen können abstrakt beschaffen sein wie eine Rolle, die einer Aktivität zugeordnet ist, oder konkret wie eine Person, die eine Aktivität einer bestimmten Prozessinstanz durchführt. Wie in Abbildung 3.69 mit farbcodierten Swimlanes skizziert, kann diese Information für ein organisatorisches Layout genutzt werden. Dieses Layout wird hauptsächlich bei Unternehmensprozessen im Graphschema (M19) eingesetzt.

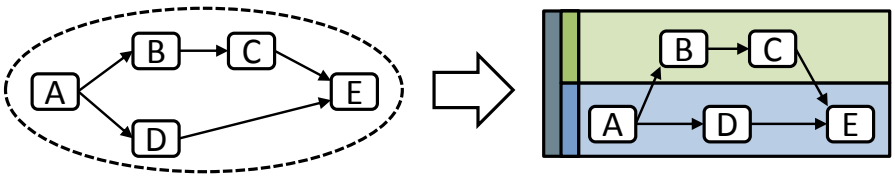


Abbildung 3.69: Skizze zu „Organisatorisch auslegen (M26)“.

Beispiele: In BPMN stehen Pools für einen Partner in einer Choreographie und Swimlanes werden für interne Organisationseinheiten verwendet. Swimlanes können hierarchisch gruppiert werden. Neben organisatorischer Information wie der Abteilung oder Rolle kann prinzipiell jede beliebige Eigenschaft einer Aktivität zur Zuordnung eingesetzt werden. Dies ist bspw. in dem Modellierungswerkzeug WebSphere Business Modeler [IBM09] durch Auswahl der abzubildenden Eigenschaft realisiert.

Durch die Abstraktion von Eigenschaften (M4) kann der Abstraktionsgrad des organisatorischen Layouts durch eine Aggregation von Swimlanes angepasst werden. Sind z.B. zwei unterschiedliche Teams derselben Abteilung in einen Prozess involviert, können diese nach der Abstraktion dieser organisatorischen Eigenschaft als eine Einheit auf Abteilungsebene betrachtet werden.

Muster 27: Räumlich auslegen

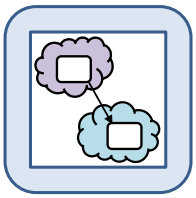


Abbildung 3.70: Symbol zu „Räumlich auslegen (M27)“.

Problem: Wie können räumliche Zusammenhänge dargestellt werden?

Schlagwörter: Verteilung darstellen; Standorte darstellen.

Beschreibung: Das räumliche Layout bewirkt eine Gruppierung und Positionierung nach Aspekten der räumlichen Verteilung. Durch Informationsanreicherung (M13, M14, M15) können Aktivitäten mit zahlreichen Eigenschaften erweitert werden, die zum Layout eingesetzt werden können. Das räumliche Layout erfordert eine spezifische Hintergrundgrafik, auf der die Prozesselemente ausgelegt werden sollen. Es unterscheidet sich durch den konkreten räumlichen Bezug von dem deutlich abstrakteren organisatorischen Layout (M26). Abbildung 3.71 skizziert ein räumliches Layout auf einem Architekturdiagramm. Die Aktivitäten *A* und *E* werden von einem Benutzer durchgeführt; die Aktivitäten *B* und *C* sind durch einen Dienst realisiert; Aktivität *D* ist in einem Datenhaltungssystem verortet.

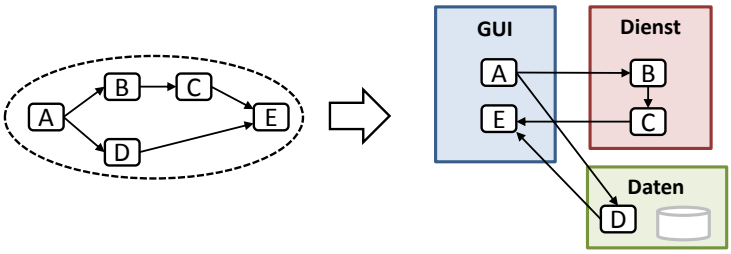


Abbildung 3.71: Skizze zu „Räumlich auslegen (M27)“.

Beispiele: Geschäftsprozesse sind mitunter über verschiedene Niederlassungen, Rechenzentren, Unternehmen, Länder und Kontinente verteilt, die sich in ihrer Zeitzone, Währung und Rechtsprechung deutlich unterscheiden können. Ein räumliches Layout, bspw. mit der Hintergrundgrafik einer Weltkarte oder einer Landkarte kann diese komplexen Sachverhalte veranschaulichen und dem Betrachter helfen, Zusammenhänge herzustellen.

Muster 28: Maßgeschneidert auslegen

Abbildung 3.72: Symbol zu „Maßgeschneidert auslegen (M28)“.

Problem: Wie können spezifische Anforderungen an das Layout berücksichtigt werden?

Schlagwörter: Benutzerdefiniert darstellen; Speziell darstellen.

Beschreibung: Das maßgeschneiderte Layout deckt die individuellen und kreativen Formen ab, die neben den bereits in Mustern beschriebenen gängigen Formen zum Layout bestehen und auf bestimmte Prozesssprachen, Interessengruppen oder Aufgaben zugeschnitten sind. Häufig erfolgt ein maßgeschneidertes Layout entsprechend bestehender Vorgaben von Nutzern, d.h. Aktivitäten und Datenelemente wurden manuell auf einer Zeichenfläche angeordnet (M13), vgl. die Skizze in Abbildung 3.73.

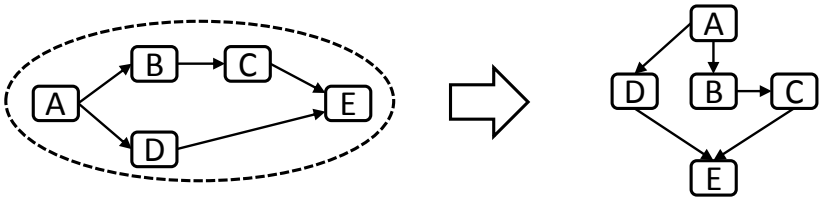


Abbildung 3.73: Skizze zu „Maßgeschneidert auslegen (M28)“.

Beispiele: Unter Zuhilfenahme von dynamischer Anreicherung (M15) und durch eine gestreckte Darstellung von Aktivitäten (M16) kann ein Prozess auf treppenförmige Weise maßgeschneidert ausgelegt und damit einem Gantt-Diagramm nachempfunden werden.

3.4.4 Muster von Vorgaben zur Durchführung von Transformationen

Die in diesem Abschnitt vorgestellten Sichtenentwurfsmuster sind übergreifende Aspekte, die zur Formulierung gängiger Anforderungen dienen, die an die Durchführung von Transformationen gestellt werden und damit Vorgaben zu deren Verhalten festlegen. Die Muster wurden durch Analyse gängiger Anforderungen im Stand der Technik von Sichten und Compliance identifiziert und, wie nachfolgend erläutert, auf den musterbasierten Sichtenentwurf abgebildet.

Die für Transaktionen bei Datenbanken geforderten ACID-Eigenschaften (Atomicity, Consistency, Isolation, Durability) könnten prinzipiell als Vorgaben für Transformationen eingesetzt werden. Sie sind allerdings für bestimmte Anwendungsfälle zu restriktiv, denn nicht bei jeder Transformation wird die „All or Nothing“-Semantik (Atomicity) bezweckt. Ebenso ist nicht in jeder Sicht die Einhaltung aller für eine Prozesssprache gültigen Konsistenzregeln (Consistency) erforderlich. Transformationen auf einen Eingabeprozess werden nach dem Stand der Technik sequentiell durchgeführt, daher kann die Betrachtung von Nebenläufigkeit vernachlässigt werden (Isolation). Die Verwendung einer Sicht als neues, gespeichertes Prozessmodell (Durability) stellt nicht die Regel dar; weit häufiger ist eine Sicht eine virtuelle, auf Nutzeranfrage generierte temporäre Darstellung eines Prozesssachverhalts. Sofern eine Sicht im Einzelfall dennoch als neues Prozessmodell gespeichert werden soll, kann dies im Zuge des Feinentwurfs festgelegt werden (vgl. Abschnitt 3.6).

Im Sichtenentwurf ist eine Unterscheidung hinsichtlich Atomicity im Einzelfall, d.h. für jede Musteranwendung gesondert zu betrachten. Consistency kann sich auf eine Menge von Musteranwendungen beziehen und ist in den Vorgabemustern Konsistenz bewahren (M29) und Ausführbarkeit bewahren (M30) als gängige Anforderung im Sichtenentwurf reflektiert.

Neben den ACID-Eigenschaften sind im Sichtenentwurf weitere Vorgaben relevant. Eine im Bereich der Modelltransformation bewährte Technik, die auch im Bereich der Sichtentransformation angewendet werden kann, besteht im Bewahren von Strukturen (M31). Vorgaben zur Protokollierung (M32) und zur Gewährleistung von Nachvollziehbarkeit (M33) sind Anforderungen aus dem Umfeld von Compliance [SAL+10a].

Muster 29: Konsistenz bewahren

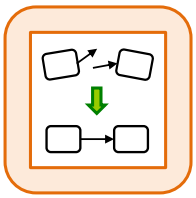


Abbildung 3.74: Symbol zu „Konsistenz bewahren (M29)“.

Problem: Wie können in einem Prozess bestehende Zusammenhänge aufrechterhalten werden, wenn die Struktur des Prozesses verändert wird?

Schlagwörter: Syntaktische Korrektheit erhalten; Syntaxfehler verhindern.

Beschreibung: Wenn Strukturveränderungen an einem Prozess vorgenommen werden (M1-M12) oder ein Prozess mit Informationen angereichert wird (M13-M15), können Konsistenzregeln des Prozessmetamodells verletzt werden. Die Regeln, nach denen Konsistenz überprüft werden kann, sind allerdings von Prozesssprache zu Prozesssprache verschieden.

Die in Abbildung 3.75 skizzierten Transformationen T_1 und T_2 stellen beide eine Anwendung des Musters Aktivitäten entfernen (M1) dar, vgl. den Eingangsprozess (a). T_1 stellt die Anwendung des Entfernens unter Bewahrung der Konsistenz dar. Die transitive Kontrollverbindung zwischen *B* und *E* wird daher aufrechterhalten. T_2 stellt die Anwendung ohne Einhaltung der Vorgabe dar. Der Abgleich, welche Transformation welche Konsistenzeigenschaften garantiert, ist eine Problemstellung des Feinentwurfs und der Implementierung.

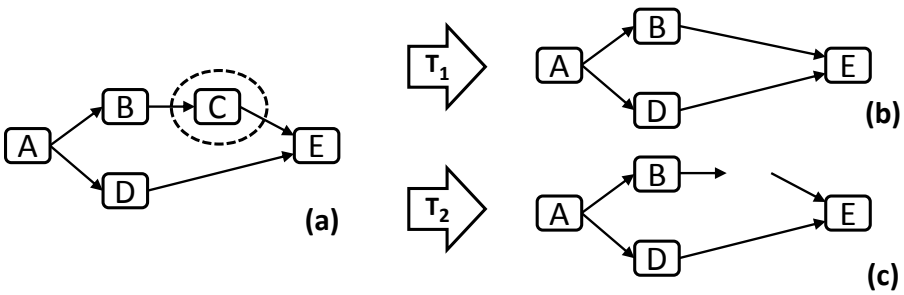


Abbildung 3.75: Skizze zu „Konsistenz bewahren (M29)“ mit Entfernen der Aktivität *C*, wobei (b) den Ausgabeprozess unter Einhaltung der Vorgabe und (c) ohne dessen Einhaltung darstellt.

Beispiele: Eine Konsistenzregel kann bspw. angeben, ob ein Prozessgraph zusammenhängend sein muss und ob nur teilweise verbundene Kontrollverbindungen erlaubt sind. Letzteres ist i.d.R. nicht der Fall, wie bspw. auch im PM-Graphen. Weiterhin können Konsistenzregeln festlegen, welche Aktivitäten welche Eigenschaften zwingend benötigen

oder in welcher Abfolge bestimmte Aktivitäten stehen müssen. Konsistenzregeln können außerdem angereicherte Informationen betreffen, bspw. definiert Bobrik Konsistenzeigenschaften über die Zustände von Aktivitäten innerhalb eines Pfades [Bob08].

Muster 30: Ausführbarkeit bewahren

Abbildung 3.76: Symbol zu „Ausführbarkeit bewahren (M30)“.

Problem: Wie kann die Ausführbarkeit eines Prozesses aufrechterhalten werden, wenn die Struktur des Prozesses verändert wird?

Schlagwörter: Syntaktische Korrektheit erhalten; Syntaxfehler vermeiden.

Beschreibung: Das Bewahren der Ausführbarkeit dient als Vorgabe zur Beschränkungen von Transformationen im Rahmen der Generierung und Materialisierung einer Sicht im Sinne der Variantenbildung. Für prozessdarstellende Sichten ist das Muster kaum relevant. Es schließt das Bewahren von Konsistenz (M29) ein und ist mit weiteren Anforderungen verbunden.

Bereits eine Modifikation einer einzigen Eigenschaft kann die Ausführbarkeit eines Prozesses beeinträchtigen. Auch können Deadlocks auftreten wenn bestimmte Aktivitäten entfernt werden. Die Skizze in Abbildung 3.77 zeigt das Entfernen des Pfades $A \rightarrow E$ mit der Annahme, dass Aktivität *B* zur Ausführbarkeit erforderlich ist. Transformation T_1 generiert den Ausgabeprozess, der die Vorgabe der Ausführbarkeit nicht erfüllt (b). T_2 realisiert das Entfernen unter Einhaltung der Vorgabe mit Atomicity (c). T_3 führt das Entfernen soweit möglich durch, womit Aktivität *C* entfernt wird, aber die zur Ausführbarkeit erforderliche Aktivität *B* erhalten bleibt (d). Was eine korrekte Ausführbarkeit im Detail umfasst, muss im Feinentwurf und der Implementierung festgelegt werden.

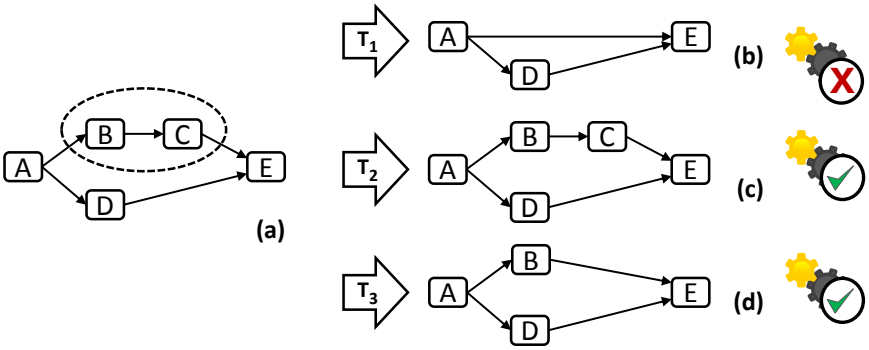


Abbildung 3.77: Skizze zu „Ausführbarkeit bewahren (M30)“.

Beispiele: Eine Sicht auf einen ausführbaren Prozess kann dazu eingesetzt werden, in dem Prozess enthaltene Hilfsaktivitäten dauerhaft zu entfernen (M1), wobei die Ausführbarkeit des strukturreduzierten Prozesses erhalten werden muss (M30). Bspw. können Aktivitäten zur

Gewährleistung einer Protokollierung oder zur Unterstützung des Monitorings in einem Prozessmodell enthalten sein, die bei Bedarf entfernt werden können. Eine derartige Sicht bewirkt die Bildung einer persistierbaren Variante (vgl. Durability).

Muster 31: Strukturen bewahren

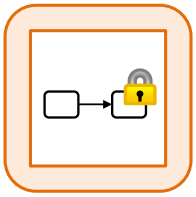


Abbildung 3.78: Symbol zu „Strukturen bewahren (M31)“.

Problem: Wie können bestimmte Strukturen von Transformationen ausgenommen werden?

Schlagwörter: Strukturen erhalten; Ausnahme festlegen; Sonderfall beschreiben; Selektieren.

Beschreibung: Mit dieser Vorgabe kann die Anforderung formuliert werden, bestimmte Elemente oder deren Eigenschaften über Transformationen hinweg nicht zu verändern. Für bestimmte Aufgaben, Fragestellungen und Interessengruppen relevante Aktivitäten können durch diese Vorgabe von Transformationen ausgenommen werden. Dieses Muster hat u.a. Einfluss auf Strukturveränderungen wie auf das Entfernen von Aktivitäten (M1), Eigenschaften (M4) und Datenelementen (M5). Die Skizze in Abbildung 3.79 zeigt das Entfernen (M1) des Pfades *A→E* mit der Vorgabe *C* zu bewahren. Diese Transformation ist mehrdeutig und kann auf zwei Arten durchgeführt werden: T_1 generiert einen Ausgabeprozess, der das Entfernen des Pfades *A→E* soweit möglich durchführt (b). T_2 stellt das Ergebnis unter Einhaltung der Atomicity-Eigenschaft beim Entfernen dar (c).

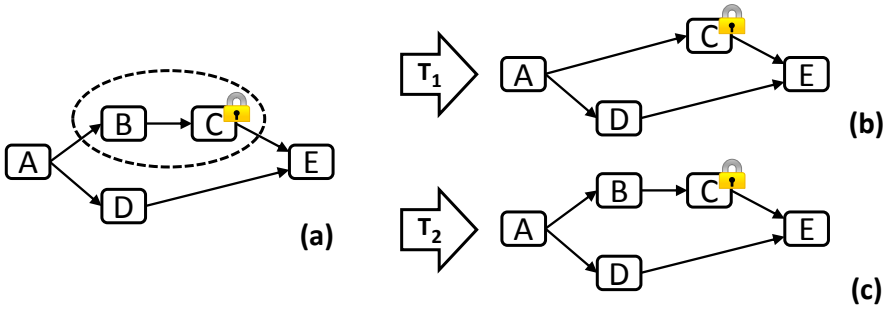


Abbildung 3.79: Skizze zu „Strukturen bewahren (M31)“.

Beispiele: Verschiedene Interessengruppen eines Prozesses können bestimmte Aktivitäten als Favoriten kennzeichnen (M13), welche ihnen als strukturelle Orientierungspunkte im Prozess dienen. Wenn eine Abstraktion durchgeführt wird, bleiben diese Orientierungspunkte erhalten und erleichtern die Navigation im reduzierten Prozess trotz einer Aktualisierung des Layouts.

Muster 32: Transformationen protokollieren

Abbildung 3.80: Symbol zu „Transformationen protokollieren (M32)“.

Problem: Wie können vorgenommene Transformationen rekonstruiert werden?

Schlagwörter: Protokollieren; Aufzeichnen; Rekonstruieren.

Beschreibung: In einer Sicht können verschiedene Transformationen mit Implementierungen unterschiedlicher Hersteller zusammenwirken und verschiedene Datenquellen einbinden. Das Protokollieren liefert die Informationen, die zur Rekonstruktion der durchgeführten Transformationen benötigt werden, vgl. die Skizze in Abbildung 3.81. Außerdem unterstützt die Protokollierung die Gewährleistung der Nachvollziehbarkeit (M33).

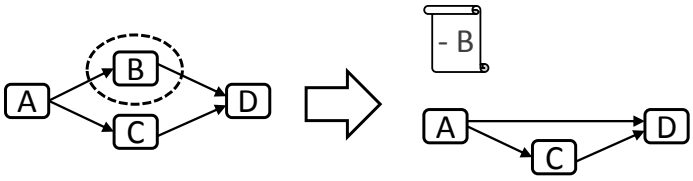


Abbildung 3.81: Skizze zu „Transformationen protokollieren (M32)“.

Beispiele: Die Protokollierung ist nicht nur im Fehlerfall für ein Debugging hilfreich, sondern kann auch zum Test einer Sichtenimplementierung erforderlich sein. Eine Protokollierung könnte zu jeder Transformation bspw. umfassen:

- den Zeitpunkt des Beginns und des Endes der Transformation,
- die Implementierung, die die Transformation durchgeführt hat,
- welche Strukturen die Transformation betreffen sollte,
- welche Vorgaben bei der Transformation berücksichtigt wurden,
- welche Strukturen die Transformation effektiv betroffen hat,
- welche Modifikationen an diesen Strukturen vorgenommen wurde,
- ob und wie ein Nutzer in die Transformation involviert war und
- welche Datenquellen bei der Transformation eingebunden wurden.

Muster 33: Nachvollziehbarkeit gewährleisten

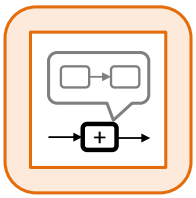


Abbildung 3.82: Symbol zu „Nachvollziehbarkeit gewährleisten (M33)“.

Problem: Wie können vorgenommene Transformationen sicher nachvollzogen werden?

Schlagwörter: Protokollieren; Aufzeichnen; Rekonstruieren; Rückverfolgen; Kontrollieren.

Beschreibung: Eine abstrahierende Sicht kann dazu verwendet werden, unwichtige Details auszulassen. Sie kann allerdings auch dazu eingesetzt werden, problematische Aspekte bewusst zu verbergen. Die Vorgabe zur Gewährleistung der Nachvollziehbarkeit unterstützt den kritischen Betrachter einer Sicht auf einen Prozess dabei, das eine vom anderen zu unterscheiden. Eine Protokollierung (M32) stellt eine wichtige Grundlage für die Nachvollziehbarkeit dar, bietet jedoch eine Angriffsfläche zur Manipulation. Zum Beispiel könnten außerhalb der protokollierten Transformationsschritte Änderungen an dem Prozess vorgenommen werden, Protokolleinträge können entfernt und Zeitstempel modifiziert werden. Im Stand der Technik tragen mehrere Ansätze zur Gewährleistung von Nachvollziehbarkeit bei:

- Mittels eindeutiger Identifikationsmerkmale, die durchgängig erhalten bleiben, kann die Spur aller im Prozess enthaltenen und transformierten Elemente nachvollzogen werden. Holmes et al. [HZD+10] verwenden bspw. Universally Unique Identifier (UUID) nach ISO/IEC 9834-8 zur Nachvollziehbarkeit im Bereich Compliance.
- Durch die Zertifizierung einer Implementierung von einem unabhängigen Dritten kann Vertrauen in die Einhaltung von Vorgaben aufgebaut werden. Durch sogenannte Compliance Level Agreements (CLA) [BAS+10] können Prüfumfänge, zu verwendende Zertifizierungsmethoden und Prüfintervalle spezifiziert werden.
- Durch gesicherte Zeitstempel können Abfolgen nachgewiesen werden [NWG01].
- Neben Prüfsummen und Signaturen muss zudem der Zugriff auf erzeugte Protokolle zugangsbeschränkt werden, wie in Abbildung 3.83 skizziert.

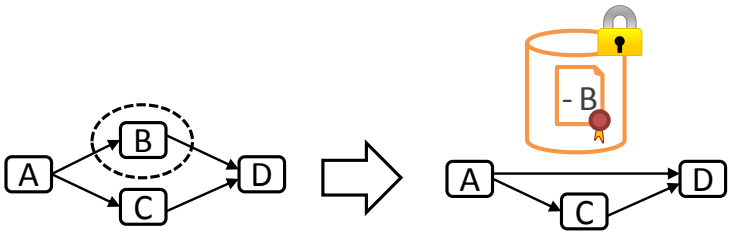


Abbildung 3.83: Skizze zu „Nachvollziehbarkeit gewährleisten (M33)“.

Beispiele: Sofern bei einer Betriebsprüfung abstrahierte Formen von Geschäftsprozessen bewertet werden, kann die Gewährleistung der Nachvollziehbarkeit der Abstraktionen rechtliche Relevanz besitzen.

3.4.5 Musterbeziehungen

Die vorgestellten Muster des Musterkatalogs wurden in die Kategorien Strukturveränderung, Informationsanreicherung, Darstellungsanpassung und Vorgabe unterteilt. Abgesehen von dieser Kategorisierung bestehen zwischen den Mustern weitere Beziehungen, die für verschiedene Zwecke genutzt werden können, bspw. für ein Empfehlungssystem zur Unterstützung des Sichtenentwurfs.

Dazu wurden die Beziehungsinformationen aus den Musterbeschreibungen extrahiert und in Tabelle 3.1 zusammengefasst. Auf Basis der Beziehungsinformationen kann ein Beziehungsgraph erzeugt werden, in dem Muster durch Knoten und deren Beziehungen durch Kanten repräsentiert sind. Um aus einem derartigen Graphen Empfehlungen generieren zu können, ist eine vordefinierte Menge von Beziehungstypen hilfreich. Zimmer [Zim95] schlägt in diesem Zusammenhang drei Beziehungstypen zwischen Entwurfsmustern vor: (1) *ähnlich zu*, (2) *kann kombiniert werden mit* und (3) *verwendet*. Abbildung 3.84 illustriert den Beziehungsgraphen, der die in Tabelle 3.1 zusammengefassten Beziehungen auf diese drei Typen reduziert.

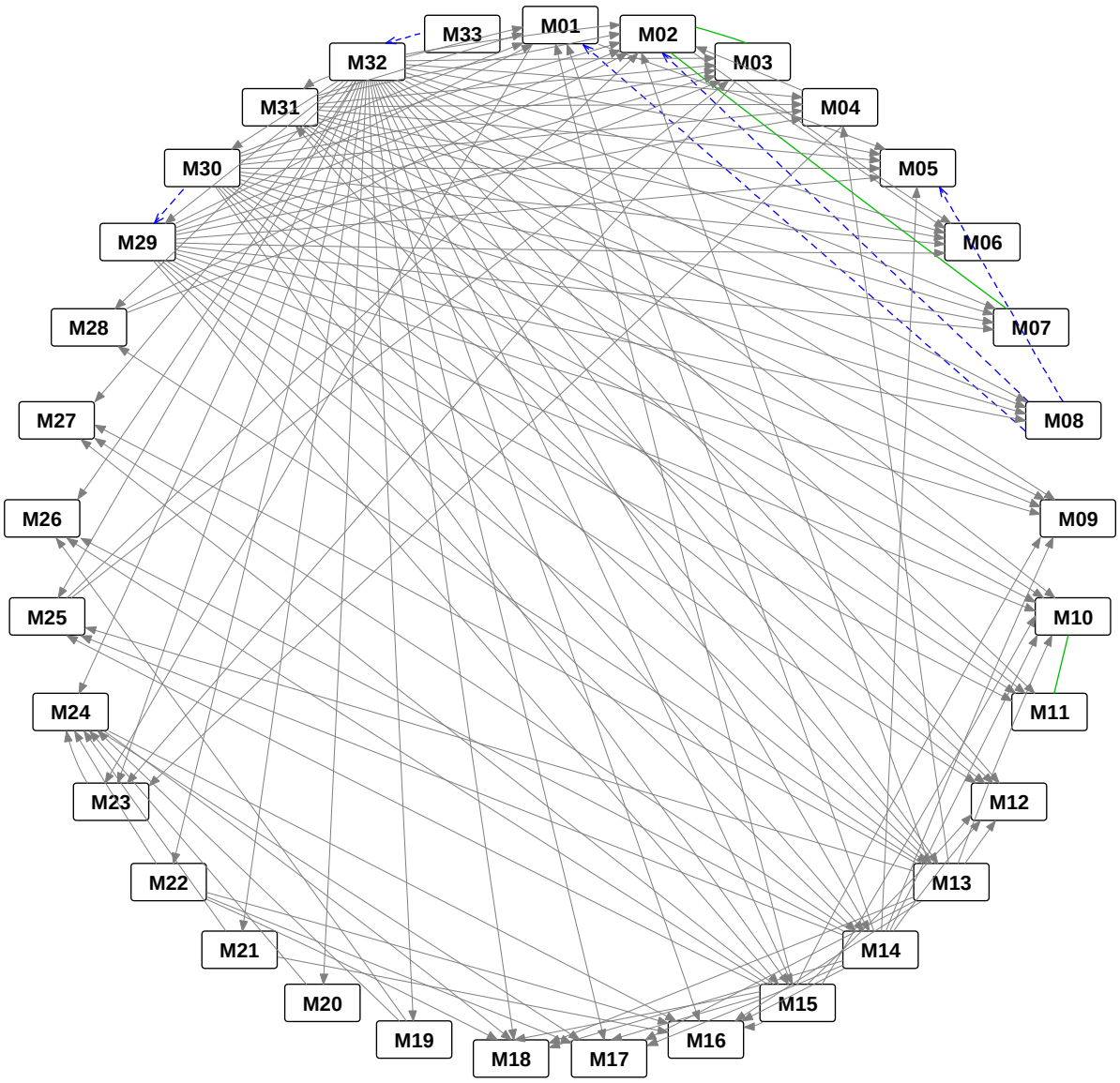


Abbildung 3.84: Beziehungsgraph der Sichtenentwurfsmuster mit den Beziehungen *gut kombinierbar* (grau), *ähnlich* (grün) und *verwendet* (blau gestrichelt).

Die Darstellung des Typs *kombinierbar* erfolgt in Grau, *ähnlich* in Grün und *verwendet* in Blau. Die Beziehung *kombinierbar* wird an dieser Stelle im Sinne von *gut kombinierbar* umgedeutet, da sich die Muster nicht gegenseitig ausschließen. Die Richtung von *kombinierbar* gibt im Beziehungsgraphen an, in welcher Reihenfolge die Muster gut kombiniert werden

können. Das Muster, von dem die Kante ausgeht, steht bei einer Kombination vorne. Die Richtung wird in Tabelle 3.1 durch einen vorangestellten oder einen nachgestellten Pfeil angegeben. Bspw. ist das Muster M1 mit dem Muster M15 gut kombinierbar, wobei die von M15 ausgehende Kante „M15→“ besagt, dass M15 bei einer Kombination dieser beiden Muster gewöhnlich vorne steht, d.h. in diesem konkreten Fall, dass ein Entfernen von Aktivitäten (M1) gut mit einer vorangehenden Anreicherung (M15) kombiniert werden kann.

Die im Beziehungsgraphen dargestellten Beziehungen vom Typ *ähnlich* umfassen nur solche (ungerichteten) Beziehungen, die explizit in den Musterbeschreibungen erwähnt werden. Aus der Strukturierung des Musterkatalogs in die Kategorien Strukturveränderung, Informationsanreicherung, Darstellungsanpassung und Vorgabe mit jeweiligen Unterkategorien ergeben sich zahlreiche weitere Ähnlichkeitsbeziehungen, die zum Erhalt der Übersichtlichkeit nicht abgebildet sind. Weitere Beziehungen bzgl. der Kombinierbarkeit von Mustern entstehen durch die gemeinsame Verwendung in Sichtenentwürfen. Auch diese sind in dem illustrierten Beziehungsgraphen nicht enthalten.

Tabelle 3.1: Musterbeziehungen der Sichtenentwurfsmuster, wobei die Beziehungstypen *kombinierbar* in Schwarz mit Kennzeichnung der Reihenfolge, *ähnlich* in Grün und *verwendet* in Blau codiert sind.

Nr.	Name	Beziehungen
M1	Aktivitäten entfernen	M15→: Nutzung von Laufzeitinformationen beim Entfernen. M29→: Zusammenhalten des Prozessgraphen.
M2	Verbundene Aktivitäten aggregieren	M13→: Einbindung und Nutzung des Prozesswissens und der Abstraktionsfähigkeit eines Nutzers. M29→: Nutzung der Konsistenzbewahrung.
M3	Unverbundene Aktivitäten aggregieren	M2: Abgrenzung durch unterschiedlichen Anwendungsbereich. →M23: In einer Tabelle können unverbundene Aktivitäten nach Eigenschaften gruppiert und hierarchisch gegliedert werden. M29→: Nutzung der Konsistenzbewahrung.
M4	Eigenschaften abstrahieren	→M2: Aktivitäten können einfacher aggregiert werden, wenn zuvor Eigenschaften abstrahiert wurden. M13→: Zuordnung von Aktivitäten zu einer Taxonomie.
M5	Datenelemente entfernen	M1→: Entfernen nicht mehr verwendeter Datenelemente nach dem Entfernen von Aktivitäten. M29→: Sofern die Vorgabe der Konsistenzbewahrung besteht, werden Datenverbindungen ebenfalls entfernt.
M6	Datenelemente aggregieren	M2→, M3→: In der vorangehenden Aggregation von Aktivitäten besteht eine weitere Möglichkeit zur Vereinfachung.
M7	Pfade aggregieren	M2: Beim alternativen Aggregieren von Aktivitäten ist die Information, dass etwas getan wird, deutlicher.
M8	Strukturen aufräumen	M1, M2, M5: Das Aufräumen kann ein Entfernen von Aktivitäten und Datenelementen beinhalten und abstrahierte Aktivitäten aggregieren.

M9	Aktivitäten einfügen	M14→, M15→: Die Informationsanreicherung kann zur Steuerung genutzt werden, an welchen Stellen welche Aktivitäten eingefügt werden sollen.
M10	Aktivitäten zerlegen	M13→, M14→, M15→: Erforderliche Informationen zum Zerlegen kommen ggf. von Datenquellen außerhalb des Prozesses.
M11	Datenelemente zerlegen	M10: Das Zerlegen von Datenelementen ist analog zum Zerlegen von Aktivitäten.
M12	Pfade zerlegen	M13→, M14→, M15→: Erforderliche Informationen zum Zerlegen kommen ggf. von Datenquellen außerhalb des Prozesses.
M13	Nutzergestützt anreichern	→M4: Informationen über die Semantik von Aktivitäten können zur Abstraktion der Beschreibung genutzt werden. →M16-M18: Die manuelle Klassifikation von Elementen kann zur Anpassung des Erscheinungsbildes eingesetzt werden. →M31: Aktivitäten, die von einer Transformation ausgenommen oder betroffen sein sollen, können manuell selektiert werden.
M14	Statisch anreichern	→M1, →M5: Eine statische Bestimmung der Relevanz kann zur abgestuften Definition von Filterregeln eingesetzt werden. →M16-M18: Mit zusätzlichen Informationen versehene Elemente können grafisch hervorgehoben werden.
M15	Dynamisch anreichern	→M1, →M9, →M10, →M16-M18, →M27: Durch die Anbindung von Informationen werden zahlreiche Fragestellungen unterstützt.
M16	Erscheinungsbild von Aktivitäten anpassen	M14→: Informationen über durchgeführte Analysen können im Erscheinungsbild kenntlich gemacht werden, z.B. mit Farbcodes. M15→: Informationen über die Ausführung von Aktivitäten können im Erscheinungsbild kenntlich gemacht werden.
M17	Erscheinungsbild von Verbindungen anpassen	M14→: Bei statischer Analyse als problematisch festgestellte Ausführungspfade farblich kennzeichnen. M15→: Laufzeitinformationen farblich kennzeichnen.
M18	Erscheinungsbild von Datenelementen anpassen	M15→: Darstellung von Laufzeitinformationen durch dynamische Informationsanreicherung.
M19	Graphschema verwenden	→M26: Im Zusammenhang mit dem organisatorischen Layout ist das Graphschema üblich.
M20	Baumschema verwenden	→M24: Im Baumschema kann ein Prozess gut von oben nach unten ausgelegt werden.
M21	Blockschema verwenden	→M16: Das Blockschema impliziert, dass das Erscheinungsbild von Aktivitäten entsprechend gestreckt wird.
M22	Codeschema verwenden	→M16-M18: Eine Farbcodierung dient der Unterscheidbarkeit, bspw. von Aktivitäten, Eigenschaften und Werten.
M23	Tabellenschema verwenden	M1→, M4→: Aktivitäten und Eigenschaften können auf einfache Weise ausgeblendet werden.

M24	Orientierung festlegen	→M16-M18: Zwischen der Orientierung und dem Erscheinungsbild der Prozesselemente kann eine Abhängigkeit bestehen. M19→: Im Graphschema ist von oben nach unten als auch von links nach rechts gängig. M20→: Im Bauschema ist von oben nach unten gängig. M21→: Im Blockschema ist von oben nach unten als auch von links nach rechts gängig. M22→: Im Codeschema ist von oben nach unten gängig. M23→: Im Tabellenschema ist von oben nach unten für Aktivitäten und von links nach rechts für Eigenschaften gängig.
M25	Phasenweise auslegen	M13→: Manuelle Kennzeichnung der Phasenzugehörigkeit. M14→: Automatische Bestimmung von Phasen. M15→: Phasenbestimmung auf Basis von Laufzeitinformation. →M2, →M3: Eine Phase kann als Grundlage zur Aggregation von Aktivitäten dienen.
M26	Organisatorisch auslegen	M14→, M15→: Organisierende Informationen können bei Bedarf in einem Anreicherungsschritt zugeordnet werden. M19→: Organisatorisches Layout ist im Graphschema gängig.
M27	Räumlich auslegen	M13-M15→: Aktivitäten können mit Eigenschaften erweitert werden, welche zum Layout verwendet werden können.
M28	Maßgeschneidert auslegen	→M2, M3: Aktivitäten können in einem Koordinatensystem ausgelegt werden, um eine optische Clusterbildung zu erzielen, die zur strukturverändernden Aggregation genutzt werden kann. M13→: Benutzer geben individuelle Koordinaten vor.
M29	Konsistenz bewahren	→M1-M12: Wenn strukturelle Veränderungen an einem Prozess vorgenommen werden, kann die Konsistenz verletzt werden. →M13-M15: Wenn ein Prozess mit Informationen angereichert wird, kann die Konsistenz verletzt werden.
M30	Ausführbarkeit bewahren	M29: Das Bewahren von Ausführbarkeit schließt das Bewahren von Konsistenz ein. →M1-M15: Bereits eine einfache Modifikation einer einzigen Eigenschaft kann die Ausführbarkeit des ganzen Prozesses beeinträchtigen.
M31	Strukturen bewahren	→M1-M12: Das Bewahren von Strukturen soll deren Entfernen oder Verändern verhindern. M13→: Interessengruppen eines Prozesses können bestimmte Aktivitäten als Orientierungspunkte kennzeichnen.
M32	Transformationen protokollieren	→M1-M31: Das Protokollieren ist ein übergreifender Aspekt, der jede Transformation betrifft.
M33	Nachvollziehbarkeit gewährleisten	M32: Das Protokollieren stellt einen wichtigen Bestandteil zur Gewährleistung der Nachvollziehbarkeit dar.

3.4.6 Vollständigkeit des Musterkatalogs

Der Musterkatalog erhebt keinen Anspruch auf Vollständigkeit. Vielmehr kann man den Katalog als erweiterbare Grundlage für den Sichtenentwurf verstehen. Ein Beispiel für eine denkbare Erweiterung des Musterkatalogs wäre das Entfernen von Kontrollverbindungen aus dem Prozessgraphen. Während ein Ausblenden von Kontrollverbindungen bereits durch die Möglichkeit gegeben ist, Kontrollverbindungen transparent darzustellen (M17), verändert diese Darstellung die eigentliche Prozessstruktur nicht und hat dadurch auch keine Auswirkung auf das Layout. Das Entfernen von Kontrollzyklen hingegen ist ein Anwendungsfall für das Entfernen von Kontrollverbindungen aus dem Prozessgraphen, bei dem der Prozessgraph dennoch zusammengehalten wird [LGC12]. Mit der Vorlage für Sichtenentwurfsmuster könnte der Musterkatalog entsprechend erweitert werden, wie Abbildung 3.85 zeigt.

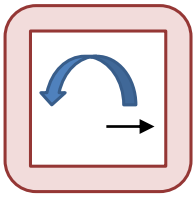


Abbildung 3.85: Mögliche Erweiterung des Musterkatalogs am Beispiel „Kontrollverbindungen entfernen“.

Neben der Erweiterung des Katalogs ist auch die Bildung von aufgaben- oder unternehmensspezifischen Teilmengen von Mustern eine Anpassungsmöglichkeit. Eine Teilmenge kann bspw. alle Muster beinhalten, die für ein bestimmtes Anwendungsgebiet nützlich sind und kann Muster aussparen, die aufgrund technischer Rahmenbedingungen oder Werkzeugeinschränkungen nicht verwendet werden können. Lägen z.B. keine Laufzeitinformationen zu einem Prozess vor, entfiele das Muster zur dynamischen Anreicherung (M15).

3.5 Klassifikation von Sichtenentwürfen

Sichten können danach klassifiziert werden, welche Muster darin angewendet werden, welchen Verwendungszweck sie haben und welcher Interessengruppe sie dienen. Sichten können außerdem den Lebenszyklusphasen von Geschäftsprozessen zugeordnet werden, wie Schumm et al. [SAL+10b] zeigen. Neben diesen Klassifikationsmerkmalen können weitere Merkmale von den Mustern abgeleitet werden, die in einem Sichtenentwurf verwendet werden. Die in diesem Abschnitt entwickelte Klassifikation von Sichtenentwürfen fokussiert auf eben diese abgeleiteten Merkmale.

3.5.1 Merkmale von Modelltransformationen

Nachfolgend werden bestehende Ansätze zur Klassifikation von Sichten und Modelltransformation zusammengefasst, die zu einer Klassifikation beitragen können.

Chinosi [Chi08] klassifiziert Sichten danach, ob sie auf einen Prozess (engl. *intra-process*) oder auf eine Choreographie angewendet werden, ob sich Änderungen in der erzeugten Sicht auf den Originalprozess zurückführen lassen (engl. *updateable*) oder nicht (engl. *read-only*)

und ob das Resultat der Transformationen ausführbar ist (engl. *executable*) oder nicht (engl. *non-executable*). Ob sich Änderungen in einer Sicht auf einen Prozess zurückführen lassen ist ein wichtiges Unterscheidungsmerkmal für einen Sichtenentwurf bzw. für dessen Implementierung. Allerdings sollte die Zwischenstufe berücksichtigt werden, dass Änderungen eingeschränkt oder nur unter Zuhilfenahme von nutzergesteuerter oder automatischer Auflösung von Mehrdeutigkeiten möglich sind. Ob das Resultat einer Transformation ausführbar ist, kann ebenfalls als essentiell betrachtet werden.

Czarnecki und Helsen [CH03] setzen die Klassifikation auf einer anderen Ebene an, indem sie eine Taxonomie zur Ordnung von Transformationsregeln aufstellen. Diese Transformationsregeln sind am ehesten mit einer konkreten Implementierung einer Musteranwendung zu vergleichen. Dadurch lassen sich die Klassifikationsmerkmale nach Czarnecki und Helsen nur bedingt zur Klassifikation für den Entwurf heranziehen. Zwei Merkmale haben dennoch Relevanz für Sichten: die Nachvollziehbarkeit und die Parametrierbarkeit.

Mens und van Gorp [MG06] schlagen eine umfassende Taxonomie von Modelltransformationen vor, die in vielen Aspekten direkt auf die Klassifikation von Sichten übertragbar ist. Zunächst wird unterschieden, ob eine Transformation mehrere Quellen (engl. *inputs*) und Ziele (engl. *outputs*) hat. Weiter unterscheiden Mens und van Gorp, ob das Ziel dieselbe Sprache wie die Quelle hat (engl. *endogenous*) oder ob es sich bei der Transformation um eine Übersetzung handelt (engl. *exogenous*). Dieses Merkmal wird in der vorliegenden Dissertation durch die grundlegende Unterscheidung von allgemeinen Sichten und sprachübergreifenden Sichten berücksichtigt. Eine Unterscheidung in horizontale und vertikale Transformationen besagt, ob sich das Ziel auf demselben Abstraktionsgrad wie die Quelle befindet (horizontal) oder nicht (vertikal). Allerdings muss an dieser Stelle eine Fallunterscheidung bei vertikal erfolgen, die besagt ob der Detailgrad größer (stärkere Konkretisierung) oder kleiner wird (stärkere Abstraktion). Bzgl. der Automatisierung einer Transformation wird dahingehend unterschieden, ob Mehrdeutigkeiten automatisch aufgelöst werden oder einen interaktiven Eingriff erfordern. Unter dem Oberbegriff der Erhaltung (engl. *preservation*) sind Merkmale zur Erhaltung von syntaktischer und semantischer Korrektheit zusammengefasst. Neben den vorgestellten Merkmalen beschreiben Mens und van Gorp auch Merkmale der Implementierung, bspw. ob eine Transformationsfunktion mit syntaktisch inkorrekten Modellen umgehen kann.

3.5.2 Merkmalsausprägungen

Die diskutierten Merkmale zur Klassifikation bieten eine umfassende Grundlage, um Sichtenentwürfe zu klassifizieren. Das vorgeschlagene Modell zur Klassifikation enthält acht Merkmale mit jeweils zwei Ausprägungen, wobei die Ausprägung stets einem Prinzip folgt: Die erste Ausprägung beschreibt ob das Merkmal nach der Transformation *unverändert* bleibt. Die zweite Ausprägung beschreibt ob sich ein Merkmal durch die Transformationen mehr oder weniger *wahrscheinlich verändert*, bzw. dass eine Veränderung möglich ist. Diese beabsichtigte Unschärfe begründet sich darin, dass eine präzise Klassifikation abhängig ist vom Eingabeprozess, von angebundenen Datenquellen, von Parametern, von der Implementierung der Transformationsfunktionen und deren Ansteuerung durch Regeln. Mit diesen

Ausprägungen kann man ausdrücken, dass eine Eigenschaft, die als Vorbedingung erfüllt war, auch nach den durchgeführten Transformationen gilt. Andererseits kann man ausdrücken, dass eine Eigenschaft, die als Vorbedingung nicht erfüllt war, auch nach den Transformationen nicht gewährleistet werden kann, ausgedrückt durch die Einordnung *unverändert*. Zur Erläuterung dieses Prinzips folgen zwei Beispiele.

Beispiel 1. Gegeben sei ein Sichtenentwurf, der beschreibt, dass alle Aktivitäten, deren Eigenschaft *name* den Wert „Compliance“ enthalten, mit der Hintergrundfarbe Orange dargestellt werden sollen. Eine stärkere Abstraktion wird durch diese Transformation nicht bewirkt. Sofern diese Transformation allerdings auf einen Prozess angewendet wird, auf den bereits strukturreduzierende Muster angewendet wurden, so ist das Resultat dieser nachfolgenden Transformation im selben Maße abstrahiert.

Beispiel 2. Gegeben sei ein Sichtenentwurf, der beschreibt, dass alle Aktivitäten entfernt werden, deren Eigenschaft *name* den Wert „Prüfen“ haben. Diese Transformation bewirkt auf einen Eingabeprozess nur dann eine Veränderung, wenn dieser tatsächlich eine Aktivität mit diesem Namen enthält. Diese Eigenschaft kann jedoch erst geprüft werden, wenn der Eingabeprozess bekannt ist. Sollte der Eingabeprozess Strukturen enthalten, deren Name „Prüfen“ ist, werden diese entfernt und folglich wäre die Struktur des Prozesses nach der Transformation verändert.

3.5.3 Klassifikation

Die nachfolgende Aufstellung zeigt die acht grundlegenden Merkmale der Klassifikation von Sichtenentwürfen mit ihren Ausprägungen:

1. Struktur (unverändert | angepasst)
2. Darstellung (unverändert | angepasst)
3. Datenquellen (unverändert | angebunden)
4. Menscheneinbindung (unverändert | ausdrücklich)
5. Konkretisierung (unverändert | stärker)
6. Abstraktion (unverändert | stärker)
7. Syntaxänderung (unverändert | eventuell inkorrekt)
8. Ausführbarkeit (unverändert | nicht gewährleistet)

Die *Anpassung einer Struktur* liegt vor, wenn strukturverändernde Muster eingesetzt werden (M1-M12). Eine solche Anpassung hat verschiedene Folgen: Während der Transformation können Mehrdeutigkeiten entstehen, die entweder automatisch oder durch menschliches Eingreifen aufgelöst werden müssen. Des Weiteren wird durch eine Strukturanpassung u.U. die Rückführbarkeit von Änderungen eingeschränkt und ggf. ist eine Auflösung von Mehrdeutigkeiten erforderlich, die automatisch oder durch menschliche Einbindung erfolgen muss. Welche Umfänge an Änderungen ganz oder zumindest teilweise rückführbar sind, muss unter Berücksichtigung der Implementierung bestimmt werden.

Eine *Anpassung der Darstellung* liegt vor, wenn Muster der Kategorie Darstellungsanpassung eingesetzt werden (M16-M28).

Eine *Anbindung von Datenquellen* liegt vor, wenn bei einer Informationsanreicherung (M13-M15) oder beim Einfügen oder Zerlegen von Strukturen, Datenelementen oder Pfaden externe Datenquellen in die Transformation eingebunden werden (M9-M12).

Die *Einbindung von Menschen* kann ausdrücklich geschehen, wenn ein Sichtenentwurf eine nutzergestützte Anreicherung mit Informationen beinhaltet (M13).

Eine Sicht muss nicht notwendigerweise eine Abstraktion vornehmen, sie kann auch eine *stärkere Konkretisierung* beinhalten. Eine Sicht auf einen Prozess wird konkreter, wenn erweiternde strukturverändernde Muster eingesetzt werden (M9-M12), die in Zusammenhang mit Mustern zur Informationsanreicherung stehen (M13-M15).

Eine *stärkere Abstraktion* wird durch den Einsatz reduzierender strukturverändernder Muster bewirkt (M1-M8). Es besteht kein Widerspruch darin, dass eine Sicht auf einen Prozess diesen zugleich konkreter als auch abstrakter macht, sofern sich die Konkretisierung und die Abstraktion auf unterschiedliche Aspekte beziehen.

Durch den Einsatz strukturverändernder Muster wird die *Syntax eventuell inkorrekt*. Ob die Syntax eines transformierten Prozesses tatsächlich inkorrekt ist, hängt zum einen davon ab, ob Elemente des erweiterten Metamodells eingesetzt werden, die nicht den syntaktischen Vorgaben des Prozesses entsprechen. Durch Voranstellen der Vorgabemuster zum Bewahren der Konsistenz (M29) oder der Ausführbarkeit (M30) kann syntaktische Korrektheit durch Vorgabe bewahrt werden, sofern sie schon beim Eingangsprozess gegeben war. D.h. durch die Anwendung von einem dieser beiden Vorgabemuster ist die Ausprägung des Merkmals Syntaxänderung *unverändert*. Letztlich ist die syntaktische Korrektheit auch abhängig von der Implementierung der Muster.

Unter der *Ausführbarkeit* eines Prozesses werden in der vorliegenden Dissertation dessen syntaktische Korrektheit und die Freiheit von Phänomenen wie Deadlocks und ewigen Wartezuständen (engl. *eternal wait*) verstanden. Wenn strukturverändernde Muster (M1-M12) eingesetzt werden, kann diese Eigenschaft nicht gewährleistet werden. Damit eine korrekte Ausführbarkeit unverändert bleibt, ist syntaktische Korrektheit notwendig, aber nicht hinreichend. Dass durch strukturverändernde Transformationen, sei es zufällig oder beabsichtigt, die syntaktische Korrektheit oder die Ausführbarkeit eines Eingangsprozesses hergestellt werden könnte, wird in der vorgeschlagenen Klassifikation nicht berücksichtigt.

Die vorgeschlagene Klassifikation bezieht sich auf den vorgestellten Umfang des Musterkatalogs. Wird dieser Umfang erweitert, sollten zusätzliche Ausprägungen von Merkmalen oder zusätzliche Merkmale in Betracht gezogen werden. Im Rahmen der vorliegenden Dissertation wurde bspw. kein Muster zur aktiven Herstellung syntaktischer Korrektheit identifiziert. Sofern der Musterkatalog um ein solches Muster zum Korrigieren der Syntax ergänzt würde, könnte bei dem Merkmal *Syntaxänderung* die weitere Ausprägung *korrekt* unterschieden werden. Analog könnte dem Merkmal der *Ausführbarkeit* die Ausprägung *ausführbar* zugeordnet werden, sofern ein weiteres Muster die aktive Herstellung der Ausführbarkeit beschreiben würde. Wird der Umfang des Katalogs hingegen reduziert, können dann ggf. Merkmale entfallen.

Entsprechend der in einer Sicht eingesetzten Muster kann somit eine Klassifikation abgeleitet werden. In Abbildung 3.86 werden die beiden Extreme der Ausprägungen in Form eines Radar-Diagramms dargestellt. Links wird ein leerer Sichtenentwurf darstellt, in dem keine Transformationen vorgenommen werden. Rechts wird ein Entwurf dargestellt, in dem alle Merkmale verändert werden. Die Beschreibung der Ausprägung ändert sich entsprechend, bspw. ist links die *Syntax unverändert*, wohingegen rechts die *Syntax eventuell inkorrekt* ist.

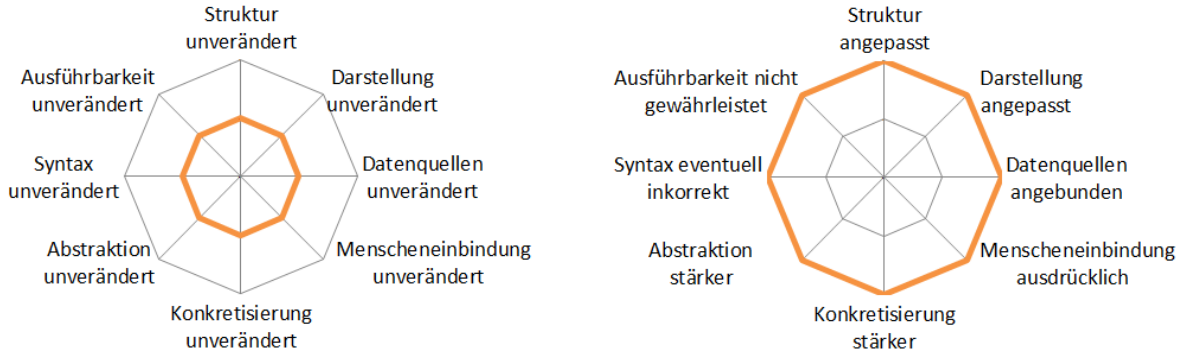


Abbildung 3.86: Unveränderte Merkmale (links) und maximal veränderte Merkmale (rechts).

Die vorgestellte Klassifikation erlaubt es, einen Sichtenentwurf auf Basis der darin enthaltenen Muster zu klassifizieren und auf die Eigenschaften einer Komposition zu schließen.

3.6 Musterbasierter Sichtenentwurf

Die mehrstufige Vorgehensweise zum musterbasierten Sichtenentwurf (vgl. Abschnitt 3.2) sieht vor, einen Sichtenentwurf zunächst grob zu spezifizieren, diesen Grobentwurf zu verfeinern und anschließend an technische Gegebenheiten anzupassen und zu implementieren, um diese Implementierung fortan zur Erzeugung von Sichten nutzen zu können. In diesem Abschnitt werden zuerst grundlegende Richtlinien zum Entwurf von Sichten vorgestellt und danach wird der grobe und feine Entwurf einer Sicht auf Basis von Sichtenentwurfsmustern diskutiert. Anschließend werden komplexitätsreduzierende Konzepte vorgeschlagen, die den Sichtenentwurf vereinfachen sollen.

3.6.1 Richtlinien zum Sichtenentwurf

Der musterbasierte Sichtenentwurf stellt eine Vorstufe der Implementierung von Sichtenfunktionalität dar und dient als abstrakte Beschreibung der anzuwendenden Transformationen. Richtlinien sollen die Tätigkeit des Entwerfens vereinfachen und dienen als grobe Orientierung und Strukturierung für die Entwurfstätigkeit. Wie auch mit anderen Entwurfstechniken ist es allerdings möglich, sinnlose oder fehlerhafte Entwürfe zu erstellen.

Richtlinie 1: Sequenzbildung.

Bobrik [Bob08] stellt fest, dass zwei Transformationen zur Erzeugung einer Sicht kommutativ sind und damit deren Reihenfolge ohne Auswirkung auf generierte Sichten geändert werden kann, wenn deren Anwendungsbereiche sich nicht überlappen. Zwei Musteran-

wendungen können *nicht kommutativ* sein und sich deren Reihenfolge dadurch auf die generierte Sicht auswirken, wenn sie ganz oder teilweise dieselben Aktivitäten, Datenelemente, Kontrollverbindungen oder deren Eigenschaften betreffen. Daher wird empfohlen, bei der Zusammenstellung von Musteranwendungen zu einem Sichtenentwurf eine Festlegung der Reihenfolge der Musteranwendungen in einer sogenannten *Mustersequenz* vorzunehmen. Diese Mustersequenz, zeilenweise gelesen von links nach rechts, berücksichtigt bestehende, notwendigerweise zyklenfreie Abhängigkeiten zwischen den einzelnen Musteranwendungen. Als Hilfestellung bei der Bildung einer Mustersequenz kann eine Darstellung der Abhängigkeiten zwischen den Musteranwendungen in einem Mustergraphen herangezogen werden. Ein Beispiel eines Mustergraphen wird in Abschnitt 3.6.2.2 vorgestellt.

Richtlinie 2: Konfliktprüfung.

Durch die Anwendung verschiedener Transformationen auf einen überlappenden Anwendungsbereich können verschiedene Arten von Konflikten auftreten, wie bspw. ein Verlust überschriebener Eigenschaftswerte (engl. *lost update*). Ein Überschreiben kann allerdings auch beabsichtigt sein. Konflikte sollten beim Entwurf einer Sicht geprüft und aufgelöst werden. Zwischen Musteranwendungen können verschiedene Arten von Konflikten bestehen:

- Zwischen Musteranwendungen kann ein Interessenskonflikt bestehen. Bspw. kann eine Musteranwendung Informationen hinzufügen, die eine andere entfernen soll.
- Werden in einer Musteranwendung Informationen entfernt, aggregiert oder modifiziert, die bei einer anderen Musteranwendung in unveränderter Form erforderlich sind, besteht ein Abhängigkeitskonflikt.
- Ein Konflikt in der Informationscodierung kann vorliegen, wenn eine durch eine Musteranwendung codierte Information durch eine andere Musteranwendung überschrieben wird. Bspw. kann eine Musteranwendung die Hintergrundfarbe einer Aktivität für den Ausführungszustand verwenden, während die Hintergrundfarbe in einer anderen Musteranwendung zur Hervorhebung dient. Eine der Informationen muss demzufolge auf eine andere Weise codiert werden.

Richtlinie 3: Gruppierung als Mittel der Modularisierung.

Nach Definition 8 fasst eine Gruppierung ein Segment einer Mustersequenz zusammen. Um die Komplexität des Entwurfs zu reduzieren, können bspw. mehrere Musteranwendungen zur Informationsanreicherung (M13-M15) zu einer Gruppierung zusammengefasst und als wiederverwendbares Modul behandelt werden.

Richtlinie 4: Vorgabenverwendung und deren Gültigkeitsbereich.

Die Anwendung von Vorgaben (M29-M33) gilt für alle nachfolgenden Musteranwendungen. Gruppierung können nach Definition 8 dazu verwendet werden, den Gültigkeitsbereich von Vorgaben einzuschränken. Wird eine Vorgabe innerhalb einer Gruppierung verwendet, ist der Gültigkeitsbereich auf die nachfolgenden Musteranwendungen innerhalb der Gruppierung beschränkt. Bspw. kann eine Gruppierung reduzierende Strukturveränderungen umfassen (M1-M8), bei denen der Prozess zusammengehalten werden soll (M29). Durch diese Richtlinie ist die Zusammensetzung von Musteranwendungen *nicht assoziativ*.

Richtlinie 5: Reihenfolgenbeachtung.

Eine häufig passende, aber nicht zwingend notwendige Reihenfolge der Musteranwendungen besteht darin,

1. zunächst allgemeine Vorgaben zum Verhalten der anzuwendenden Transformationen festzulegen (M29-M33),
2. dann eine Informationsanreicherung vorzunehmen (M13-M15),
3. anschließend die Struktur zu verändern (M1-M12)
4. und abschließend die Darstellung des transformierten Eingabeprozesses festzulegen (M16-M28).

Dieser Richtlinie liegt folgende Argumentation zugrunde:

- Vorgaben beziehen sich i.d.R. auf alle Musteranwendungen und stehen daher am Anfang der Mustersequenz.
- Musteranwendungen zur Strukturveränderungen können angereicherte Informationen einbeziehen, sofern die Informationsanreicherung vorher geschieht.
- Musteranwendungen zur Anpassung des Erscheinungsbilds können angereicherte Informationen einbeziehen, sofern die Informationsanreicherung vorher geschieht.
- Die Bestimmung des Layouts eines Prozesses erfordert, dass der Prozess nicht nachträglich im Abstraktionsgrad angepasst wird. Musteranwendungen zur Festlegung der Darstellung stehen daher i.d.R. am Ende einer Mustersequenz.

3.6.2 Vom Grobentwurf zum Feinentwurf

Der Übergang vom Grobentwurf zum Feinentwurf stellt nicht nur eine Anforderungsanalyse zur Vorbereitung der Implementierung dar. Der Übergang erfüllt auch den Zweck, Fehler im Entwurf zu korrigieren, die technische Machbarkeit festzustellen, Kosten einer Umsetzung abzuschätzen und mögliche Entwurfs- und Umsetzungsalternativen abzuwägen.

3.6.2.1 Grobentwurf

Ein grober Sichtenentwurf bietet zu einer bestimmten Problemstellung einen Lösungsansatz in Form einer Mustersequenz, der eine textuelle Beschreibung und eine grafische Skizze sowie Beispiele beigefügt sind. Eine Mustersequenz listet die auf einen Eingabeprozess anzuwendenden Transformationen auf und gibt Informationen über deren Anwendungsbereich. Die Klassifikation eines Sichtenentwurfs kann aus den verwendeten Mustern abgeleitet werden. Die Mustersequenz und folglich auch die Klassifikation können sich allerdings im Zuge der Verfeinerung und Implementierung ändern und daher ist eine Zusicherung der klassifizierenden Eigenschaften im Stadium des Grobentwurfs einer Sicht nur eingeschränkt belastbar.

Abbildung 3.87 zeigt die Mustersequenz eines beispielhaften Grobentwurfs. Der Mustersequenz zufolge soll ein Eingabeprozess zunächst mit Laufzeitinformationen angereichert werden (M15), anschließend sollen nicht relevante Aktivitäten entfernt werden (M1), ab-

schließend sollen Aktivitäten hervorgehoben werden, die den Kunden betreffen (M16). Die dargestellte Skizze lässt einen gewissen Freiraum zur Interpretation des Entwurfs.

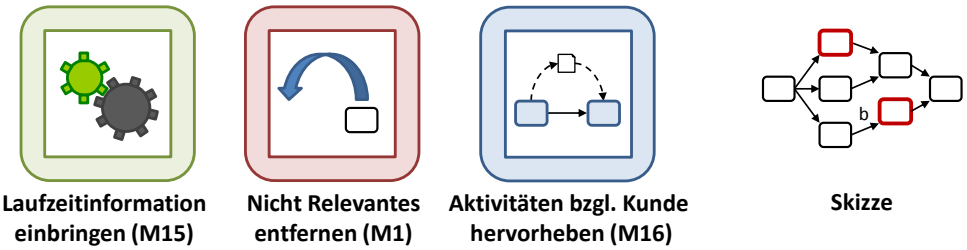


Abbildung 3.87: Beispiel der Mustersequenz eines Grobentwurfs mit einer einfachen Skizze.

3.6.2.2 Feinentwurf

Im Übergang vom Grobentwurf zum Feinentwurf werden unkonkrete Anforderungen geschärft und der Sichtenentwurf präzisiert. Konflikte und Widersprüche werden entfernt und ggf. werden komplizierte Musterkonstrukte durch einfachere oder elegantere ersetzt. Nicht erfüllbare Anforderungen werden entfernt und der Sichtenentwurf wird entsprechend überarbeitet oder neu konzipiert. Für die Erstellung eines Feinentwurfs sind *drei Aufgaben* maßgeblich, die nachfolgend erläutert werden:

- Anforderungsanalyse
- Verfeinerung der Mustersequenz
- Präzisierung der verfeinerten Mustersequenz

Anforderungsanalyse. Der Übergang vom Grobentwurf zum Feinentwurf ist an den organisatorischen und technischen Gegebenheiten eines Unternehmens ausgerichtet. Dies beinhaltet die geplante Nutzung der Sicht, was mit Anwendungsfällen (engl. *use cases*) in der Softwareentwicklung vergleichbar ist. Außerdem müssen die konkreten Prozessmodelle betrachtet werden, auf die die Sicht angewendet werden soll. Als Konsequenz erfordert der Übergang vom Grobentwurf zum Feinentwurf eine Abstimmung der Anforderungen der Interessengruppen mit dem IT-Bereich, der die Sicht als technische Lösung spezifiziert und diese nach der Implementierung bereitstellt.

Die Verfeinerung umfasst die Erfassung, Spezifikation und Abstimmung der Anforderungen an eine Sicht. Diese Anforderungserhebung und -analyse betrifft verschiedene Aspekte:

- Zweck: Was wird mit der Sicht bezweckt, welche Fragen sollen beantwortet werden, welche Aufgaben und welche Phasen des Prozesslebenszyklus sollen unterstützt werden? Sollen in der Sicht bestimmte Interaktionen oder Änderungen möglich sein und sollen sich diese auf den Ursprungsprozess auswirken (vgl. Rückführbarkeit)? Soll eine Kopie des Prozesses erstellt werden (vgl. Durability)?
- Nutzung: Wie oft wird diese Sicht benötigt (z.B. einmalig, regelmäßig)? Soll ein Nutzer in die Erzeugung der Sicht interaktiv einbezogen werden, etwa um Mehrdeutigkeiten aufzulösen oder um die Sicht über Parameter zu konfigurieren?

- Parameter: Welche Aspekte der Sicht sollen konstant und welche dynamisch sein?
- Eingabeprozesse: Ist die Sicht spezifisch für einen bestimmten Prozess oder soll die Sicht auf mehrere Prozesse anwendbar sein? Welche Beschaffenheit haben diese Prozesse bzgl. Format, Prozesssprache, Konsistenz und Automatisierungsgrad?
- Vorbedingungen: Müssen semantische Annotationen im Vorfeld erfolgen (M13) oder ist eine andere Anpassung der zu unterstützenden Prozesse erforderlich, damit eine Sicht aus der Perspektive des Nutzers konsistent und korrekt ist?
- Datenquellen: Welche nicht im Prozessmodell enthaltenen Daten werden benötigt und in welchen Datenquellen befinden sich diese Daten?
- Musteranwendungen: Zu jeder Musteranwendung muss der Anwendungsbereich und die Funktionsweise der Transformation spezifiziert werden. Soll die Anwendung nach „All or Nothing“ (vgl. Atomicity) erfolgen?

Verfeinerung der Mustersequenz. Die Anforderungsanalyse liefert die zur Verfeinerung einer Mustersequenz erforderlichen Informationen. Eine Verfeinerung der beispielhaften Mustersequenz aus Abbildung 3.87 ist in Abbildung 3.88 dargestellt. Sie enthält nun die Vorgabe, dass der Prozess bei den Transformationen nicht in einzelne unverbundene Prozessfragmente zerfallen, sondern zusammengehalten werden soll (M29). Die Verfeinerung zeigt auch, dass eine Anreicherung mit Aktivitätszuständen ausreichend ist (M15), während der Grobentwurf die unpräzise Beschreibung „Laufzeitinformationen“ erwähnt. Unter der in der Anforderungsanalyse gewonnenen Kenntnis, welche Prozesse unterstützt werden sollen, kann die entsprechende Datenquelle für die Aktivitätszustände identifiziert werden. „Nicht Relevantes“ im Grobentwurf ist im Feinentwurf auf ausstehende und übersprungene Aktivitäten eingegrenzt worden (M1).

Für die Hervorhebung wurde das Vorgehen abgestimmt, alle Aktivitäten auszugrauen (M16), ausgewählte Aktivitäten hervorzuheben (M16) und den Aktivitätszustand darzustellen. Das Auswahlkriterium hervorzuhebender Aktivitäten konnte nach Analyse der zu unterstützenden Prozesse auf den Zugriff auf „customerData“ festgelegt werden.

Eine Anforderung, dass der vereinfachte Prozess als Graph dargestellt werden soll, war zuvor nur in der Skizze impliziert und wurde nun explizit durch eine Musteranwendung festgelegt (M19). Die verfeinerte Skizze, wie in Abbildung 3.88 (a-e) beispielhaft gezeigt, soll die Kommunikation über die prognostizierten Effekte der Musteranwendungen unterstützen.

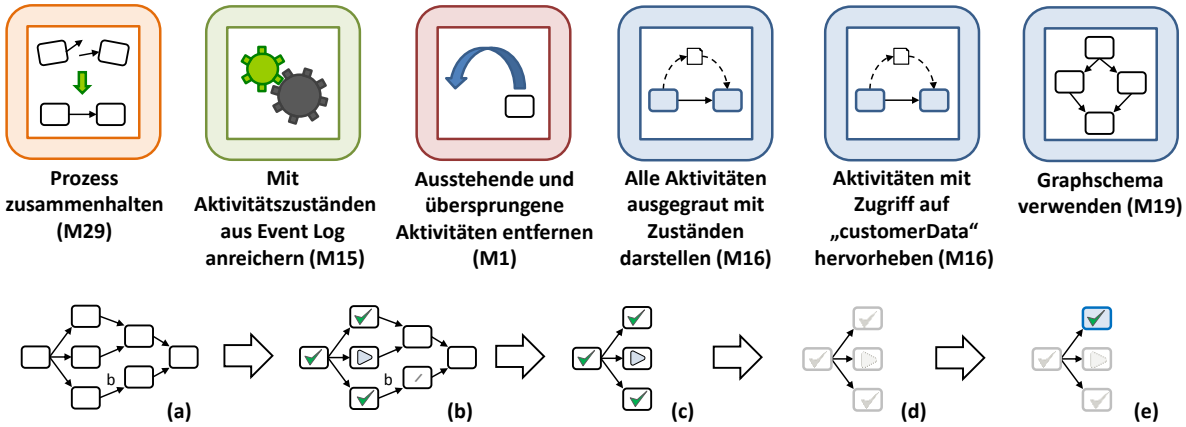


Abbildung 3.88: Beispiel der Mustersequenz eines verfeinerten Grobentwurfs mit detaillierter Skizze.

Die verfeinerte Mustersequenz in Abbildung 3.88 berücksichtigt die Abhängigkeiten, die zwischen einzelnen Musteranwendungen bestehen und die mögliche Abfolge einschränken:

- Alle Musteranwendungen sollen die Vorgabe zum Zusammenhalten des Prozesses (M29) berücksichtigen.
- Das Entfernen ausstehender und übersprungener Aktivitäten (M1) benötigt Informationen über Aktivitätszustände (M15).
- Nur die nach dem Entfernen (M1) im Prozess verbleibenden Aktivitäten sollen dargestellt werden (M16).
- Die Darstellung von Aktivitäten mit Aktivitätszuständen (M16) benötigt Informationen über die Aktivitätszustände (M15).
- Die Darstellung im Graphschema (M19) soll auf Basis des reduzierten Prozessgraphen stattfinden und ist daher dem Entfernen von Aktivitäten (M1) nachgelagert.
- Bei der Reihenfolge von Musteranwendungen zur Anpassung des Erscheinungsbilds (M16-M18) kann ein kaskadenartiges Verhalten[4] angenommen werden, wodurch eine nachfolgende Musteranwendung (hier: Hervorhebung) die Wirkung einer vorangehenden Anwendung (hier: Ausgrauen) überschreibt. Das Ausgrauen (M16) erfolgt daher vor dem Hervorheben (M16). Für die tatsächliche Implementierung der Sicht ist lediglich der im Entwurf angeforderte Effekt relevant.

Wie in der ersten Richtlinie zum Sichtenentwurf (Sequenzbildung) angemerkt, kann eine Darstellung der Abhängigkeiten in Form eines Mustergraphen als Hilfsmittel zur Bildung einer Mustersequenz herangezogen werden. Abbildung 3.89 stellt die beschriebenen Abhängigkeiten zwischen den Musteranwendungen aus dem vorangehenden Beispiel als Mustergraph, d.h. als Kontrollflussgraph nach dem Verständnis eines PM-Graphen dar (vgl. Definition 1), wobei eine Kontrollverbindung für die Abhängigkeit von zwei Musteranwendungen steht. Die Nummerierung der Mustersymbole und die in Blau hervorgehobenen

[4] Vergleichbar zur Cascading Style Sheets Technologie (CSS) [W3C13]

Kontrollverbindungen zeigen eine unter Beachtung der Abhängigkeiten mögliche Reihenfolge der Musteranwendungen auf, die der in Abbildung 3.88 dargestellten Mustersequenz entspricht. Da zwischen der Darstellung im Graphschema (M19) und der Darstellung von Aktivitäten mit Aktivitätszuständen (M16) keine Abhängigkeit besteht, gibt es in diesem Beispiel drei Möglichkeiten zur Bildung der Mustersequenz. Eine davon wurde in Abbildung 3.89 ausgewählt und hervorgehoben. Sie erhält den direkten Zusammenhang zwischen den beiden Musteranwendungen bzgl. des Erscheinungsbilds von Aktivitäten (M16).

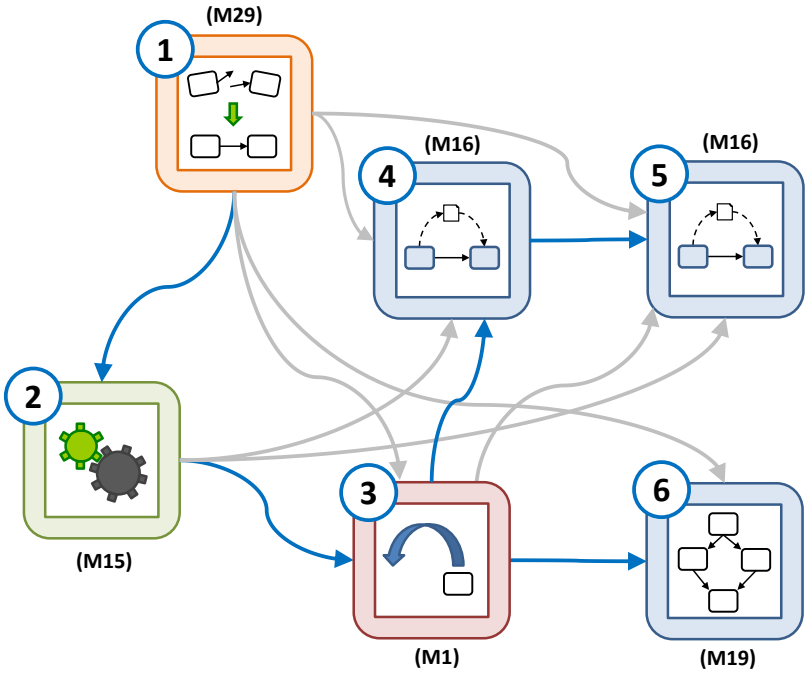


Abbildung 3.89: Mustergraph zur Darstellung der Abhängigkeiten der Musteranwendungen eines musterbasierten Sichtenentwurfs.

Die Darstellung der Abhängigkeiten von Musteranwendungen in einem Kontrollflussgraphen hat mehr Ausdruckskraft als die auf eine mögliche Reihenfolge reduzierte Schreibweise in einer Mustersequenz. Diese Ausdruckskraft macht einen musterbasierten Sichtenentwurf einerseits vollständiger, andererseits komplexer. Die in der vorliegenden Dissertation vorgestellten Sichtenentwürfe werden anhand einer Mustersequenz beschrieben, welche die Abhängigkeiten der einzelnen Musteranwendungen berücksichtigt, die damit verbundene Komplexität durch die einfache Schreibweise aber verbirgt.

Präzisierung der verfeinerten Mustersequenz. Um einen Sichtenentwurf weiter zu präzisieren, empfiehlt sich eine Überführung der Mustersequenz in eine tabellarische Form. Die Präzisierung stellt eine Vorstufe zur Implementierung einer Sicht dar und ist von den Möglichkeiten verfügbarer Transformationsfunktionen nicht gänzlich zu trennen. Dieser Umstand wird im Rahmen der dienstbasierten Realisierung von Sichtenentwürfen in Abschnitt 9.1 vertieft und anhand domänenspezifischer Sprachen zur Parametrierung von Transformationsfunktionen und zur grafischen Konfiguration veranschaulicht.

Tabelle 3.2 zeigt die präzisierte Mustersequenz für das vorangehende Beispiel.

Tabelle 3.2: Tabellarische Form einer präzisierten verfeinerten Mustersequenz eines Sichtenentwurfs.

Nr.	Symbol	Beschreibung	Präzisierung
M29		Prozess zusammenhalten	Konsistenz: Kontrollabhängigkeiten erhalten
M15		Mit Aktivitäts-zuständen anreichern	Instanz ID: Auswahl durch Nutzer Datenquelle: Event Log
M1		Ausstehende und über-sprungene Aktivitäten entfernen	Anwendungsbereich: .zustand = (inaktiv | übersprungen) Fehlerfall: ignorieren
M16		Alle Aktivitäten ausgrauen	Anwendungsbereich: * RGB-Farbcode: 200/200/200
M16		Aktivitäten mit Zugriff auf „customerData“ hervorheben	Anwendungsbereich: .eingabe = customerData | .ausgabe = customerData RGB-Farbcode: 250/200/0
M19		Graphschema verwenden	Orientierung: default

3.6.3 Komplexitätsreduzierende Konzepte im Sichtenentwurf

Die Komplexität des Sichtenentwurfs kann durch verschiedene Konzepte reduziert werden. Dazu gehören Modularisierung, Wiederverwendung und der Einsatz von allgemeinen und werkzeugspezifischen Entwurfsvorlagen.

Modularisierung. Thematisch zusammengehörige Segmente einer Mustersequenz können durch eine Gruppierung zusammengefasst werden, vgl. Definition 8. Wie in Abbildung 3.90 beispielhaft dargestellt, kann ein Segment einer Mustersequenz gruppiert, von dem sichten-entwerfenden Nutzer mit einem passenden Namen versehen und diese Gruppierung aggre-giert dargestellt werden.

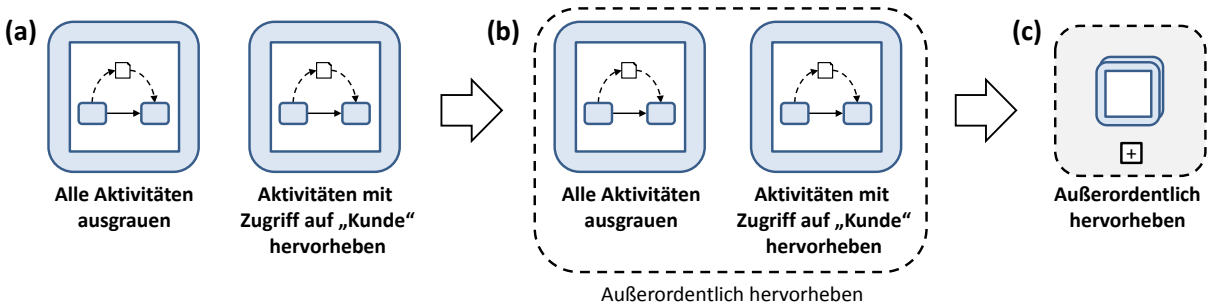


Abbildung 3.90: Gruppierung (b) und Aggregation (c) eines Segments (a) zur Komplexitätsreduzierung.

Wiederverwendung. Neben der Verbesserung der Übersicht kann Modularisierung auch zur Wiederverwendung beitragen. Abbildung 3.91 (a) zeigt die Gruppierung aus dem vorigen Beispiel. Das Hervorheben von Aktivitäten ist in dem Beispiel auf ein bestimmtes Merkmal konkretisiert: der Zugriff auf „Kunde“. Durch eine Parametrisierung (b) wird dieses Merkmal abstrahiert. Das entstandene Modul kann als Baustein in weiteren Entwürfen genutzt werden, indem es an geeigneter Stelle in eine Mustersequenz eingefügt wird und der Parameter `<Merkmal>` passend gesetzt wird. Derartige Module können zu einer Sammlung bewährter Vorgehensweisen im Sichtenentwurf beitragen und damit wiederum Muster bilden, die im Vergleich zu den Sichtenentwurfsmustern komplexer und zugleich konkreter sind.

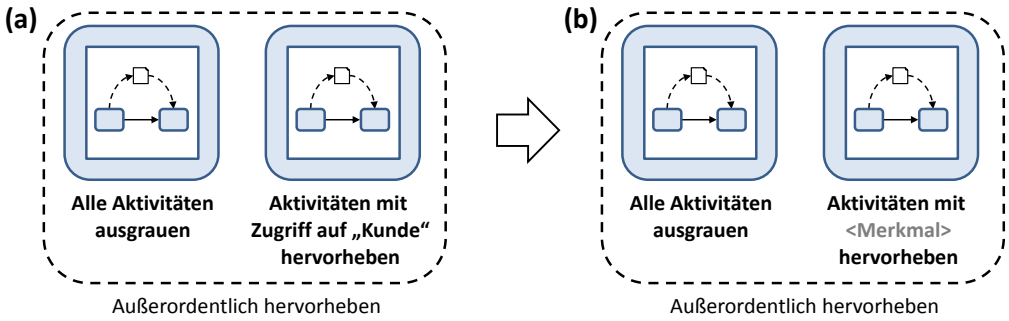


Abbildung 3.91: Parametrisierung eines Mustersegments zur Unterstützung der Wiederverwendung.

Entwurfsvorlagen. Neben modularen Bausteinen können auch Vorlagen zur Wiederverwendung beitragen und den Entwurf unterstützen. Anstatt einen Entwurf von Grund auf neu zu entwickeln, kann eine Entwurfsvorlage eine Grundlage darstellen, in der bewährte oder in einem Unternehmen gängige Anforderungen und Normen bereits enthalten sind. Wie in Abbildung 3.92 skizziert, kann eine Entwurfsvorlage Platzhalter enthalten, um Stellen zur Verfeinerung des Entwurfs explizit zu kennzeichnen. Ein Platzhalter kann im Rahmen der Verwendung einer Vorlage durch eine oder mehrere Musteranwendungen ersetzt, oder auch ungenutzt entfernt werden.

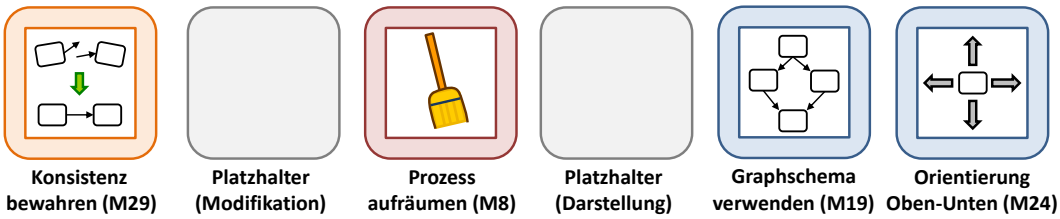


Abbildung 3.92: Beispiel einer Vorlage als Mittel der Vereinfachung und Standardisierung des Entwurfs.

Werkzeugspezifische Entwurfsvorlagen. Der Einsatz von Vorlagen im Sichtenentwurf kann neben der Vereinfachung, Beschleunigung und Vereinheitlichung des Entwurfs auch die Implementierung vereinfachen. Dies erfordert, dass die in Vorlagen enthaltenen Platzhalter dahingehend eingeschränkt werden, dass nur Muster darin verwendet werden können, die von einer vorhandenen Werkzeugkette unterstützt werden. Muster können durchaus auch mehrfach angewendet werden.

Abbildung 3.93 illustriert die Constraints der Platzhalter aus der vorangehenden Entwurfsvorlage. In der Entwurfsvorlage wurde der Platzhalter zur Modifikation zweigeteilt: In dem Platzhalter zur Anreicherung ist nur die ggf. mehrfach angewendete nutzergestützte Anreicherung zulässig (M13), welche bspw. in einem verfügbaren Modellierungswerkzeug implementiert werden kann. In dem Platzhalter für Strukturveränderung können beliebige Kombinationen der Muster Entfernen von Aktivitäten (M1), Aggregation verbundener Aktivitäten (M2) und Eigenschaften abstrahieren (M4) verwendet werden, welche z.B. durch einen bestehenden Web Service realisiert werden können. Sofern eine bestimmte Reihenfolge festgelegt werden soll, können die entsprechenden Muster in einer Gruppierung als fest vorgegebene Sequenz zusammengefasst werden. Werkzeuge, die zur Darstellung von Prozessen verwendet werden, haben normalerweise eine Reihe grafischer Funktionen, die die Darstellung eines Prozesses weitgehend festlegen. Ein Teil dieser Funktionen kann ggf. überschrieben werden, wie etwa durch benutzerdefinierte Aktivitätsvorlagen. Der Platzhalter der Darstellung ist hier beispielhaft eingeschränkt, sodass nur das Erscheinungsbild von Aktivitäten (M16) angepasst werden kann.

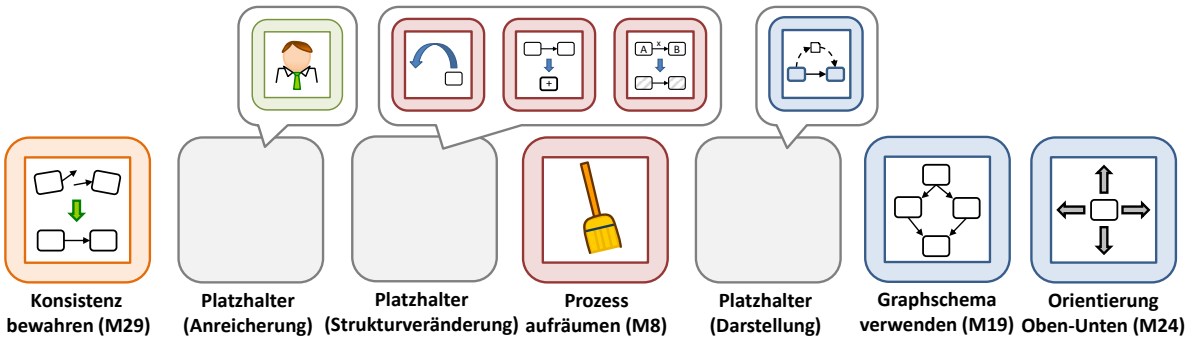


Abbildung 3.93: Beispiel einer Entwurfsvorlage mit werkzeugspezifisch eingeschränkten Platzhaltern.

3.7 Informationsdesign im Lebenszyklus von Geschäftsprozessen

Die in diesem Kapitel entwickelte Methode des musterbasierten Sichtenentwurfs erlaubt die Formulierung von Informationsbedarfen einer Prozesssicht. Durch diesen Ansatz entsteht die Aufgabe, Informationsbedarfe verschiedener Interessengruppen mit Möglichkeiten zur Realisierung der geforderten Sichtenfunktionalität abzugleichen und die Überführung der abgestimmten Anforderungen in eine technische Implementierung zu begleiten. Diese Aufgabe wird fortan als *Informationsdesign für Geschäftsprozesse* (engl. *Information Design for Business Process Management*) bezeichnet [Sch11b].

Informationsdesign. Die Aufgabe des Informationsdesigns ist thematisch im Umfeld der geschäftsnahen IT angesiedelt, d.h. in der Prozessberatung und -automatisierung, in der IT-Architektur und im Consulting mit Prozessbezug.

Abbildung 3.94 erweitert die Vorgehensweise des musterbasierten Entwurfs und der dienstorientierten Implementierung von Sichten auf Geschäftsprozesse um die Aufgabe des Informationsdesigns, vgl. Abbildung 3.1 in Abschnitt 3.2.

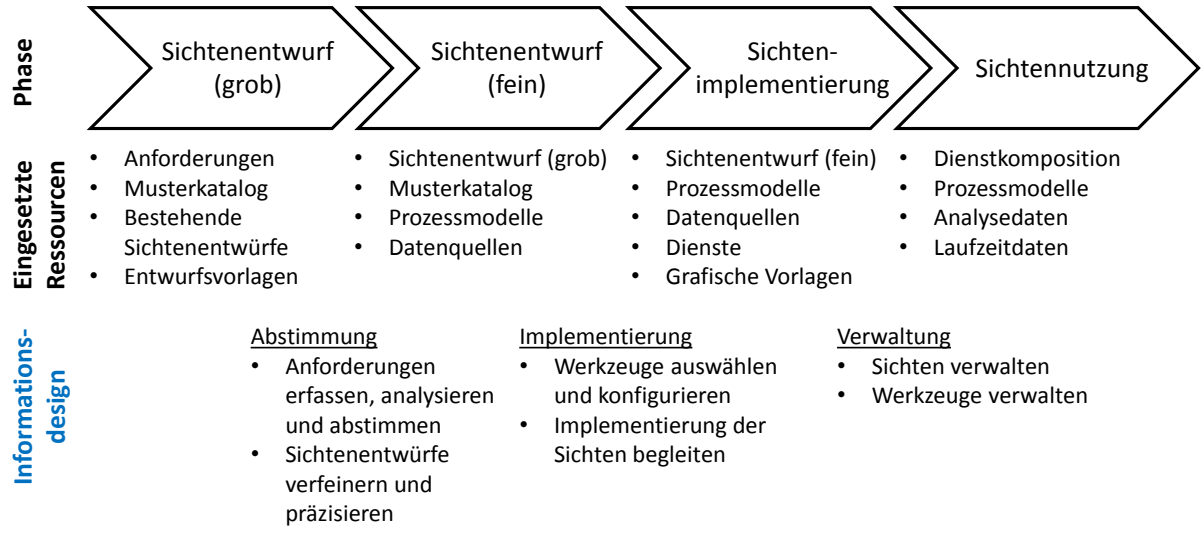


Abbildung 3.94: Informationsdesign für Geschäftsprozesse als Schnittstelle zwischen Anforderungen und der Bereitstellung von Informationen im Prozesszusammenhang.

Das Informationsdesign erfüllt drei elementare Funktionen, durch die es eine Schnittstelle zwischen Anforderungen und der Bereitstellung von Informationen im Prozesszusammenhang bildet: (1) die Abstimmung von Anforderungen, (2) die Begleitung der Implementierung und (3) eine übergreifende Verwaltung:

1. Abstimmung: Anforderungen an eine Sicht werden konkretisiert und an technische und organisatorische Rahmenbedingungen angepasst. Dies umfasst eine Prüfung, ob die gewünschte Sicht mit den Rollen und Rechten der anfordernden Interessengruppe realisierbar ist und mit unternehmensinternen Richtlinien, Betriebsvereinbarungen und gesetzlichen Vorgaben vereinbar ist. Man kann diesen Schritt als Bestandteil einer Compliance-Prüfung der umzusetzenden Sicht betrachten.

2. Implementierung: Es werden verschiedene Realisierungsmöglichkeiten der geforderten Sichtenfunktionalität geprüft und gegenübergestellt. In der dienstorientierten Implementierung umfasst diese Funktion die Auswahl und Bereitstellung von Transformationsdiensten sowie deren Komposition und Konfiguration, damit diese komponierten Dienste zur Erzeugung von Sichten genutzt werden können.

3. Verwaltung: Die Verwaltung (engl. *governance*) umfasst die für Sichten relevanten Umfänge: Anforderungen, Prozesse, Datenquellen, Schnittstellen (API), Sichtenentwurfsmuster, Musterkataloge, Sichtenentwürfe, Entwurfsvorlagen, Werkzeuge, Implementierungen und Nutzer. Des Weiteren ist die Definition und Pflege semantischer Richtlinien für die Prozessannotation essentiell. Die Verwaltung umfasst zudem die Wartung von Sichten bei Prozessänderungen (engl. *change management*).

Das Informationsdesign wird als ein Bindeglied zwischen den Lebenszyklusphasen betrachtet, vgl. Abbildung 3.95. Es verarbeitet Anforderungen aus der Prozessmodellierung, Monitoring und Analyse und stimmt gegebene Umsetzungsmöglichkeiten mit zuständigen Bereichen für die Prozessimplementierung und Prozessausführung ab. Konkrete Teilaufgaben

können durchaus auf andere Rollen verteilt werden. Experten für Grafikdesign können bspw. involviert werden, wie Abbildung 3.95 zeigt. Außerdem wird die Aufgabe der Implementierung um sichtenbezogene Implementierungsumfänge erweitert.

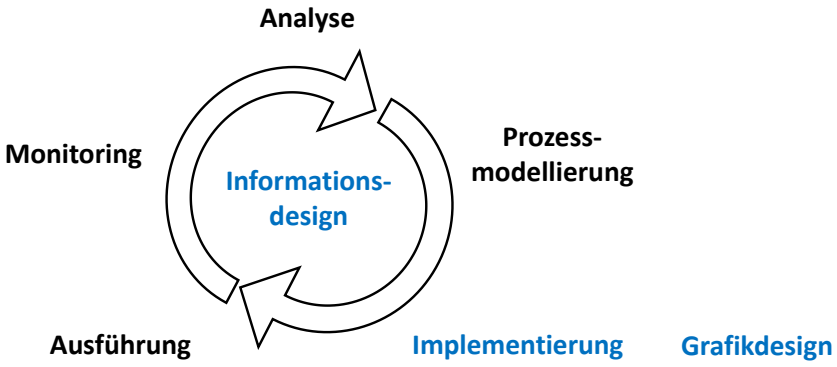


Abbildung 3.95: Informationsdesign im Lebenszyklus von Geschäftsprozessen.

KAPITEL 4

BEISPIELHAFTE SICHTENENTWÜRFE FÜR DEN PROZESSLEBENSZYKLUS

In diesem Kapitel werden beispielhafte Sichten zur Unterstützung der Phasen des Prozesslebenszyklus entworfen. Die Sichtenentwürfe betreffen gängige prozessbezogene Informationsbedarfe, die durch die Anwendung und Komposition der identifizierten Sichtenentwurfsmuster unterstützt werden. Zwar gibt es bereits Beiträge und Werkzeuge, die ähnliche Sichten oder Kerngedanken daraus diskutieren oder implementieren [SAL+10b], allerdings verwenden diese Arbeiten die Muster *implizit*. Das Innovative der Sichtenentwürfe besteht im Einsatz der Mustersprache und dem damit verbundenen hohen Grad an Abstraktion, der eine einfache Beschreibung und Vergleichbarkeit ermöglicht und als Ausgangspunkt zur Verfeinerung und Implementierung herangezogen werden kann. Die Sichtenentwürfe sind grob gehalten, beziehen sich nicht auf konkrete Prozesse und nehmen keinen Bezug auf konkrete Werkzeuge für deren Realisierung. Die Entwürfe sind dadurch auf das Wesentliche reduziert.

4.1 Vielseitig verwendbare Sichten

Vielseitig verwendbare Sichten sind keiner Lebenszyklusphase exklusiv zugeordnet, sondern können in der Modellierung, im Monitoring und in der Analyse verwendet werden.

4.1.1 Entfernen von Details

Problem: Das Entfernen von irrelevanten Details ist seit jeher ein Mittel zur Abstraktion. Aber wie kann das Wichtige vom Unwichtigen unterschieden werden?

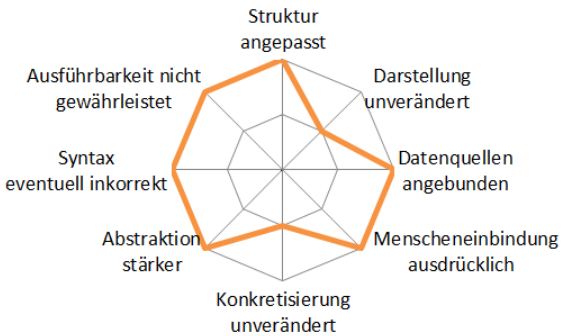


Abbildung 4.1: Klassifikation des Entwurfs.

Lösungsansatz: Wichtigkeit hängt von der Rolle und Aufgabenstellung des Betrachters ab, der sich mit einem prozessbezogenen Sachverhalt mit einem definierten Zeitbezug befasst. Es werden alle gängigen Mittel eingesetzt, um zunächst die in einem Prozess enthaltenen Elemente nach verschiedenen Gesichtspunkten zu bewerten, vgl. die Mustersequenz in Abbildung 4.2 (a). Neben Know-how über den Prozess (M13) stehen evtl. auch Informationen über Ausführungen zur Verfügung (M15), die bei der Bewertung von Relevanzen eingesetzt werden

können. Weitere Datenquellen können Hintergrundinformationen zu Aktivitäten und Datenelementen beitragen (M15). Auf dieser Informationsgrundlage weist eine Analysefunktion den Prozesselementen verschiedene Relevanzwerte zu, die sich von Angelegenheit zu Angelegenheit unterscheiden können. Die Festlegung der Relevanzfunktion und die Konkretisierung, welche Informationen in diese Funktion hineinspielen, werden als Bestandteil des Feinentwurfs betrachtet. Diese Mustersequenz kann in unterschiedlichen Szenarien als Baustein wiederverwendet werden, vgl. Abbildung 4.2 (b).

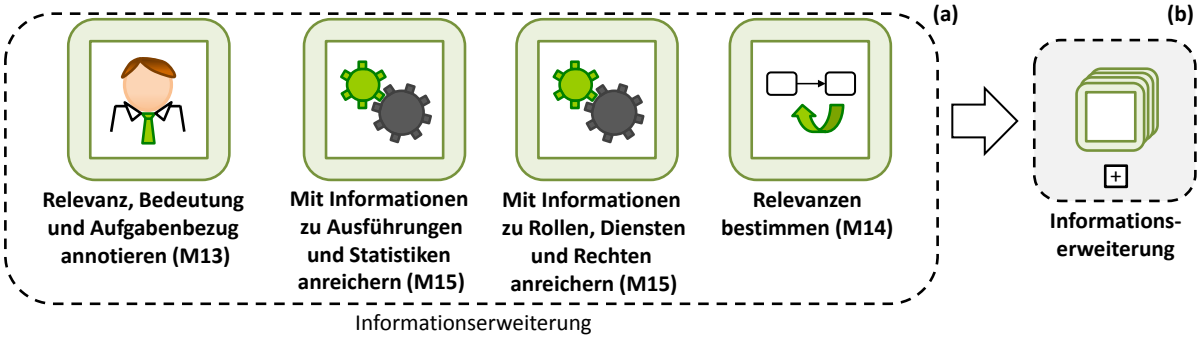


Abbildung 4.2: Mustergruppierung zur Informationserweiterung.

Nachdem die Relevanz der einzelnen Elemente bekannt ist, wird das Wichtige bewahrt und alles andere entfernt, wie Abbildung 4.3 zeigt. Beim Entfernen soll die syntaktische Korrektheit erhalten bleiben und der Prozess dadurch zusammengehalten werden (M29). Die Vorgabe, Rollen- und Aufgabenbezogenes zu bewahren (M31) dient dazu, relevante Aktivitäten und Datenelemente zu bewahren, während die anderen Aktivitäten und Datenelemente entfernt werden (M1, M5). Ungenutzte Elemente werden abschließend entfernt (M8).

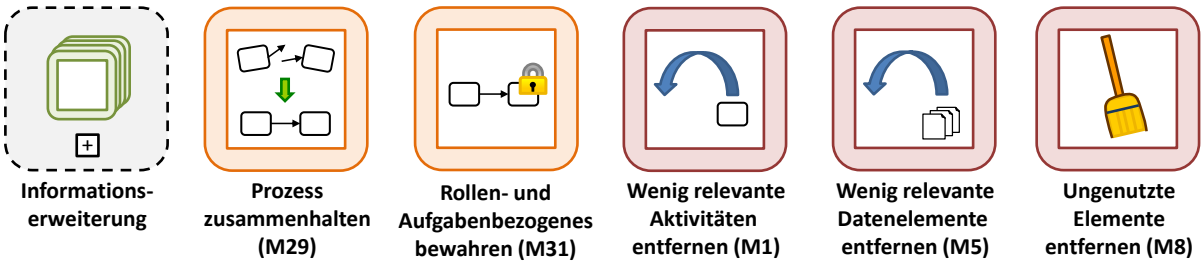


Abbildung 4.3: Mustersequenz zum Entfernen von Details.

Skizze: Die Anwendung der Mustersequenz ist in Abbildung 4.4 skizziert. Abbildung 4.4 (b) veranschaulicht das Resultat einer Informationserweiterung, in der allen Elementen eine Relevanz zugeordnet ist (#1 oder #2). Aktivität *C* ist außerdem mit einer Nutzerannotation versehen. Anschließend werden alle Aktivitäten und Datenelemente mit Relevanz #2 entfernt (M1, M5). Das Entfernen generiert den in Abbildung 4.4 (c) dargestellten Ausgabeprozess, wobei die annotierte Aktivität bewahrt wurde.

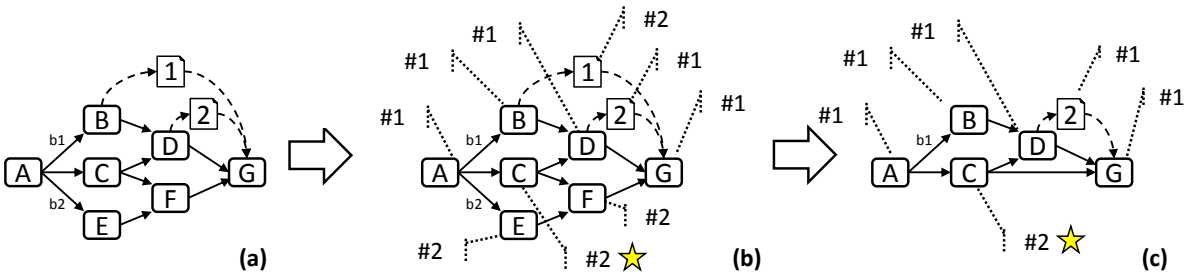


Abbildung 4.4: Skizze zur Informationserweiterung (b) und dem anschließenden Entfernen von Aktivitäten und Datenelementen mit Relevanz #2 (c).

Beispiele: In der Sicht ViewByRelevance beschreibt Bobrik [Bob08] eine allgemeine Relevanzfunktion, die sich auf verschiedene Eigenschaften von Elementen beziehen kann, wie bspw. die der Aktivität als Bearbeiter zugeordnete Rolle. Polyvyanyy et al. [PSW08] erwähnen als Relevanzkriterien u.a. die Dauer und Kosten von Aktivitäten und wie häufig diese ausgeführt wurden. Die Auswahl eines Grenzwertes der Relevanz erfolgt bei Polyvyanyy et al. mit einem Schieberegler.

Die Patentanmeldung [GHN+09] schlägt vor, einfache technische BPEL-Aktivitäten wie `<assign>` oder `<empty>` zu filtern und somit den Fokus auf als wichtiger betrachtete Aktivitäten wie `<invoke>`, Human Tasks und Kontrollstrukturen wie `<if>` zu legen.

4.1.2 Zusammenfassen bekannter Sachverhalte

Problem: Durch ein Entfernen gehen u.U. nicht nur Details oder Einzelheiten, sondern auch Zusammenhänge verloren. Um diese Zusammenhänge zu erhalten, aber die Anzahl der Elemente zu reduzieren, besteht die Möglichkeit, Strukturen zu aggregieren. Aber wie kann ein Sachverhalt als solcher erkannt und abstrakt, aber treffend beschrieben werden?

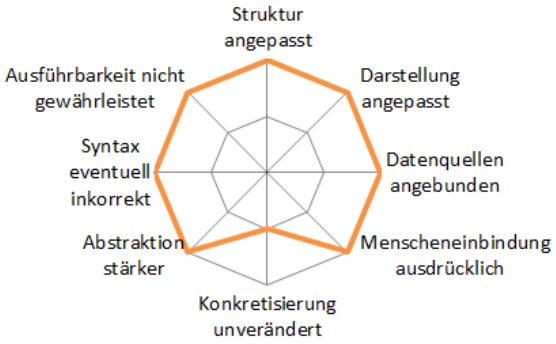


Abbildung 4.5: Klassifikation des Entwurfs.

Lösungsansatz: Das Zusammenfassen bekannter Sachverhalte kann mehrere Informationsquellen einbinden, wie die zugehörige Mustersequenz in Abbildung 4.6 zeigt. Die Nutzung des Expertenwissens von Nutzern über einen Prozess ist ein Ansatz, der die derzeitigen Fähigkeiten von automatischen Techniken bei weitem übersteigt. Daher ist die manuelle Selektion zusammenhängender Strukturen (Prozessfragmente) und deren passende Bezeichnung (M13) eine Aufgabe, die einmalig pro zu unterstützendem Prozess durchgeführt und bei Änderungen am Prozess nachgepflegt werden muss. Diese Vorbereitung ist nicht auf Aktivitäten beschränkt, sondern kann auch Datenelemente einbeziehen. Die Erkennung von bekannten Strukturen bietet die Möglichkeit, derartiges Expertenwissen in einem Automatismus einzusetzen und somit zur Abstraktion wiederzuverwenden (M14). Nach der Erkennung können die gekennzeichneten Strukturen und Datenelemente aggregiert werden (M2, M6). Zur einfacheren Unterscheidbarkeit für den

Nutzer werden die aggregierten Strukturen und Datenelemente grafisch als erkannter Sachverhalt gekennzeichnet und mit der vom Nutzer vorgegebenen Bezeichnung dargestellt (M16, M18).

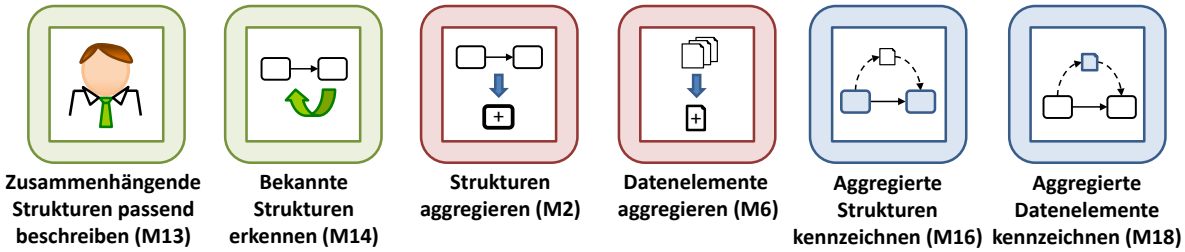


Abbildung 4.6: Mustersequenz zum Zusammenfassen bekannter Sachverhalte.

Skizze: Abbildung 4.7 (b) skizziert die Annotation bekannter oder auch erkannter Strukturen (*A*, *B*, *C*, *D*) mit einer passenden Bezeichnung (hier *A+*). Die Datenelemente (*1*, *2*) werden ebenfalls mit einem passenden Begriff versehen. Der nachfolgende Schritt aggregiert die zusammengehörigen Aktivitäten und Datenelemente und benennt die auf diese Weise zusammengefassten Sachverhalte entsprechend. Auf eine Anpassung der Darstellung wurde in der Skizze verzichtet.

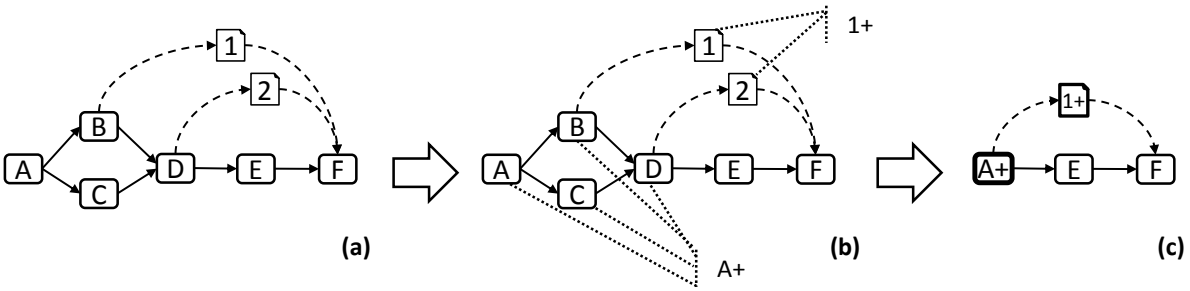


Abbildung 4.7: Skizze zur Annotation bekannter Sachverhalte (b) und deren Zusammenfassung (c).

Beispiele: Das Erkennen von Sachverhalten wurde in verschiedenen Vorarbeiten untersucht. Zunächst wurde vorgeschlagen, Prozessfragmente mit Compliance-Bezug mit treffenden Begriffen zu versehen und diese zu aggregieren, vgl. Schumm et al. [SAL+10b]. Einem entsprechenden Erkennungsalgorithmus wurde in der studentischen Arbeit [Pos10b] nachgegangen. Bibliotheken für Prozessfragmente [SKK+11] können einer automatisierten Erkennung (M14) als Datenquelle dienen. Rensink und Zambon [RZ12] diskutieren eine ähnliche Technik der Graphabstraktion, die auf der Erkennung und der Aggregation bekannter, isomorpher Sub-Graphen basiert.

Bekannte Sachverhalte können durchaus domänenspezifisch sein: In der PICTURE-Methode [PIC06] sind etwa 30 Bausteine definiert, mittels derer die Tätigkeiten in Prozessen der öffentlichen Verwaltung abstrakt beschrieben werden können. Die Annotation der Strukturen eines detaillierten Prozesses dieser Domäne mit der Information, welchem dieser Bausteine sie entsprechen, kann eine Grundlage für das Zusammenfassen darstellen. Dieser Ansatz ist möglicherweise auf andere Domänen übertragbar.

4.1.3 Hervorhebung bestimmter Aspekte

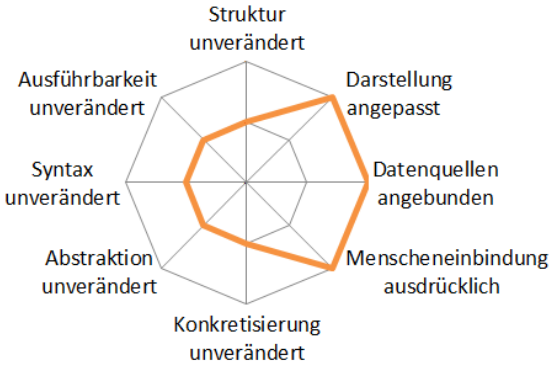


Abbildung 4.8: Klassifikation des Entwurfs.

Problem: Das Entfernen von Details und das Abstrahieren von Zusammenhängen stellen eine Möglichkeit zur Komplexitätsreduzierung dar. Allerdings gehen durch diese Techniken Detailinformationen verloren. Gibt es eine Alternative, bei der die Detailinformationen erhalten bleiben?

Lösungsansatz: Das Hervorheben des Wesentlichen ist seit jeher ein wichtiger Bestandteil des menschlichen Erkenntnisprozesses [Hir91]. Hervorhebung wird häufig über eine Anpassung der ursprünglich verwendeten Farbe oder Größe umgesetzt. Wie eine Farbe wahrgenommen wird ist allerdings abhängig von der Kultur und Sozialisation des Betrachters. Die Farbe Rot wird bspw. nach westlichem Verständnis als dynamischste, aber auch aggressivste Farbe betrachtet. Sie kann durchaus zur Hervorhebung eingesetzt werden, jedoch bei übermäßigem Gebrauch Reizbarkeit, Wut und Stress beim Betrachter auslösen [Thö12]. Große Elemente wie bspw. ein Elefant werden i.d.R. als wichtiger angesehen wie kleine, etwa eine Maus. Analog dazu wären groß dargestellte Aktivitäten wichtiger als kleine.

Eine Größenanpassung hat gegenüber der farblichen Hervorhebung zwei Nachteile: Zum einen muss ein Prozess ggf. neu ausgelegt werden, wodurch sich Aktivitäten in ihrer Platzierung im Koordinatensystem verschieben. Zum anderen kann durch die Größe nur ein einziger Aspekt codiert werden. Bei einer Farbanpassung können hingegen mehrere Aspekte mit unterschiedlichen Farben codiert werden, etwa durch einen Farbverlauf oder durch eine Aufteilung der Zeichenfläche einer Aktivität in mehrere Bereiche. Die Ausprägung der einzelnen Aspekte kann dann durch unterschiedliche Farbintensität codiert werden. Die Skizze in Abbildung 4.9 (a) zeigt einen mit der Ausprägung eines Aspekts angereicherten Prozess, bspw. die von einem Nutzer wahrgenommene Relevanz einzelner Aktivitäten. Entsprechend des zugeordneten Wertes kann die Größe einer Aktivität angepasst werden, wie in Abbildung 4.9 (b) dargestellt.

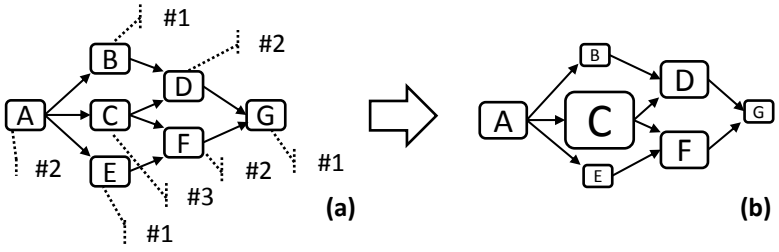


Abbildung 4.9: Eine Größenanpassung stellt die Ausprägung eines bestimmten Aspekts dar.

Die Mustersequenz in Abbildung 4.10 zeigt einen Grobentwurf zur Hervorhebung durch Farbanpassungen. Zunächst werden mit der Gruppierung zur Informationserweiterung (vgl. Abschnitt 4.1.1) Informationen integriert, die zur Unterscheidung der hervorzuhebenden von den nicht hervorzuhebenden Strukturen dient. Die Gruppierung beinhaltet das Einbringen von

Informationen durch Nutzer (M13), die Anreicherung durch statische Analysen (M14) und die Anreicherung mit Laufzeitinformationen (M15). Anschließend werden diese Informationen zur Bewertung der Wichtigkeit herangezogen (M14). Nach dieser Informationserweiterung werden die Farben von Aktivitäten, Verbindungen und Datenelementen angepasst. Die Codierung, welche Farben und Hinweissymbole verwendet werden sollen und auf welche Strukturen diese in welcher Art anzuwenden sind, sind Umfang des Feinentwurfs.

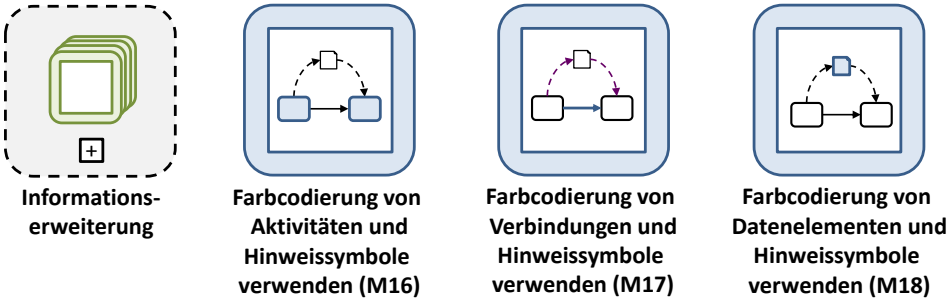


Abbildung 4.10: Mustersequenz zur Hervorhebung bestimmter Aspekte.

Skizze: Abbildung 4.11 (c) skizziert eine Hervorhebung von Aktivitäten (*A*, *F*), Datenelementen (*1*) und Kontrollverbindungen (*A*→*B*, *A*→*C*) mit Farben und Hinweissymbolen. Der Hervorhebung geht die Informationserweiterung voran (b).

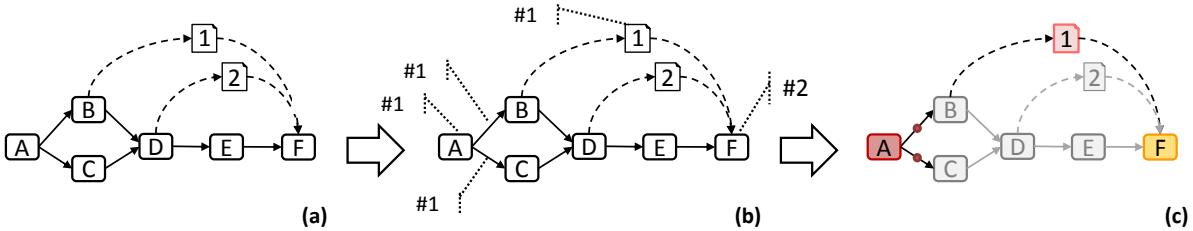


Abbildung 4.11: Skizze zur Hervorhebung bestimmter Aspekte.

Eine Variante besteht im Hervorheben eines Schwerpunkts, wie in Abbildung 4.12 (b) skizziert. Der Schwerpunkt *C* ist mit größter Farbintensität hervorgehoben und *i* Vorgängeraktivitäten und *j* Nachfolgeraktivitäten mit abnehmender Farbintensität, wobei in der Skizze $i = j = 1$. Relevante Vorgänger und Nachfolger können über den Kontrollfluss, aber auch über den Datenfluss bestimmt werden.

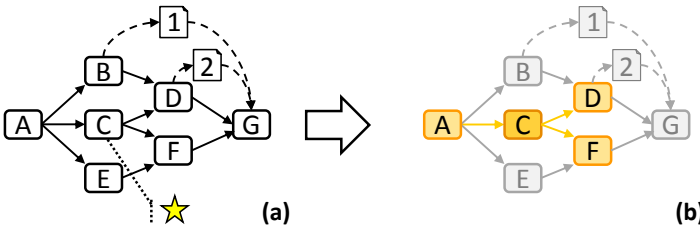


Abbildung 4.12: Skizze zur farblichen Hervorhebung eines als wichtig ausgezeichneten Schwerpunkts.

Beispiele: Die Hervorhebung wird häufig in Modellierungswerkzeugen eingesetzt, um die syntaktische Validierung zu unterstützen: Zunächst werden die im Prozessmodell enthaltenen Strukturen syntaktisch überprüft (M14) und anschließend werden Fehler und Warnungen mit Hinweissymbolen kenntlich gemacht, z.B. mit roten Fehlersymbolen und gelben Warnsymbolen an Aktivitäten (M16). Das Werkzeug Eclipse BPEL Designer [Ecl11] nutzt bspw. Hinweissymbole auf Aktivitäten, um auf Syntaxfehler aufmerksam zu machen.

In dem Modellierungswerkzeug Enterprise Architect 10 [Spa13] kann der Nutzer einen Fokus auf ein Diagrammelement mit dem Radius direkt benachbarter Elemente setzen. Der Fokus bewirkt entweder ein Ausgrauen (M16) oder ein Entfernen (M1) aller Elemente außerhalb des Fokus. Einen Fokus durch das Abschneiden aller Strukturen, die außerhalb liegen, beschreibt Bobrik [Bob08] in der Sicht CutProcess. Wie von Schumm et al. [SAL+10b] vorgeschlagen wurde, können auch mehrere Schwerpunkte sinnvoll sein, z.B. bei allen Aktivitäten, die sich aktuell in der Ausführung befinden.

4.1.4 Informationsschutz

Problem: Unterschiedliche Interessengruppen sind über verschiedene Lebenszyklusphasen hinweg in Prozesse involviert und benötigen Kenntnis über sie betreffende Bereiche. Allerdings sind Geschäftsprozesse Güter, die es zu schützen gilt. Wie könnte eine Sichtbarkeits- und Zugriffssteuerung nach Rollen und Rechten erreicht werden?

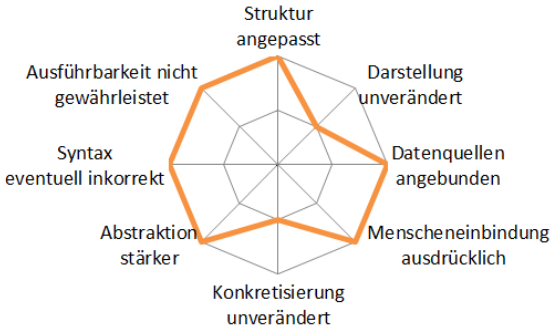


Abbildung 4.13: Klassifikation des Entwurfs.

Lösungsansatz: Abbildung 4.14 zeigt eine Mustersequenz zur Spezifikation des Informationsschutzes in einem Geschäftsprozess. Die Sicht setzt eine Zuordnung der im Prozess enthaltenen Strukturen nach Zugriffsrechten voraus (M13). Diese kann durch Sicherheitsstufen wie z.B. öffentlich, intern, vertraulich oder durch Gruppenberechtigungen erfolgen. Bei Änderungen am Prozess muss diese Zuordnung entsprechend aktualisiert werden. In dieser Sicht soll der Zusammenhang der zugänglichen Informationen erhalten bleiben (M29). Alle Aktivitäten (M1), Datenelemente (M5) und deren Eigenschaften (M4) für die keine Berechtigung besteht, werden aus dem Prozess entfernt. Abschließend werden überflüssige Elemente aus dem Prozess entfernt, um die Komplexität zu reduzieren (M8).

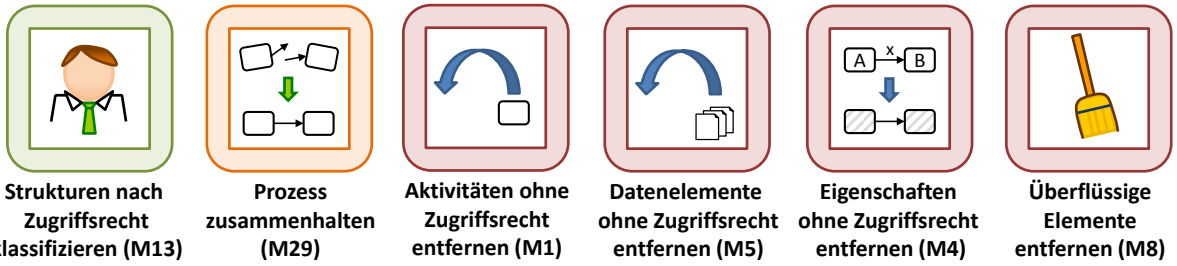


Abbildung 4.14: Mustersequenz zum Informationsschutz.

Skizze: Die Skizze in Abbildung 4.15 ähnelt dem Entfernen von Details (vgl. Abschnitt 4.1.1), allerdings werden in dieser Sicht die vermeintlich relevanten Aspekte entfernt. Die Vereinfachung des Prozesses ist lediglich ein Nebeneffekt dieser Sicht. Die Skizze zeigt einen Prozess (a), für den bestimmte Berechtigungen gesetzt werden (b). In der Skizze steht *x* für volle Sichtbarkeit, *r* für eingeschränkte Sichtbarkeit und - für keine Berechtigung. Strukturen mit - werden entfernt und Strukturen mit *r* abstrahiert (c).

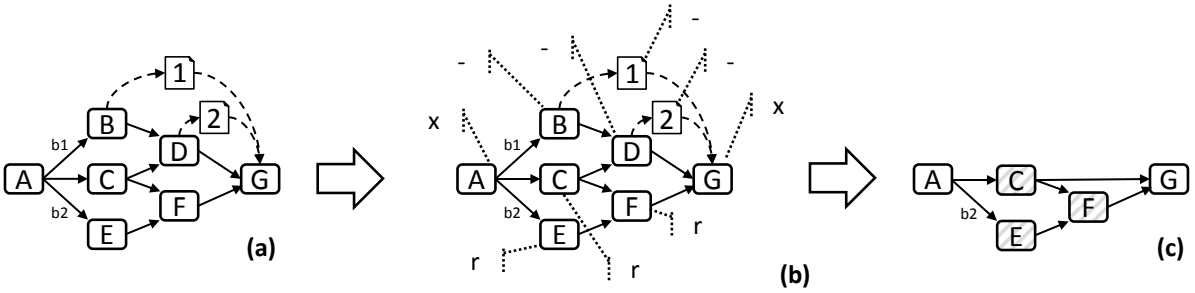


Abbildung 4.15: Skizze zum Informationsschutz.

Beispiele: Informationsschutz ist eine häufig auftretende und wichtige Anforderung. Naef et al. [NSS01] diskutieren Schatten-Prozesse als strukturreduzierte Sichten zur Einhaltung von Anforderungen zum Informationsschutz. BPEL standardisiert eine Unterstützung der Beschreibung von Public Views, bei der Aktivitäten, Eigenschaften und Ausdrücke als `opaque` verborgen werden können, wobei die Information über deren Existenz erhalten bleibt [OAS07]. Der Einsatz von Graphtransformationen kann an dieser Stelle als Technik zur Erzeugung von Public Views verwendet werden [EG08]. Der Einsatz von Annotationen zur Transformationssteuerung wurde von Benameur et al. [BMR08] eingehender diskutiert. Die von Reichert et al. [RBR+10] diskutierten Zugriffskontrollmechanismen in dem Werkzeug Proviado stellen eine mögliche Implementierung dieses Sichtenentwurfs für die Nutzung im Monitoring dar.

4.2 Sichten für die Prozessmodellierung

4.2.1 Variierende Granularität

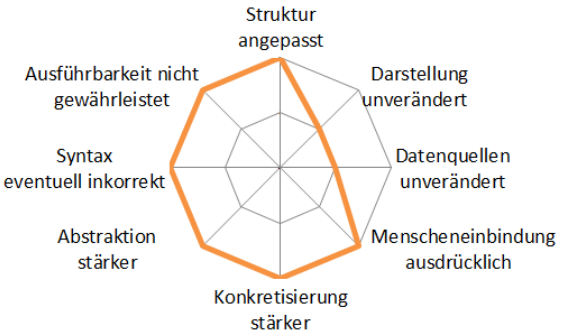


Abbildung 4.16: Klassifikation des Entwurfs.

Problem: Der Abstraktionsgrad kann innerhalb eines Prozessmodells variieren, wenn dieses durch Einfluss verschiedener Personen erstellt wurde und deren Beschreibung einen unterschiedlichen Grad an Detaillierung aufweist [JFE+11]. Unterschiedliche Kenntnisse über den Prozess, unterschiedliche Wahrnehmung der Prozessgrenzen und ein unterschiedlich tiefes Verständnis der Prozesssprache tragen ebenfalls dazu bei, dass variierende Granularität entsteht. Anzeichen für variierende Granularität innerhalb eines Prozesses sind außerordentliche Kosten oder eine überdurch-

schnittliche Ausführungsdauer von Aktivitäten im Vergleich zu den übrigen Aktivitäten im Prozess [DHL+12]. Dauert bspw. die Ausführung einer bestimmten Aktivität i.d.R. mehrere Tage, aber die der übrigen Aktivitäten dauern nur wenige Minuten, so liegt eine variierende Granularität vor. Granularität ist nicht nur für das Verständnis, sondern auch für die Effektivität der Wiederverwendung von Prozesswissen von Bedeutung, da ein Prozess und wiederzuverwendende Strukturen im Abstraktionsgrad zusammenpassen sollten [HRL09]. Wie kann man mit variierender Granularität umgehen?

Lösungsansatz: Bei der Prozessmodellierung mit variierender Granularität ist entscheidend, ob bestimmte Muster generell unterstützt werden – unabhängig von der Abfolge in einer Sequenz. Die für variierende Granularität maßgeblichen Muster können untergliedert werden in Muster zur Erhöhung des Abstraktionsgrades (s. Abbildung 4.17) und in Muster zur stärkeren Konkretisierung und Verfeinerung (s. Abbildung 4.18). Sie decken die beiden Fälle ab, dass bestimmte Strukturen zu feingranular oder zu grobgranular sind.

Zu Feingranular: Um die Granularität von detaillierten Prozesssachverhalten zu verringern, kann das Entfernen von Aktivitäten (M1), Datenelementen (M5) und Eigenschaften (M4) verwendet werden. Auch eine Aggregation von Eigenschaften (M4), Aktivitäten (M2, M3) und Datenelementen (M6) kann zur Erhöhung der Abstraktion genutzt werden. Bestimmte Aktivitätssequenzen können als Pfad aggregiert werden (M7). Durch eine Protokollierung vorgenommener Abstraktionsschritte (M32) können diese erneut auf das vollständige Modell angewendet werden und somit einen Replay-Mechanismus realisieren. Dadurch bleibt der Ursprungsprozess erhalten und abstrakte Sichten können bei Bedarf erzeugt werden.

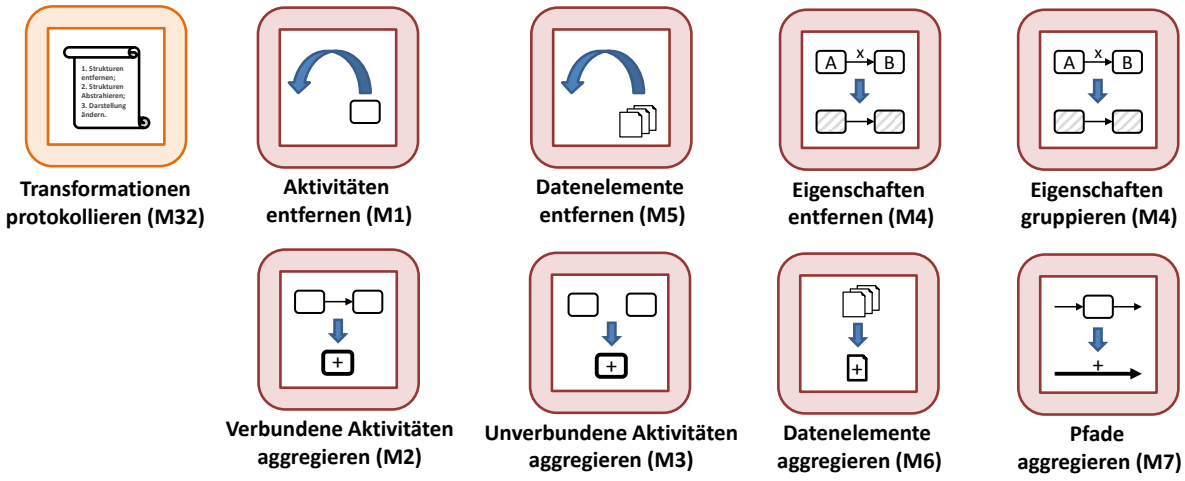


Abbildung 4.17: Auflistung von Mustern zur Unterstützung variierender Granularität (Abstraktion).

Zu Grobgranular: Durch eine Verfeinerung können zu grobgranulare Strukturen in einer Sicht vorübergehend detailliert werden. Sofern vorgenommene Verfeinerungen entsprechend protokolliert werden (M32), können diese bei Bedarf erneut auf den Ursprungsprozess angewendet werden.

Sofern die verfeinerte Sicht persistiert wird und den Ursprungsprozess ersetzt, kann die Protokollierung genutzt werden, um eine Mustersequenz zu erzeugen, die die ursprüngliche Granularität wiederherstellt. Dazu muss für jede angewendete verfeinernde Transformation

eine inverse reduzierende Transformation aufgezeichnet werden. Wird bspw. eine zusätzliche Aktivität eingefügt (M9), kann sie für die ursprüngliche Granularität ausgeblendet werden (M1). Wird eine Aktivität für eine Verfeinerung zerlegt (M10), kann sie für die ursprüngliche Granularität mit der ursprünglichen Bezeichnung aggregiert werden (M2). Ebenso können Datenelemente durch eine Zerlegung verfeinert (M11) und der ursprüngliche Zustand durch eine Aggregation wiederhergestellt werden (M6). Analog können Kontrollverbindungen als Pfad zerlegt (M12) als auch wieder zu einer Kontrollverbindungen aggregiert werden (M7).

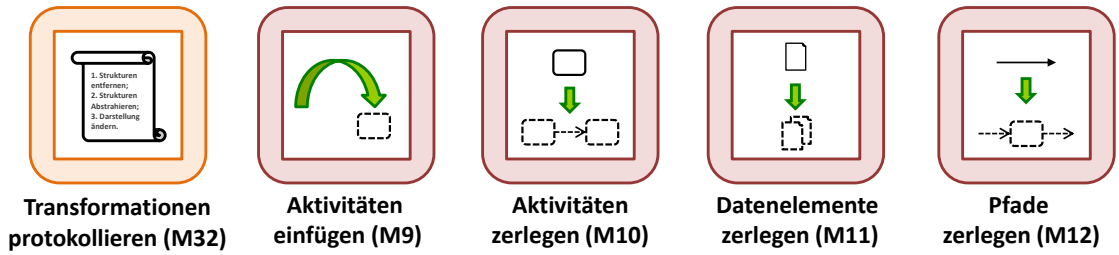


Abbildung 4.18: Auflistung von Mustern zur Unterstützung variierender Granularität (Verfeinerung).

Skizze: Der Sachverhalt variierender Granularität ist in Abbildung 4.19 skizziert. Die Transformation von einem Ursprungsprozess (b) zu einer abstrahierenden Sicht (a) repräsentiert den Fall eines zu feingranularen Prozesses, wobei Aktivität *C* und Datenelement *1* ausgeblendet werden. Die verfeinernde Sicht (c) repräsentiert den Fall eines zu grobgranularen Prozesses, wobei die Aktivität *F* in die Aktivitäten F_1 und F_2 zerlegt wurde.

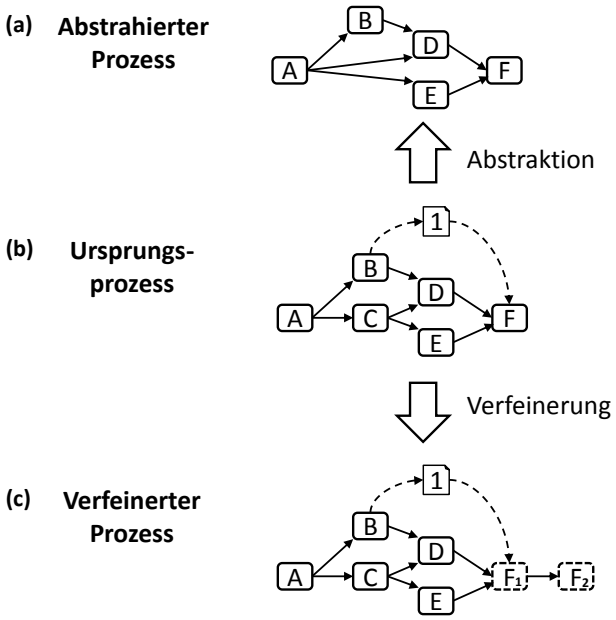


Abbildung 4.19: Skizze zur Angleichung der Granularität eines Ursprungsprozesses durch Abstraktion (a) und Verfeinerung (c).

Beispiele: Kolb und Reichert [KR13a] diskutieren verschiedene Abstraktionsstufen zur Unterstützung der Modellierung, wobei sie zur Abstraktion von den in Abbildung 4.17 gezeigten Mustern Gebrauch machen. Zhao et al. [ZLS+09] zeigen mit den FlexView-Framework eine andere Herangehensweise für den hierarchischen Aufbau aggregierter Strukturen, um so ein Navigieren zwischen unterschiedlichen Abstraktionsebenen zu ermöglichen.

Während manuelles und dauerhaftes Einfügen von Aktivitäten (M9) in Modellierungswerkzeugen eine elementare Funktion darstellt, ist das Zerlegen von atomaren Aktivitäten (M10), Datenelementen (M11) und Verbindungen (M12) bisher nicht gängig. Weber et al. [WRR08] beschreiben zwar die Bildung von Hierarchien durch das Extrahieren und Einfügen von Sub-Prozessen als übliche Technik in der Modellierung, allerdings haben Sub-Prozesse eine bestimmte Ausführungssemantik, während aggregierte und zerlegte Aktivitäten zu Darstellungszwecken dienen.

4.2.2 Ausführungskonfiguration

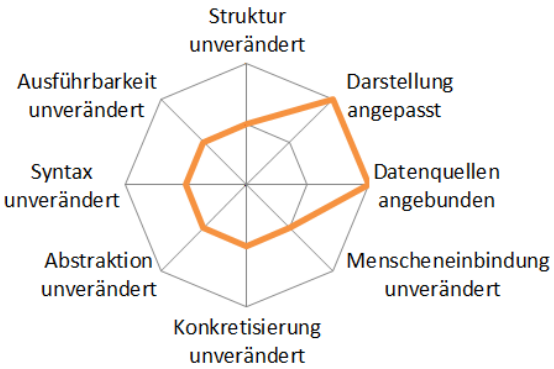


Abbildung 4.20: Klassifikation des Entwurfs.

Problem: Konfigurationen zur Ausführung eines Workflows werden i.d.R. separat vom Prozessmodell verwaltet, damit die Wiederverwendbarkeit erhöht und ein Prozessmodell nicht durch diese Informationen belastet wird. Diese Konfigurationen umfassen bspw. nicht-funktionale Eigenschaften wie etwa Sicherheitseinstellungen. Durch die Trennung der Konfiguration vom Prozessmodell müssen die Zusammenhänge jedoch manuell hergestellt werden, wodurch eine Gedächtnisbelastung entsteht. Gibt es eine Möglichkeit, diese Informationen temporär zusammenzuführen?

Lösungsansatz: Informationsintegrierende Sichten können dazu eingesetzt werden, gesondert vom Prozessmodell gehaltene Konfigurationsaspekte im Prozesskontext zur Analyse und Änderung temporär zugänglich zu machen, vgl. [SAL+10b]. Nach einer Änderung wird die aktualisierte Konfiguration wieder gesondert vom Prozessmodell gespeichert. Eine Informationsintegration findet in diesem Sinne nur temporär im Speicher des Modellierungswerkzeugs statt. Wie die Mustersequenz in Abbildung 4.21 zeigt, ist der Erhalt der Ausführbarkeit (M30) eine Anforderung dieser Sicht. Als erster Transformationsschritt wird die zu integrierende Konfiguration mit den Prozesselementen verknüpft (M15). Bestimmte Eigenschaften können über Anpassungen am Erscheinungsbild von Aktivitäten (M16) und Daten (M18) codiert werden. Mit dem Einsatz des Graphschemas (M19) und der Organisation nach einer ausgewählten Konfigurationseigenschaft mittels Swimlanes (M26) kann eine weitere Konfigurationseigenschaft dargestellt und durch das Verschieben einer Aktivität auf eine andere Swimlane geändert werden.

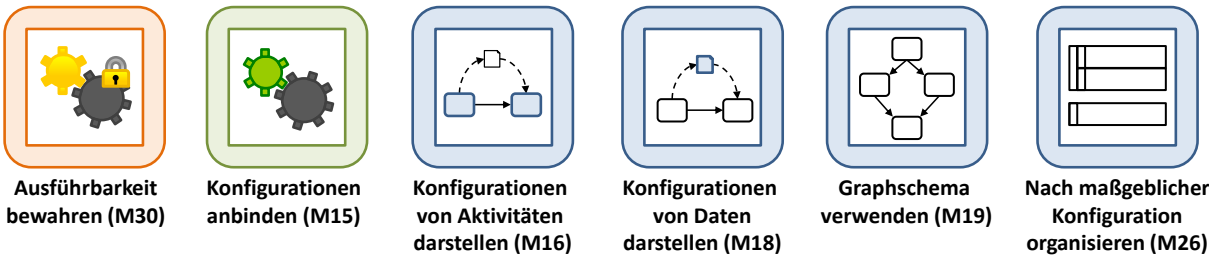


Abbildung 4.21: Mustersequenz für eine Sicht auf die Ausführungskonfiguration eines Prozesses.

Skizze: Die Skizze in Abbildung 4.22 zeigt einen Eingabeprozess (a), der mit Konfigurationsinformationen verknüpft wird (b), die zum Layout mit Swimlanes verwendet wird (c). Weitere Informationen werden durch die Rahmenfarbe und Hinweissymbole codiert.

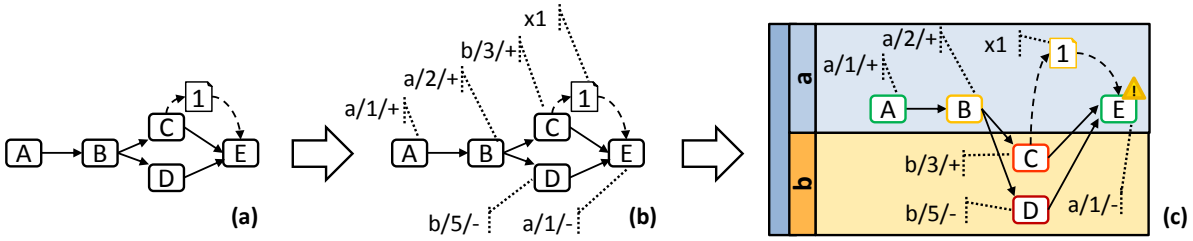


Abbildung 4.22: Skizze zur Sicht auf die Ausführungskonfiguration eines Prozesses.

Beispiele: Verschiedene Werkzeuge bieten eine integrierte Sicht zur Unterstützung der Ausführungskonfiguration, etwa zur Anpassung eines Deployment Descriptors. Die Technik der Nutzung von Swimlanes zur Codierung und der einfachen Anpassung von Aktivitätsbezogenen Einstellungen wird z.B. im WebSphere Business Modeler [IBM09] angeboten.

4.3 Sichten für das Monitoring und die Analyse von Prozessen

Sichten für das Monitoring und die Analyse sind vornehmlich an den dynamischen Aspekten von Prozessen ausgerichtet. Eine Anreicherung mit statischen und dynamischen Daten ermöglicht eine grafische Darstellung dieser Daten in ihrem Kontext, dem Geschäftsprozess.

4.3.1 Vereinfachte Prozessinstanz

Problem: Durch eine Anreicherung mit Laufzeitinformationen kommen weitere Details hinzu und die Prozesskomplexität wird erhöht. Wie kann die Komplexität wieder reduziert werden, ohne signifikante Informationen zu verlieren?

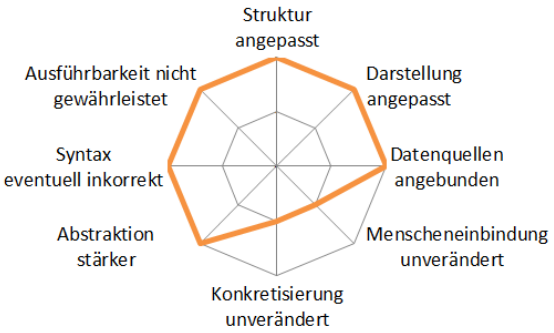


Abbildung 4.23: Klassifikation des Entwurfs.

Lösungsansatz: Laufzeitinformationen geben Aufschluss über den Status einer Prozessinstanz und sie stellen ein Merkmal dar, um im Kontext der Prozessinstanz Wichtiges von Unwichtigem unterscheiden zu können. Diejenigen Elemente, die für eine bestimmte Prozessinstanz unwichtig sind, können ausgeblendet werden.

Die in Abbildung 4.24 dargestellte Mustersequenz spezifiziert die Erzeugung einer Sicht auf eine vereinfachte Prozessinstanz. Die Sequenz beginnt mit der Anreichung des Prozesses mit Laufzeiten und Aktivitätszuständen (M15). Für die reduzierenden Strukturveränderungen gilt die Vorgabe, den Prozess zusammenzuhalten (M29). Übersprungene Aktivitäten und Aktivitäten mit kurzer Ausführungsdauer werden ausgeblendet (M1). Die Angabe eines Schwellenwerts bzgl. der Ausführungsdauer bietet die Möglichkeit, diese Transformation im Feinentwurf zu parametrieren. Die Darstellung von Aktivitäten in einem Kompaktmodus mit den maßgeblichen Informationen Zustand, Typ und Bezeichnung (M16) reduziert und abs-

trahiert die tatsächlich verfügbare Informationsmenge. Ein Layout im Graphschema (M19) mit einer Orientierung von oben nach unten (M24) eignet sich bspw. für eine webbasierte Monitoring-Anwendung technischer Prozesse, da technische Prozesse häufig von oben nach unten modelliert werden und Webbrowser zudem ein vertikales Scrollen in der Bedienung bevorzugt unterstützen.

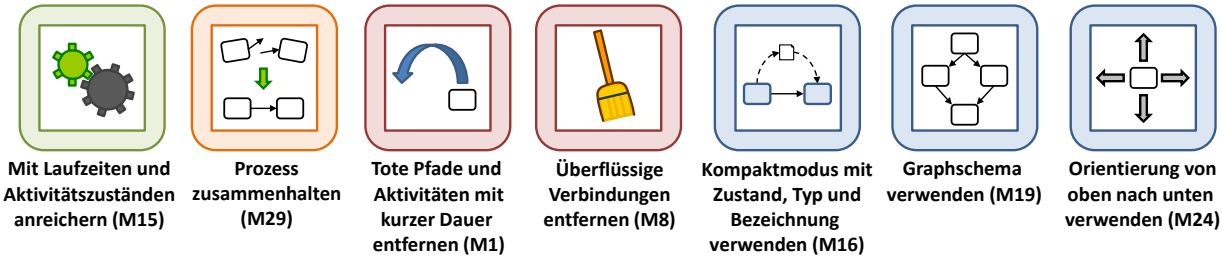


Abbildung 4.24: Mustersequenz für eine vereinfachende Sicht auf eine Prozessinstanz.

Skizze: Die Skizze in Abbildung 4.25 zeigt ein Prozessmodell (a), das mit Laufzeitinformationen angereichert wird (b). In der nachfolgenden Vereinfachung (c) wird Aktivität *B* entfernt, da diese unter einem Schwellenwert liegt; Aktivität *F* wird entfernt, da diese übersprungen wurde. Das Resultat dieser Transformationen wird als Graph von oben nach unten ausgelegt und der Aktivitätszustand über die Rahmenfarbe codiert (d).

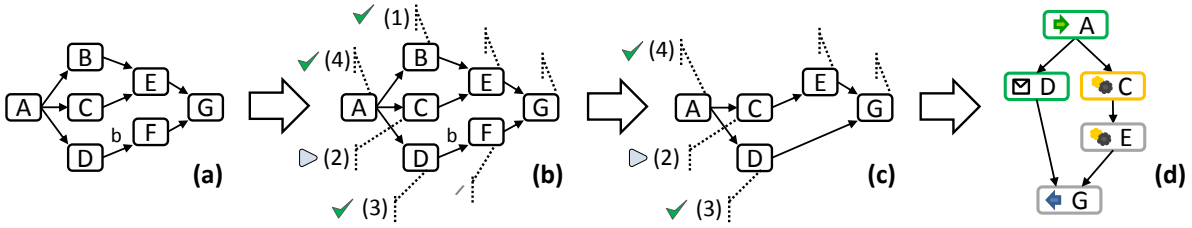


Abbildung 4.25: Skizze für eine vereinfachende Sicht auf eine Prozessinstanz.

Beispiele: Sichten für das Monitoring wurden u.a. von Bobrik [Bob08] diskutiert. Bobrik sieht neben dem Einsatz grafischer Vorlagen zur Darstellung von Startzeit, Stopzeit, Zustand und Dienstgüteeigenschaften außerdem vereinfachende Transformationen vor. In der Sicht AggrExecutedPart werden abgeschlossene und übersprungene Strukturen aggregiert. In der Sicht ShowExecutedPath werden tote Pfade entfernt.

4.3.2 Einfache Zeitanalyse

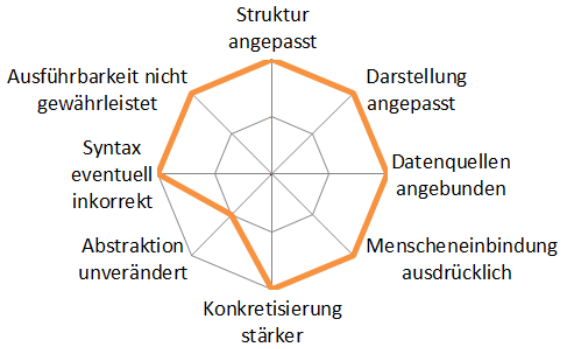


Abbildung 4.26: Klassifikation des Entwurfs.

Problem: Neben der Identifikation von Engpässen (engl. *bottlenecks*) ist Zeitmanagement ein wichtiger Aspekt in der Prozessanalyse. Liegezeiten zwischen Aktivitäten können die gesamte Dauer eines Prozesses negativ beeinflussen. Derartige Laufzeitinformationen stehen ggf. in Form von protokollierten Events zur Verfügung, jedoch nicht im unmittelbaren Kontext, dem Prozess. Wie können diese Laufzeitinformationen mit dem Prozess verknüpft werden?

Lösungsansatz: Um Engpässe von einer gewöhnlichen Ausführung zu unterscheiden ist die Angabe plausibler Zielwerte grundlegend (M13). Die für eine einfache Zeitanalyse relevanten Start- und Stopzeiten von Aktivitäten werden aus der Datenhaltung der Monitoring-Umgebung extrahiert und mit dem Prozessmodell verknüpft (M15). Auf dieser Basis können anschließend die Liegezeiten berechnet werden (M14). Liegen die berechneten Liegezeiten zwischen zwei Aktivitäten über dem Schwellenwert, können diese Liegezeiten als Warteaktivitäten greifbar gemacht werden (M9). Zur Unterstützung der Analyse werden Start- und Stopzeiten mit einbezogen (M16) und eingefügte Warteaktivitäten als Uhr dargestellt (M16).

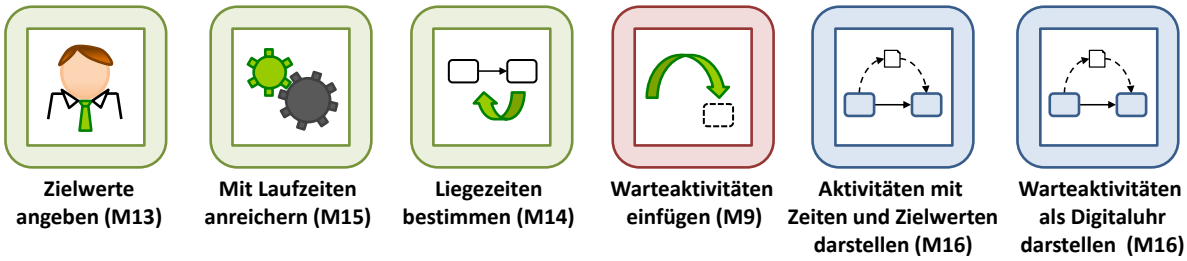


Abbildung 4.27: Mustersequenz für eine einfache Zeitanalyse.

Skizze: Die Skizze in Abbildung 4.28 zeigt einen Eingabeprozess (a), der mit Start- und Stopzeiten angereichert wird, anhand derer die Liegezeiten zwischen Aktivitäten bestimmt werden (b). Ist ein Zielwert verfehlt, kann das in der Sicht (c) durch einen roten Aktivitätsrahmen codiert werden, wie bei Aktivität *D* beispielhaft dargestellt. Eine nähere Betrachtung der Start- und Stopzeiten dieser Skizze ergibt, dass Aktivität *D* am Rand des Arbeitstages um 16:20 Uhr begonnen wurde und am darauffolgenden Morgen um 09:10 Uhr abgeschlossen wurde. Ferner bestehen gewisse Liegezeiten auf den Pfaden *A*→*C* und *B*→*D*.

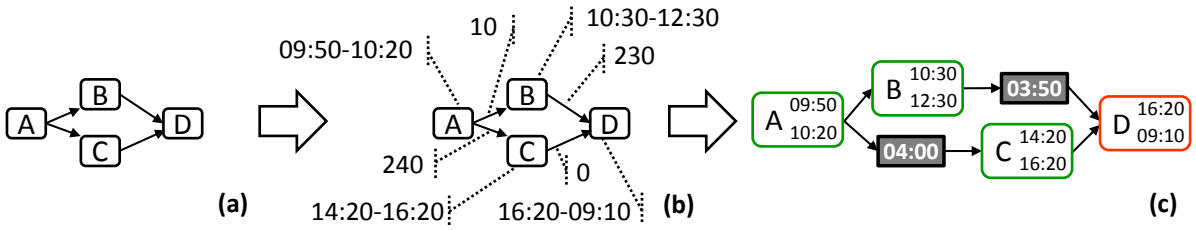


Abbildung 4.28: Skizze für eine einfache Zeitanalyse.

Beispiele: Lange Wartezeiten bei einem Untersuchungsprozess in der Notaufnahme sind für Patienten u.U. kritisch. Wartezeiten können sich durch die Dauer der Untersuchungsschritte ergeben, jedoch können auch Liegezeiten eine Ursache für eine lange Dauer darstellen.

4.4 Übersicht der Verwendung der Sichtenentwurfsmuster

Tabelle 4.1 zeigt die Verwendung der Sichtenentwurfsmuster in den Sichtenentwürfen. An der Zahl der Verwendung der einzelnen Sichtenentwurfsmuster wird deutlich, dass das Entfernen von Strukturen unter Einhaltung von Konsistenz häufig ein Bestandteil der Sichtenentwürfe ist. Ebenfalls ist auffällig, dass die Informationsanreicherung intensiv verwendet wird. Auch wird das Erscheinungsbild von Aktivitäten häufig angepasst.

Tabelle 4.1: Verwendung der Sichtenentwurfsmuster in den Sichtenentwürfen.

			1. Entfernen von Details	2. Zusammenfassen bek. Sachverhalte	3. Hervorhebung bestimmter Aspekte	4. Informationsschutz	5. Variierende Granularität	6. Ausführungskonfiguration	7. Vereinfachte Prozessinstanz	8. Einfache Zeitanalyse	**Anzahl an Verwendungen**
Strukturveränderung	Reduzierung	1. Aktivitäten entfernen	x			x	x		x		6
		2. Verbundene Aktivitäten aggregieren		x			x				2
		3. Unverbundene Aktivitäten aggregieren					x				1
		4. Eigenschaften abstrahieren				x	x				2
		5. Datenelemente entfernen	x			x	x				3
		6. Datenelemente aggregieren		x			x				2
		7. Pfade aggregieren					x				1
		8. Strukturen aufräumen	x			x			x		3
	Erweiterung	9. Aktivitäten einfügen					x			x	2
		10. Aktivitäten zerlegen					x				1
		11. Datenelemente zerlegen					x				1
		12. Pfade zerlegen					x				1
Informations-anreicherung		13. Nutzergestützt anreichern	x	x	x	x				x	7
		14. Statisch anreichern	x	x	x			x		x	6
		15. Dynamisch anreichern	x		x				x	x	6
Darstellungsanpassung	Erscheinungsbild	16. Erscheinungsbild von Aktivitäten anpassen		x	x			x	x	x	7
		17. Erscheinungsbild von Verbindungen anpassen			x			x			3
		18. Erscheinungsbild von Datenelementen anpassen		x	x						3
	Schema	19. Graphschema verwenden						x	x		2
		20. Baumschema verwenden									0
		21. Blockschema verwenden									0
		22. Codeschema verwenden									0
		23. Tabellenschema verwenden									0
	Layout	24. Orientierung festlegen							x		1
		25. Phasenweise auslegen									0
		26. Organisatorisch auslegen						x			1
		27. Räumlich auslegen									0
		28. Maßgeschneidert auslegen									0
Vorgabe		29. Konsistenz bewahren	x			x			x		3
		30. Ausführbarkeit bewahren						x			1
		31. Strukturen bewahren	x								1
		32. Transformationen protokollieren					x				1
		33. Nachvollziehbarkeit gewährleisten									0
		Anzahl verwendeter Muster	**8**	**6**	**6**	**6**	**12**	**6**	**7**	**5**	

KAPITEL 5

SPRACHÜBERGREIFENDE SICHTEN

Sprachübergreifende Sichten erweitern die Möglichkeiten, die durch den musterbasierten Sichtenentwurf erschlossen wurden. Sie umfassen die strukturelle und visuelle Anpassung eines Prozesses hinsichtlich der ihm zugrunde liegenden Prozesssprache, um dadurch dem Bedarf einer bestimmten Interessengruppe nachzukommen.

In diesem Kapitel werden drei zentrale Ansätze zum Entwurf sprachübergreifender Sichten diskutiert. *Prozessübersetzende Sichten* basieren auf der Erzeugung persistierbarer Prozessmodelle unter Einsatz von Übersetzungstechniken. Sie sind im Stand der Technik vorhanden, wurden jedoch bisher nicht mit den umfassenden Transformationsmöglichkeiten des vorgestellten Musterkatalogs in Beziehung gesetzt. Das Konzept der *Sprachemulation* ahmt eine Prozesssprache grafisch nach, ohne ein persistierbares Prozessmodell zu erzeugen, das tatsächlich der Syntax der nachgeahmten Prozesssprache entspricht. Dieses Konzept wurde bereits an verschiedenen Stellen eingesetzt. Es wird in diesem Kapitel unter Einsatz der Mustersprache angewendet, um eine vereinfachende Prozessdarstellung zu beschreiben. Der Stand der Technik wird durch *instanzprojizierende Sichten* erweitert, die eine vom Ursprungsprozess und dessen Prozesssprache losgelöste Möglichkeit zum Entwurf von Sichten auf laufende Prozesse schaffen. Bei einer Instanzprojektion werden Zustandsinformationen von einem mit Laufzeitinformationen angereicherten Prozessmodell auf ein anderes, vordefiniertes Prozessmodell übertragen, wobei das Ziel einer solchen Instanzprojektion in einer anderen Prozesssprache vorliegen kann und nicht notwendigerweise syntaktisch korrekt sein muss. Bspw. könnte ein Prozessfragment als Ziel einer Instanzprojektion verwendet werden.

Die Inhalte in diesem Kapitel basieren im Wesentlichen auf Beiträgen zur sprachübergreifenden Zustandsübertragung (Schumm et al., [SLL+11]) und deren Einsatz für BPEL und Chevron-Diagramme (Schumm et al., [SKL+11]) sowie der Konzeption des sichtenbasierten Werkzeugs zum Monitoring von Prozessen (Schumm et al., [SLL11]). Außerdem beinhaltet der Artikel zur Visualisierung und Modellierung von BPEL-Prozessen mit BPMN (Schumm et al., [SKL+09]) den Grundgedanken der Sprachemulation. Die in diesem Kapitel gezeigten musterbasierten Sichtenentwürfe wurden im Rahmen der vorliegenden Dissertation erstellt und nicht zuvor veröffentlicht.

5.1 Konzeption des sprachübergreifenden Sichtenentwurfs

Nachfolgend werden Begrifflichkeiten zur Beschreibung sprachübergreifender Sichten definiert. Auf dieser Basis wird der Bezug zum musterbasierten Sichtenentwurf hergestellt.

5.1.1 Terminologie sprachübergreifender Sichten

Für die Betrachtung der Formen sprachübergreifender Sichten auf Prozessmodelle und Prozessinstanzen ist eine Erweiterung des in Abschnitt 3.2 vorgestellten konzeptionellen Modells und seiner Terminologie erforderlich.

Definition 9: Die *Quellsprache* einer sprachübergreifenden Sicht ist die Prozesssprache des Eingabeprozesses.

Definition 10: Die *Zielsprache* einer sprachübergreifenden Sicht ist die Prozesssprache des Ausgabeprozesses.

Definition 11: Eine *Sprachemulation* ist die Darstellung eines Prozesses oder einer Prozessinstanz in einer von der Quellsprache verschiedenen Zielsprache auf Basis von Sichtentransformationen.

Definition 12: Eine *Prozessübersetzung* bezeichnet die Übersetzung eines Prozesses von einer Quellsprache in eine Zielsprache auf Basis von Übersetzungsalgorithmen.

Definition 13: Eine *Zustandsübertragung* ist eine Abbildung des Zustands einer oder mehrerer Aktivitäten auf den Zustand einer anderen Aktivität. Eine Zustandsübertragung ist sprachübergreifend, wenn die Aktivitäten unterschiedlichen Quell- und Zielsprachen zugeordnet sind.

Definition 14: Eine *Datenübertragung* ist eine Abbildung von Datenelementen und Kontextinformationen eines mit Laufzeitinformationen angereicherten Prozessmodells auf Datenelemente und Kontextinformationen eines anderen Prozessmodells. Eine Datenübertragung ist sprachübergreifend, wenn die Datenelemente und Kontextinformationen unterschiedlichen Quell- und Zielsprachen zugeordnet sind.

Definition 15: Eine *Zustandsprojektion* bezeichnet die Durchführung der Gesamtheit der definierten Zustandsübertragungen auf Basis angereicherter Laufzeitinformationen.

Definition 16: Eine *Datenprojektion* bezeichnet die Durchführung der Gesamtheit der definierten Datenübertragungen auf Basis angereicherter Laufzeitinformationen.

Definition 17: Eine *Instanzprojektion* umfasst die Zustandsprojektion und Datenprojektion eines mit Laufzeitinformationen angereicherten Prozessmodells (Quellmodell) auf ein anderes Prozessmodell (Zielmodell). Eine Instanzprojektion ist sprachübergreifend, wenn Quellmodell und Zielmodell in unterschiedlichen Sprachen vorliegen.

5.1.2 Hilfselemente für den sprachübergreifenden Sichtenentwurf

Um eine nahtlose Verbindung sprachübergreifender und herkömmlicher Sichtenentwürfe herzustellen, wurden zwei Hilfselemente erstellt, die wie ein Sichtenentwurfsmuster in Entwürfen verwendet werden können. Der gewählte Stil der Beschreibung dieser Hilfselemente unterstützt deren Verwendung im Kontext des musterbasierten Sichtenentwurfs. Die Farbcodierung *lila* hebt die Symbole sprachübergreifender Transformationen mit Kurzbezeichnung *S* von den Sichtenentwurfsmustern mit Kurzbezeichnung *M* ab. Während die Prozessübersetzung (S1) noch als bewährte Technik betrachtet werden kann, stellt das Projizieren des Zustands (S2) eine neuartige Technik dar und ist damit auch kein Muster im Sinne der Sichtenentwurfsmuster. Diese beiden Hilfselemente werden an dieser Stelle grob beschrieben und deren Bezug zum musterbasierten Sichtenentwurf in den nachfolgenden Abschnitten dieses Kapitels vertieft.

Sprachübergreifendes Hilfselement 1: Prozess übersetzen

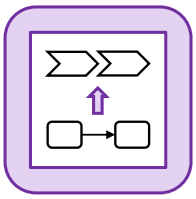


Abbildung 5.1: Symbol zu „Prozess übersetzen (S1)“.

Problem: Wie kann ein Prozess in einer anderen Prozesssprache beschrieben werden?

Schlagwörter: Prozess dolmetschen; Prozess übertragen.

Beschreibung: Die Übersetzung von Prozessmodellen dient unterschiedlichen Zwecken:

- Übersetzung eines fachlichen Prozessmodells in eine ausführungsnahe Sprache mit dem Ziel der Automatisierung des übersetzten Prozesses.
- Übersetzung eines Prozessmodells in eine formale Sprache mit dem Ziel der Verifikation und dem Nachweis bestimmter Eigenschaften des übersetzten Prozesses.
- Übersetzung eines ausführbaren Prozessmodells in eine fachlich genutzte Sprache mit dem Ziel verbesserter Kommunikation zwischen IT und Business auf Basis des übersetzten Prozesses.

Abbildung 5.2 skizziert die Übersetzung eines PM-Graphen nach BPMN. Wie diese Skizze andeutet, können bei der Übersetzung Elemente wie etwa Gateways hinzugefügt und Elemente wie Aktivitäten und Bedingungen modifiziert werden. Abhängig von den jeweiligen Prozesssprachen und dem konkreten Eingabeprozess einer Übersetzung können auch Elemente entfernt, neu arrangiert oder durch andere Konstrukte ersetzt werden.

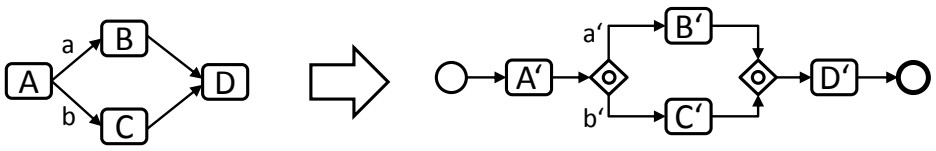


Abbildung 5.2: Skizze zu „Prozess übersetzten (S1)“.

Beispiele: Motiviert von den beschriebenen Zwecken der Prozessübersetzung wurden verschiedene Übersetzungsalgorithmen und Ansätze vorgeschlagen, wie bspw. zur Übersetzung von BPMN zu BPEL [ODH+08], von EPK, BPEL und YAWL zu Petri Netzen [LVD08] als auch zur Abbildung von BPEL auf BPMN zur Verbesserung des Business-IT-Alignment [WDG+08].

Sprachübergreifendes Hilfselement 2: Zustand projizieren

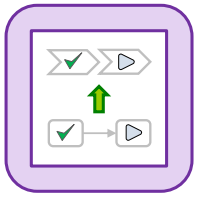


Abbildung 5.3: Symbol zu „Zustand projizieren (S2)“.

Problem: Wie kann der Ausführungszustand eines Prozesses auf Basis eines anderen Prozesses dargestellt werden?

Schlagwörter: Zustand übertragen; Zustand abbilden.

Beschreibung: Der Ansatz der Zustandsprojektion besteht darin, einen bereits mit Zustandsinformationen angereicherten Prozess als Ausgangsbasis heranzuziehen und dessen Zustand durch Anwendung verschiedener Formen der Zustandsübertragung auf ein Zielmodell zu projizieren, welches sich semantisch vollkommen vom Eingabeprozess unterscheiden und zudem in einer anderen Prozesssprache beschrieben sein kann. Eine Zustandsprojektion als Gesamtabbildung ist *surjektiv*, d.h. es gibt für jedes Element des Zielmodells eine Übertragungsregel mit der sein Zustand bestimmt werden kann. Eine Einzelabbildung, die durch eine einzelne Übertragungsregel beschrieben wird, kann eine oder mehrere Aktivitäten des Quellmodells einbeziehen. Wird bei einer Regel nur eine Aktivität des Quellmodells einbezogen, ist diese Einzelabbildung bijektiv (d.h. eineindeutig).

Die Skizze in Abbildung 5.4 (a) deutet verschiedene Regeln zur Zustandsübertragung von einem PM-Graphen auf ein Prozessmodell in einer anderen Prozesssprache an. Abbildung 5.4 (b) konkretisiert diese Zustandsübertragungsregeln. Abbildung 5.4 (c) veranschaulicht den Effekt der konkreten Durchführung einer Zustandsprojektion auf Basis angereicherter Laufzeitinformationen.

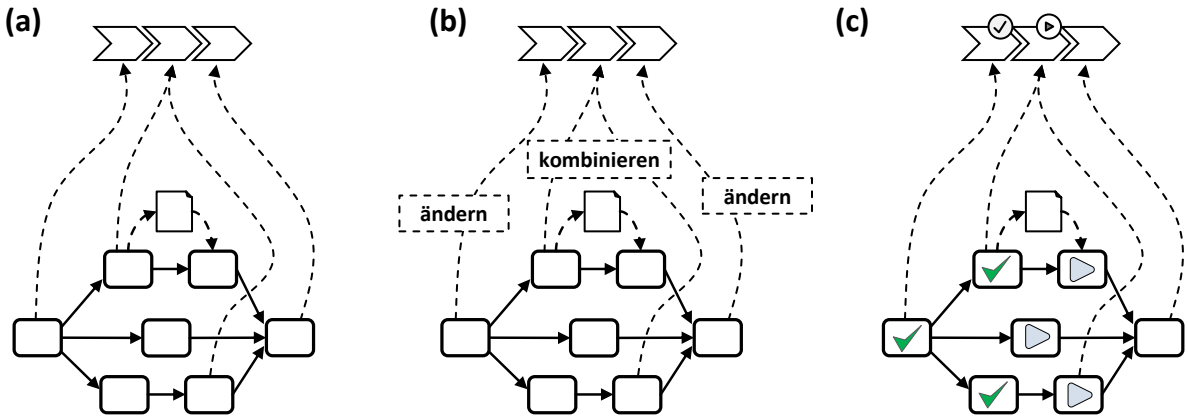


Abbildung 5.4: Skizze für (a) Zustandsübertragungen von einem PM-Graphen auf ein Prozessmodell in einer anderen Prozesssprache, (b) Zustandsübertragungsregeln und (c) die Durchführung der Zustandsprojektion.

Beispiele: Beim Monitoring von Geschäftsprozessen ist es Stand der Technik, einen Ausführungszustand in Form vorliegender Events zu aggregieren, auf ein beliebiges grafisches Modell abzubilden [ME07] oder auf ein Prozessmodell zu übertragen [Ora10]. Auch sind abstrahierende Sichten auf Prozessmodelle, die mit Ausführungsinformationen angereichert sind, lange bekannt [NSS01][Müh04]. Ferner bestehen umfassende Zustandsabbildungsfunktionen zur Ableitung des Zustands von aggregierten Strukturen [Bob08]. Außerdem liegen Arbeiten zur Prüfung der Zustandskonsistenz von mit Laufzeitinformationen angereicherten und abstrahierten Prozessmodellen vor [Bob08][SFW11]. Die von Schumm et al. [SLL+11] beschriebene Methode der Zustandsprojektion ist eine Erweiterung des Standes der Technik beim Monitoring von Geschäftsprozessen.

5.1.3 Rahmenbedingungen sprachübergreifender Sichten

Prozesssprachen haben mitunter eine unterschiedliche Ausdrucksstärke. Diese lässt sich etwa durch eine Workflow-Pattern-Analyse bestimmen und somit vergleichen [AHK+03]. Je stärker sich Prozesssprachen unterscheiden, desto schwieriger ist die Abbildung eines Prozesses in einer anderen Sprache. Ggf. kann nur eine unvollständige Abbildung erreicht werden, bspw. wenn eine Sprache keine beliebigen Kontrollflusszyklen unterstützt (engl. *arbitrary cycles*). Um lediglich ein Grundverständnis eines Prozesses zu vermitteln, ist eine vollständige Abbildung aber auch nicht erforderlich. Eine Lockerung der Semantik erlaubt den Übergang von einer Sprache in eine andere ohne komplexe Ersatzkonstrukte einsetzen zu müssen, welche bei unterschiedlicher Ausdrucksstärke erforderlich werden können. Bzgl. der Ausführung können sich Prozesssprachen nicht nur in ihrer Syntax und Semantik, sondern auch in den Zustandsräumen unterscheiden, die für die Aktivitäten und die Prozessinstanz gelten. Bei einer Übertragung von Zuständen über Sprachgrenzen hinweg müssen daher die Mengen möglicher Zustände aufeinander abgebildet werden.

Für prozessübersetzende Sichten kann die vorgestellte Sichtenklassifikation (Abschnitt 3.5) nicht verwendet werden, da sich ein Ausgabeprozess durch eine Übersetzung in jedem klassifizierenden Merkmal vom Eingabeprozess unterscheiden kann. Auch für instanzprojizierende Sichten ist die eingeführte Klassifikation nicht anwendbar, da sich das Prozess-

modell in der Zielsprache vom Eingabeprozess vollkommen unterscheiden kann. Für sprachemulierende Sichten findet die vorgestellte Klassifikation hingegen Anwendung.

5.2 Prozessübersetzende Sichten

Übersetzungsalgorithmen sind komplex und stellen ein eigenes Forschungsfeld dar. Davon unbelassen kann das Konzept der Prozessübersetzung an sich als gegeben betrachtet und durch das Hilfselement zur Prozessübersetzung (S1) im Sichtenentwurf eingesetzt werden, um Anforderungen an sprachenübergreifende Sichten zu formulieren. Für die Implementierung einer prozessübersetzenden Sicht ist die Bereitstellung einer Übersetzungsfunktion erforderlich, welche aus einem Eingabeprozess in einer Quellsprache einen Ausgabeprozess in einer Zielsprache erzeugt. Auf Basis bestehender Technologien und Ansätze ist dies für mehrere Sprachenpaare bereits möglich, jedoch nicht Umfang der vorliegenden Dissertation.

5.2.1 Prozess vereinfachen und anschließend übersetzen

Problem: Durch unterschiedliche Ausdrucksstärke von Sprachen sind vollumfänglich übersetzte Prozesse u.U. sehr komplex und können verschiedene Ersatzkonstrukte enthalten, welche die ursprüngliche Semantik weitgehend abbilden, jedoch schwer verständlich sind. Wie kann die aus einer Übersetzung resultierende Komplexität reduziert werden?

Lösungsansatz: Je nach Zweck der Übersetzung ist ein vollumfänglich übersetztes Prozessmodell ggf. gar nicht erforderlich. Der Lösungsansatz besteht darin, einen Prozess vor der Übersetzung auf das für einen bestimmten Verwendungszweck Wesentliche zu reduzieren und anschließend den vereinfachten Prozess zu übersetzen.

Die Vereinfachung wird vor der Übersetzung vorgenommen, wie die Mustersequenz in Abbildung 5.5 zeigt. Eine Vorgabe bei der Anwendung der Transformationen ist der Erhalt der Übersetzbarkeit (M29). Diese Vorgabe soll verhindern, dass Strukturen aus dem Prozess entfernt werden, die von dem Übersetzungsalgorithmus zwingend erwartet werden. Die Markierung relevanter Strukturen (M13) stellt den interaktiven Teil der Sicht dar. Die Markierung steuert das nachfolgende Entfernen von nicht als relevant markierten Aktivitäten (M1) und Datenelementen (M2). Die anschließende Bereinigung dient dem Entfernen nicht mehr benötigter Elemente (M8). Abschließend wird der vereinfachte Prozess übersetzt (S1). Die Aggregation bestimmter Strukturen (M2) vor einer Übersetzung stellt eine naheliegende Erweiterungsmöglichkeit dieses Entwurfs dar.

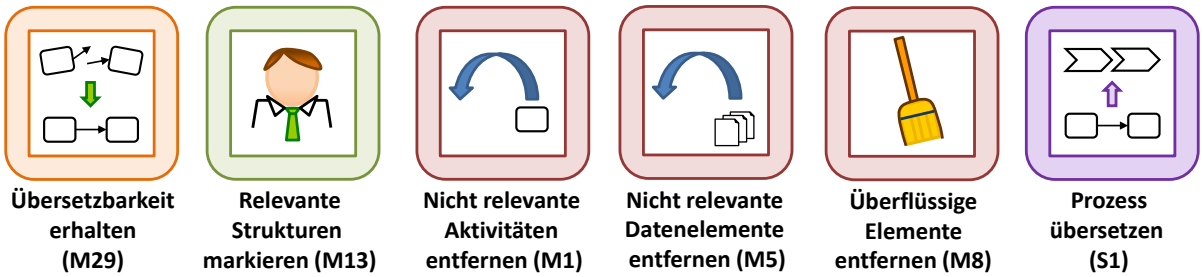


Abbildung 5.5: Übersetzung eines Prozessmodells mit vorangehender Vereinfachung.

Skizze: Die Skizze in Abbildung 5.6 zeigt einen Eingabeprozess, bei dem die für die Übersetzung relevanten Strukturen entsprechend markiert wurden (a). Die nicht als relevant gekennzeichneten Aktivitäten und Datenelemente werden anschließend entfernt (b). Danach wird der vereinfachte Prozess in die Zielsprache übersetzt (c).

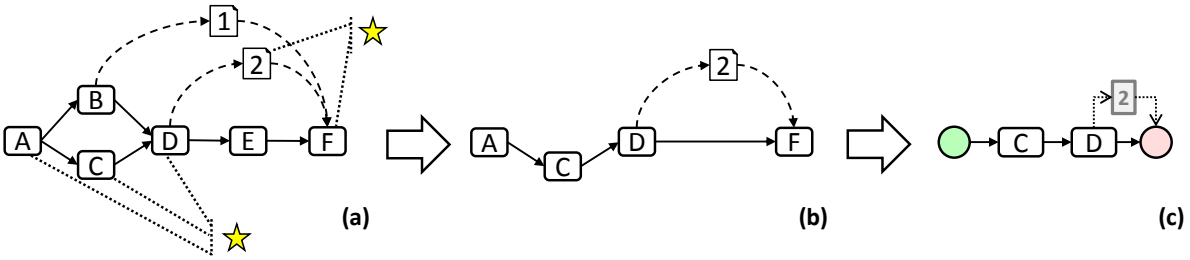


Abbildung 5.6: Skizze zur Übersetzung eines vereinfachten Prozessmodells.

Beispiele: Klai et al. [KTD09] setzen eine der Prozessübersetzung vorangehende Prozessabstraktion ein, um die anschließend auf dem übersetzten Prozess durchgeführte Prüfung auf das Vorhandensein von Deadlocks zu vereinfachen und zu beschleunigen.

5.2.2 Sichten auf übersetzte Prozesse

Problem: Anforderungen und Informationsbedarfe bzgl. der Prozessdarstellung gehen bei der Übersetzung verloren. Wie können diese Anforderungen und Informationsbedarfe trotz einer Prozessübersetzung berücksichtigt werden?

Lösungsansatz: Sichten können nicht nur vor, sondern auch nach der Übersetzung eines Prozesses eingesetzt werden, wie in der Mustersequenz in Abbildung 5.7 beispielhaft gezeigt. Das Segment zur Vereinfachung eines zu übersetzenden Prozesses (Abschnitt 5.2.1) ist zu einer Gruppierung zusammengefasst und aggregiert dargestellt. Nach der Vereinfachung wird der Prozess übersetzt (S1). Durch nachfolgende Musteranwendungen kann nach einer Prozessübersetzung bspw. der Datenfluss ausgeblendet werden (M5). Weiterhin könnte durch eine Anpassung des Erscheinungsbildes eine Hervorhebung aller von Menschen durchzuführenden Aktivitäten vorgenommen werden (M16).

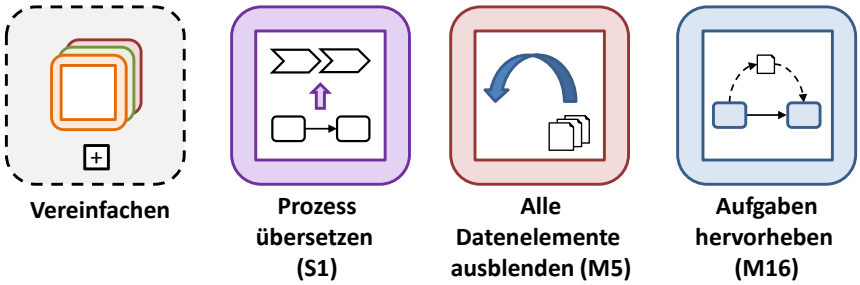


Abbildung 5.7: Auf übersetzte Prozessmodelle können weitere Muster angewendet werden.

Skizze: Die Skizze in Abbildung 5.8 zeigt, wie ein Eingabeprozess (a) nach dessen Vereinfachung und Übersetzung (b) durch die Anwendung weiterer Sichtentransformationen auf die Informationsbedarfe verschiedener Interessengruppen zugeschnitten werden kann (c).

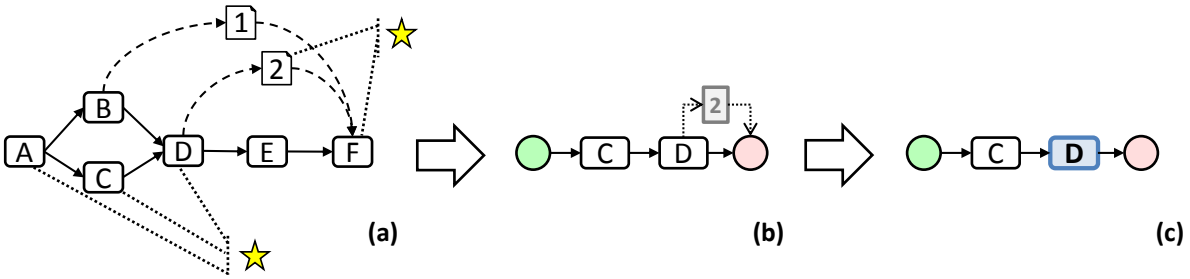


Abbildung 5.8: Skizze zur Anwendung einer Sicht auf einen vereinfachten und übersetzten Eingabeprozess.

Beispiele: Ein vereinfachter Prozess wird zur Prüfung des Vorhandenseins von Deadlocks übersetzt, der übersetzte Prozess geprüft und mit den Ergebnissen dieser Prüfung angereichert (M14). Die angereicherten Informationen werden anschließend zur Hervorhebung problematischer Ausführungspfade verwendet (M16, M17).

5.3 Sprachemulierende Sichten

Durch die Anwendung von Sichtentransformationen wird bei einer sprachemulierenden Sicht eine Zielsprache grafisch nachgeahmt. Die Anwendung von Sichtenentwurfsmustern mit Schwerpunkt auf der Darstellung vermittelt einem Betrachter den Eindruck, dass ein Prozess in einer bestimmten Sprache vorliegen würde. Ein neues, der Syntax der nur optisch emulierten Zielsprache entsprechendes, persistierbares Prozessmodell wird bei einer Sprachemulation *nicht* erzeugt. Eine Sprachemulation kann sowohl auf einem Modell als auch auf einem mit Laufzeitinformationen angereicherten Modell erfolgen. Eine Sprachemulation ist nicht notwendigerweise vollständig, konsistent, noch semantisch äquivalent zu dem Eingabeprozess, auf den sie angewendet wird. Der Zweck der Sprachemulation liegt in der Verbesserung der prozessbezogenen Kommunikation zwischen unterschiedlichen Interessengruppen.

Sprachemulationen sind im Stand der Technik an verschiedenen Stellen zu finden, wobei der in der vorliegenden Dissertation verwendete Begriff der Sprachemulation in diesem Zusammenhang nicht gebräuchlich ist, sondern diese Technik unter dem allgemeinen Begriff der Visualisierung zu finden ist. Mendling und Ziemann [MZ05] stellen bspw. eine Visualisierung von BPEL durch EPK vor. Schumm et al. [SKL+09] haben eine Sprachemulation von BPEL durch BPMN diskutiert, welche die grafische Darstellung von Aktivitäten, Kontrollfluss und Aspekte des Layouts umfasst, allerdings auch offenbart, dass eine Modellierung in einer sprachemulierenden Sicht nur eingeschränkt möglich ist. Ein vergleichbarer Ansatz wird in dem Modellierungswerkzeug NetBeans BPEL Designer [Net06] eingesetzt, Einschränkungen bestehen jedoch bei der Aktivität `<flow>`. Eine Sprachemulation der Prozesssprache ADEPT [DR09] durch BPMN wird von Kolb und Reichert [KR13b] demonstriert.

In den folgenden Abschnitten wird gezeigt, wie die Sprachemulation durch eine einfache Art der Darstellung die Komplexität von Geschäftsprozessen reduzieren und wie diese Sprachemulation unter Einsatz der Mustersprache beschrieben werden kann.

5.3.1 Ein Metamodell für Chevron-Diagramme

Die Ausführungen in diesem Abschnitt basieren auf einem technischen Bericht, in dem abstrakte *Chevron-Diagramme* zum Monitoring von BPEL Prozessen eingesetzt wurden, vgl. Schumm et al. [SKL+11]. Das Grundkonzept von Chevron basiert auf sogenannten Chevron-prozessen, die in dem Werkzeug Microsoft PowerPoint [Mic10] als sogenannte SmartArt zur Modellierung bzw. zur Darstellung von Prozessen verwendet werden können. Der Name leitet sich von der Form eines (Dach-)Sparrens ab (engl. *chevron*), welche zur Darstellung von Aktivitäten genutzt wird. Diese Form der Prozessbeschreibung und -darstellung wird in der Literatur und in der Geschäftswelt vielfach verwendet. Bspw. setzte bereits Fahrwinkel [Fah95] die Sparrenform zur Repräsentation von Geschäftsprozessen und Aktivitäten ein. Zudem hat Bobrik [Bob08] in einer vereinfachenden Sicht einen Richtungspfeil zur Darstellung aggregierter Strukturen verwendet. Auch in der vorliegenden Dissertation wird die Sparrenform an mehreren Stellen eingesetzt, vgl. bspw. Abbildung 3.1. Die Form erinnert an eine Wertschöpfungskette nach Porter [Por85], hat i.d.R. jedoch eine andere, allerdings nicht explizit festgelegte Semantik. Um diese Lücke der nicht festgelegten Semantik zu schließen, wird im Folgenden ein Metamodell vorgeschlagen, welches die Darstellungsform von Chevron-Diagrammen als abstrakte, nicht ausführbare Prozesssprache greifbar macht und einen Bezug zum Monitoring herstellt. Eine mögliche Form der Serialisierung von Prozessen in Chevron wird im Realisierungsteil diskutiert, vgl. dazu Abschnitt 9.2.4.

Metamodell. Das in Abbildung 5.9 dargestellte konzeptionelle Modell zeigt den Aufbau und die Bestandteile eines Prozessmodells in Chevron. Ein *Prozess* enthält mindestens eine *Phase*, wobei jede Phase eine beliebige Zahl an *Aktivitäten* enthalten kann. Jedes Element eines Prozesses in Chevron kann durch eine eindeutige Kennzeichnung identifiziert werden (ID). Des Weiteren besitzt jedes Element einen Anzeigenamen, eine Beschreibung und einen Status. In Chevron wird der Begriff *Status* anstatt *Zustand* verwendet, um die verschiedenen Zustandsräume von PI-Graphen und Chevron eindeutig unterscheiden zu können. Semantisch ist ein Status als Synonym zu einem Zustand zu verstehen. Die Menge der möglichen Status *S* ist für alle Elemente eines Prozesses in Chevron gleich, wobei *S* = {*Unbearbeitet, In Bearbeitung, Abgeschlossen*}.

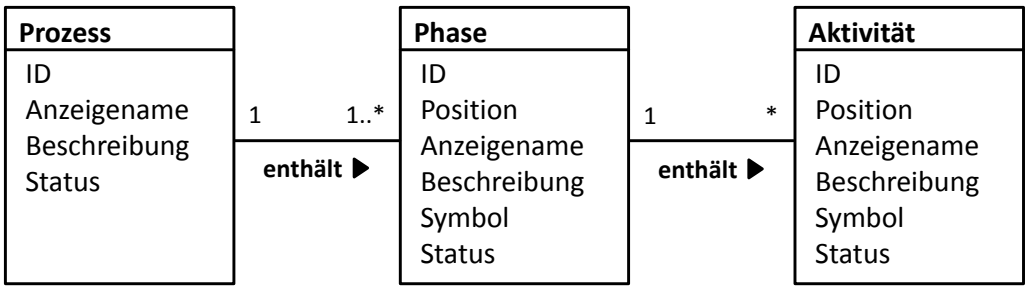


Abbildung 5.9: Konzeptionelles Modell von Chevron.

In Chevron gibt es weder Datenfluss noch Datenelemente und in Chevron bestehen keine Regeln zum Kontrollfluss. Das Merkmal der Position lässt sowohl bei Phasen wie auch bei Aktivitäten die Bildung einer Reihenfolge für die Darstellung zu. Die Position lässt allerdings nur einen eingeschränkten Rückschluss auf Kontrollabhängigkeiten zu. Chevron hat damit

eine bewusste Unschärfe. Diese Unschärfe ist zum einen ein Vorteil gegenüber komplexeren Prozesssprachen, da kein Wissen über Kontrollkonstrukte für das Prozessverständnis erforderlich ist. Allerdings kann diese Unschärfe auch als Nachteil ausgelegt werden, da jeder Betrachter eines Prozesses in Chevron ein anderes Verständnis von dem Prozess haben kann.

Erscheinungsbild. Die Möglichkeiten zur Anpassung des Erscheinungsbildes (M16) einer Phase und Aktivität in Chevron sind in Abbildung 5.10 dargestellt. Phasen und Aktivitäten können optional mit einem Symbol versehen sein, welches die Semantik der Phase oder Aktivität grafisch darstellt. Der Status wird durch ein statusrepräsentierendes Symbol dargestellt. Dadurch stehen die Rahmen- und Hintergrundfarbe für weitere Zwecke der Informationscodierung zur Verfügung, bspw. zur Hervorhebung. Die Beschreibung einer Aktivität oder Phase kann bspw. als Hinweisfenster (engl. *tooltip*) eingeblendet werden.

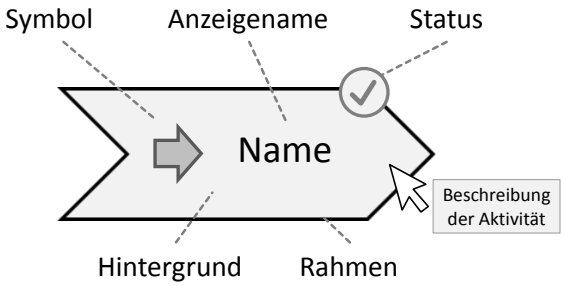


Abbildung 5.10: Möglichkeiten zur Anpassung des Erscheinungsbildes einer Phase oder Aktivität in Chevron, angepasst nach [SKL+11].

Layout. Das Layout eines Chevron-Diagramms erfolgt in einer Orientierung (M24) von links nach rechts für Phasen und von oben nach unten für Aktivitäten, aufsteigend nach dem Wert der Eigenschaft *Position*.

Beispiel. Abbildung 5.11 zeigt ein mit Laufzeitinformationen angereichertes Chevron-Diagramm. Es beinhaltet die Phasen *Bestellung*, *Produktion* und *Lieferung*. Die Phase *Produktion* enthält die Aktivitäten *Verfügbarkeit prüfen*, *Produkte herstellen* und *Produkte überprüfen*. Jede Phase ist mit einem Symbol und einem Anzeigenamen dargestellt. Aktivitäten sind nur mit einem Anzeigenamen dargestellt. Die ersten beiden Phasen sind durch ein Statussymbol als *Abgeschlossen* gekennzeichnet, während sich die dritte Phase *In Bearbeitung* befindet. Die Aktivitäten *Produkte herstellen* und *Produkte überprüfen* haben den Status *Unbearbeitet* und tragen in diesem Beispiel kein Statussymbol. Eine Aussage, warum diese Aktivitäten nicht ausgeführt wurden, ist nicht dargestellt und liegt im Interpretationsspielraum: die Aktivität *Verfügbarkeit prüfen* könnte ergeben haben, dass die bestellten Produkte vorrätig waren und sie direkt ausgeliefert werden konnten.

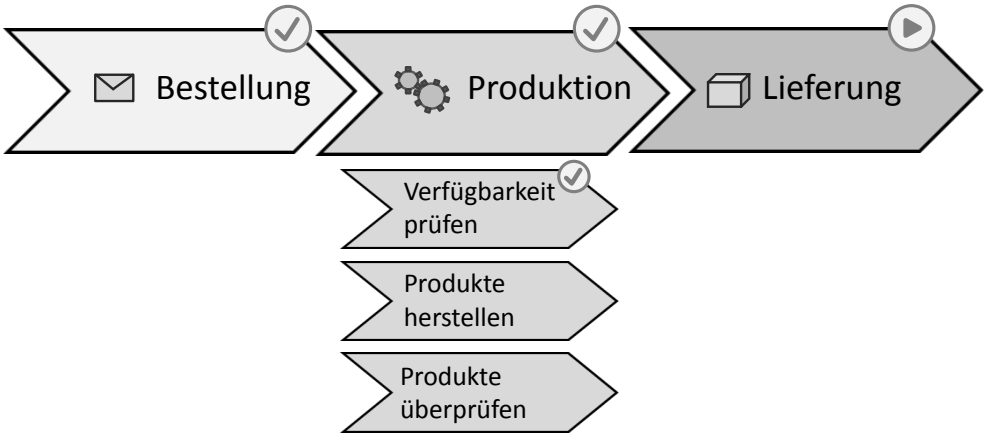


Abbildung 5.11: Darstellung eines Prozesses in Chevron mit drei Phasen, angepasst nach [SKL+11].

Zustandsmodell. Für Chevron kann ein einfaches und abstraktes Zustandsmodell für Instanzen, Phasen und Aktivitäten definiert werden, das deren Lebenszyklus beschreibt. Wie in Abbildung 5.12 dargestellt, lässt das hier vorgeschlagene Zustandsmodell die Bedingungen zum Zustandsübergang offen.

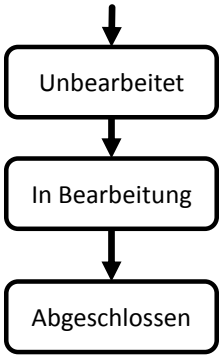


Abbildung 5.12: Zustandsmodell für Instanzen, Phasen und Aktivitäten.

Anpassungsmöglichkeiten des Metamodells. Mit Ausnahme des konzeptionellen Modells, dem Grundprinzip des Erscheinungsbildes und dem Layout von Phasen und darin enthaltenen Aktivitäten kann Chevron beliebig angepasst, d.h. erweitert oder auch eingeschränkt werden.

Potentielle Erweiterungen: Zusätzliche Eigenschaften wie bspw. die Start- und Stopzeit einer Ausführung, der Bearbeiter einer Aktivität oder der Verantwortliche einer Phase könnten angegeben werden. Außerdem bietet sich die Möglichkeit, Phasen mit enthaltenen Aktivitäten zu aggregieren (M2).

Potentielle Einschränkungen: Um die maximale Komplexität eines Prozesses in Chevron zu begrenzen, könnte die Anzahl an Phasen und der darin enthaltenen Aktivitäten limitiert werden. Auch eine Zeichenbegrenzung bei der Benennung von Phasen ist eine naheliegende Einschränkungsmöglichkeit von Chevron.

5.3.2 Sprachemulation von Prozessmodellen durch Chevron

Problem: Jede Prozesssprache hat eine eigene Semantik, unterschiedliche Symbole und kann unterschiedlich strukturiert sein, bspw. auf Basis eines Graphen oder auf Basis von Regeln. Selbst ein vereinfachter Prozess kann kaum verständlich sein, wenn die darin verwendeten Symbole und Pfeilformen dem Betrachter nicht bekannt sind. Aber wie kann ein Prozess in einer anderen Prozesssprache dargestellt werden, ohne ihn vollumfänglich übersetzen zu müssen?

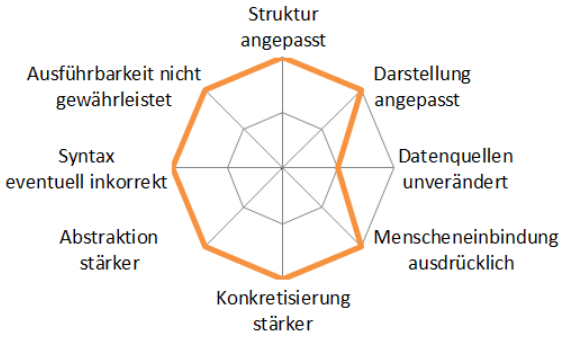


Abbildung 5.13: Klassifikation des Entwurfs.

Lösungsansatz: Die Semantik von Chevron ist im Vergleich zu komplexen Prozesssprachen wie BPEL oder BPMN leicht zu erlernen und es gibt in Chevron nur einen Symboltyp, der einem Betrachter bekannt sein muss. Eine Sprachemulation durch Chevron bietet daher eine Möglichkeit zur Prozesskommunikation, die von der Prozesssprache des Eingabeprozesses und von deren Komplexität abstrahiert. Für die Emulation werden relevante Aktivitäten eines Eingabeprozesses manuell zu Phasen zugeordnet. Diese Zuordnung unterstützt die nachgelagerte Anwendung des „teile und herrsche“-Prinzips. Eine treffende Benennung und Bebilderung einer Phase nach Vorgaben eines Nutzers vermeidet das Labeling Problem bei der Aggregation von Strukturen und gewährleistet, dass die Phasen auf den Kontext und den Inhalt eines Prozesses abgestimmt sind.

Die Mustersequenz in Abbildung 5.14 beschreibt, wie ein Prozess durch die Anwendung von Sichtenentwurfsmustern durch Chevron emuliert werden kann. Zunächst weist ein Nutzer als relevant wahrgenommene Aktivitäten verschiedenen Phasen zu (M13). Für verschiedene Arbeitsgebiete könnte die Menge an geeigneten Phasen vordefiniert werden. Für Prozesse in der Simulationsdomäne sind etwa die Phasen *Pre-Processing*, *Simulation* und *Post-Processing* gängig [SGK+11]; der Prozess einer Filmproduktion besteht bspw. aus den Phasen *Development*, *Pre-Production*, *Production* und *Post-Production* [SRH+06]. Optional können kurze Bezeichnungen an die so gekennzeichneten Aktivitäten annotiert werden, die anstatt der normalen Aktivitätsbezeichnungen angezeigt werden sollen. Anschließend werden die Aktivitäten im Prozess automatisch durchnummeriert und darzustellende Phasen auf Basis der annotierten Informationen bestimmt (M14). Die Nummerierung von Aktivitäten kann bspw. nach einer Breitensuche im Prozessgraphen erfolgen und die Phasennummerierung nach Reihenfolge des ersten Auftretens während der Aktivitätsnummerierung erfolgen.

Ein so annotierter Prozess wird anschließend vereinfacht, indem alle Datenelemente (M5) und alle Aktivitäten ohne Phasenzugehörigkeit (M1) entfernt werden. Zur Darstellung der Phasen in Chevron werden als Phase gekennzeichnete Aktivitäten in den Prozess eingefügt (M9). Da Chevron kein Kontrollflusskonzept besitzt, ist eine Integration in den bestehenden Kontrollfluss nicht erforderlich. Zur Reduzierung der Komplexität werden anschließend Aktivitäten von einelementigen Phasen entfernt (M1). Kontrollverbindungen werden für die Darstellung in Chevron ausgeblendet (M17). Aktivitäten und als Phase gekennzeichnete

Aktivitäten werden in Sparrenform dargestellt, wobei normale Aktivitäten eine kleinere Form annehmen. Das Layout erfolgt in maßgeschneiderter Form nach Chevron-Vorgabe (M28), wodurch Phasen nach Position aufsteigend sortiert von links nach rechts ausgelegt werden. Ihnen zugeordnete Aktivitäten werden aufsteigend sortiert von oben nach unten unterhalb der jeweiligen Phase ausgelegt.

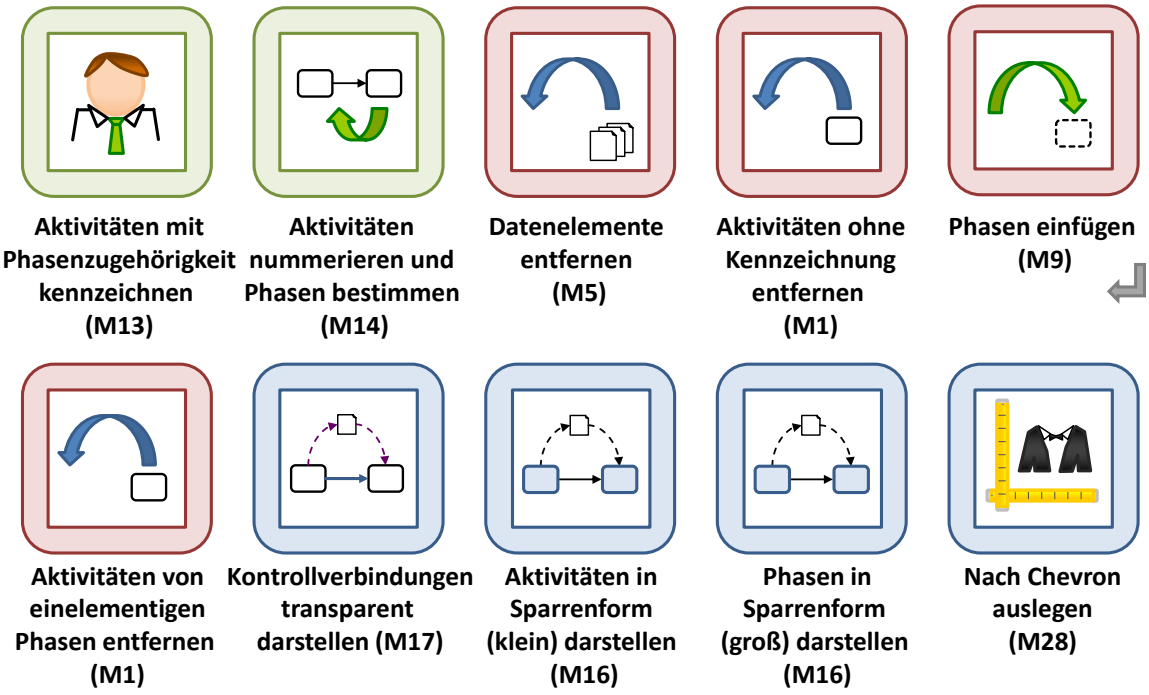


Abbildung 5.14: Mustersequenz für eine Sprachemulation durch Chevron.

Skizze: Die Skizze in Abbildung 5.15 (a) zeigt einen mit drei Phasen (*A*, *B*, *C*) gekennzeichneten Eingabeprozess, der durch verschiedene Transformationen der Chevron-Darstellung nachempfunden wurde. In Abbildung 5.15 (b) sind die Aktivitäten des Eingabeprozesses nummeriert worden. Auch wurden alle Datenelemente sowie die nicht gekennzeichneten Aktivitäten *X*, *Y* und *Z* entfernt. Für jede Phase wurde eine Aktivität mit der Phasenbezeichnung in den Prozess eingefügt (*A*, *B*, *C*). In Abbildung 5.15 (c) wurden die Aktivitäten *b1* und *c1* der einelementigen Phasen *B* und *C* entfernt, Kontrollverbindungen ausgeblendet und das Layout nach Chevron vorbereitet. In Abbildung 5.15 (d) wurde das Erscheinungsbild der Aktivitäten angepasst und die Sprachemulation durch Chevron damit abgeschlossen.

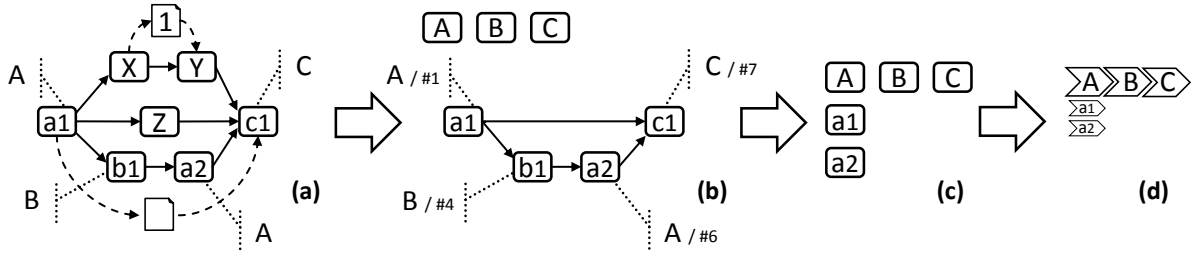


Abbildung 5.15: Skizze für eine für eine Sprachemulation durch Chevron.

Diskussion: Über die Lebensdauer eines Geschäftsprozesses kommen viele verschiedene Interessengruppen mit dem Prozess in Berührung, wie z.B. Sachbearbeiter, Analysten und Betriebsprüfer. Mit einer Sprachemulation durch Chevron kann jede dieser Interessengruppen darin unterstützt werden, rasch ein Grundverständnis über einen Prozess zu erlangen – unabhängig davon, in welcher Prozesssprache der Prozess tatsächlich vorliegt. Während die bisher vorgestellten Sichtenentwürfe eine graphbasierte Prozesssprache vorausgesetzt haben, kann die Sprachemulation durch Chevron auch auf nicht graphbasierte Sprachen angewendet werden. Die einzige Voraussetzung ist, dass ein durch Chevron zu emulierender Prozess Aktivitäten enthält. Die Chevron-Sprachemulation ließe sich bspw. auch auf Prozesse in ConDec [PA06] anwenden, einer deklarativen Prozesssprache, in der Constraints zwischen Aktivitäten definiert werden, um so die möglichen Ausführungsreihenfolgen einzuschränken und das Ablaufverhalten vorzugeben. Für eine Darstellung in Chevron sind lediglich mit Phasenkennzeichnungen annotierte Aktivitäten erforderlich. Allerdings erfordert die Bestimmung der Reihenfolge der Phasen und Aktivitäten eine Analyse der Prozesssemantik, sofern deren Position nicht manuell annotiert werden soll (M13).

Bobrik [Bob08] schlägt eine sehr ähnliche Sicht vor, um einen schnellen Überblick über einen Prozess oder eine Instanz zu bieten: Als Basis dient ein bereits vollständig in Phasen unterteilter Prozess (M25). Die Aktivitäten werden entsprechend der Phasenzugehörigkeit aggregiert (M3) und die phasenrepräsentierenden aggregierten Strukturen in Form eines Richtungspfeil dargestellt (M16) und nebeneinander von links nach rechts ausgelegt (M24). Laufzeitinformationen (M15) werden bei der Aggregation berücksichtigt und der aggregierte Zustand wird durch ein Symbol auf dem Richtungspfeil dargestellt (M16).

Dyer et al. [DHL+12] beschreiben eine Methode zur Modellierung und stufenweisen Verfeinerung eines Prozessmodells. Die Modellierung beginnt mit einer sogenannten Discovery Map, in der Milestones definiert werden und diesen Milestones Aktivitäten zugeordnet werden. Die Milestones sind von links nach rechts ausgelegt und die ihnen zugeordneten Aktivitäten sind unterhalb der Milestones platziert, jeweils von oben nach unten. Diese Menge an Aktivitäten wird anschließend in einer verfeinernden Sicht von einem Nutzer organisatorisch in Swimlanes angeordnet, der Kontrollfluss wird festgelegt und weitere Aktivitäten werden hinzugefügt. Die vorgestellte Sicht zur Sprachemulation durch Chevron kann analog zu dieser Vorgehensweise verstanden werden, wobei die Sprachemulation durch Chevron in umgekehrter Richtung vorgeht, vom Konkreten zum Abstrakten.

5.3.3 Sprachemulation von Prozessinstanzen durch Chevron

Problem: Die Kommunikation über Prozesse ist nicht nur für deren Modelle relevant, sondern sollte auch für Instanzen dieser Modelle möglich sein. Allerdings haben unterschiedliche Prozesssprachen verschiedene und mitunter komplexe Zustandsmodelle. Wie können diese Zustandsmodelle in eine Sprachemulation einbezogen werden?

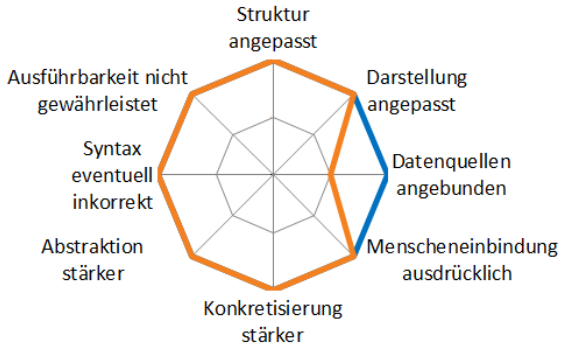


Abbildung 5.16: Klassifikation des Entwurfs.

Lösungsansatz: Die Sprachemulation von komplexen Prozessen durch Chevron wird als Ausgangsbasis verwendet und um Musteranwendungen erweitert, welche den Laufzeitaspekt einbringen, vgl. die Änderung der Klassifikation in Abbildung 5.16, hervorgehoben in Blau. Die erforderlichen Erweiterungen sind in Abbildung 5.17 aufgeführt.

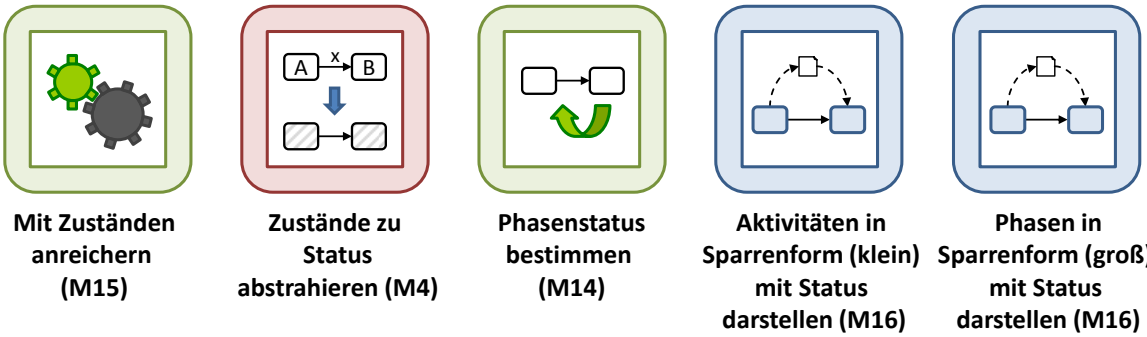


Abbildung 5.17: Erforderliche Erweiterungen für die Sprachemulation einer Prozessinstanz mit Chevron.

Grundlage des Laufzeitaspekts ist eine Anreichung des Eingabeprozesses mit Informationen zu den Zuständen von Aktivitäten (M15). Zur Nutzung des einfachen Zustandsmodells von Chevron ist eine Abstraktion des Zustands zu Chevron-Status erforderlich (M4). Für die in Definition 2 zum PI-Graphen angegebenen Aktivitätszustände (vgl. Abschnitt 2.1.4) kann eine Abstraktion des Zustands einer Aktivität wie folgt vorgenommen werden:

- Wenn der Zustand der Aktivität *Inaktiv* oder *Übersprungen* ist, gilt der Status *Unbearbeitet* und
- wenn der Zustand der Aktivität *Erledigt* ist, gilt der Status *Abgeschlossen*,
- ansonsten gilt der Status *In Bearbeitung*.

Dadurch werden die Aktivitätszustände *Bereit*, *Aktiv*, *Unterbrochen* und *Fehlerhaft* eines PI-Graphen gleichermaßen auf den Chevron-Status *In Bearbeitung* abgebildet.

Der Phasenstatus (M14) könnte auf Basis der ihr zugeordneten Aktivitäten bestimmt werden:

- Wenn alle der Phase zugeordneten Aktivitäten den Status *Abgeschlossen* haben, wird der Status der Phase auf *Abgeschlossen* gesetzt,

- Wenn alle der Phase zugeordneten Aktivitäten den Status *Unbearbeitet* haben, wird der Status der Phase auf *Unbearbeitet* gesetzt und
- ansonsten wird der Status der Phase auf *In Bearbeitung* gesetzt.

Eine Phase wird demnach nur dann als *Abgeschlossen* betrachtet, wenn alle darin enthaltenen Aktivitäten den Status *Abgeschlossen* haben. Andere Möglichkeiten zur Bestimmung des Status einer Phase sind aber durchaus denkbar.

Das letzte Segment des Sichtenentwurfs ist die Anpassung des Erscheinungsbildes (M16), um die Status mit einem entsprechenden Symbol darzustellen.

Abbildung 5.18 zeigt die Erweiterung der Mustersequenz zur Sprachemulation mit Chevron aus Abschnitt 5.3.2 mit den Musteranwendungen zur Unterstützung des Laufzeitaspekts.

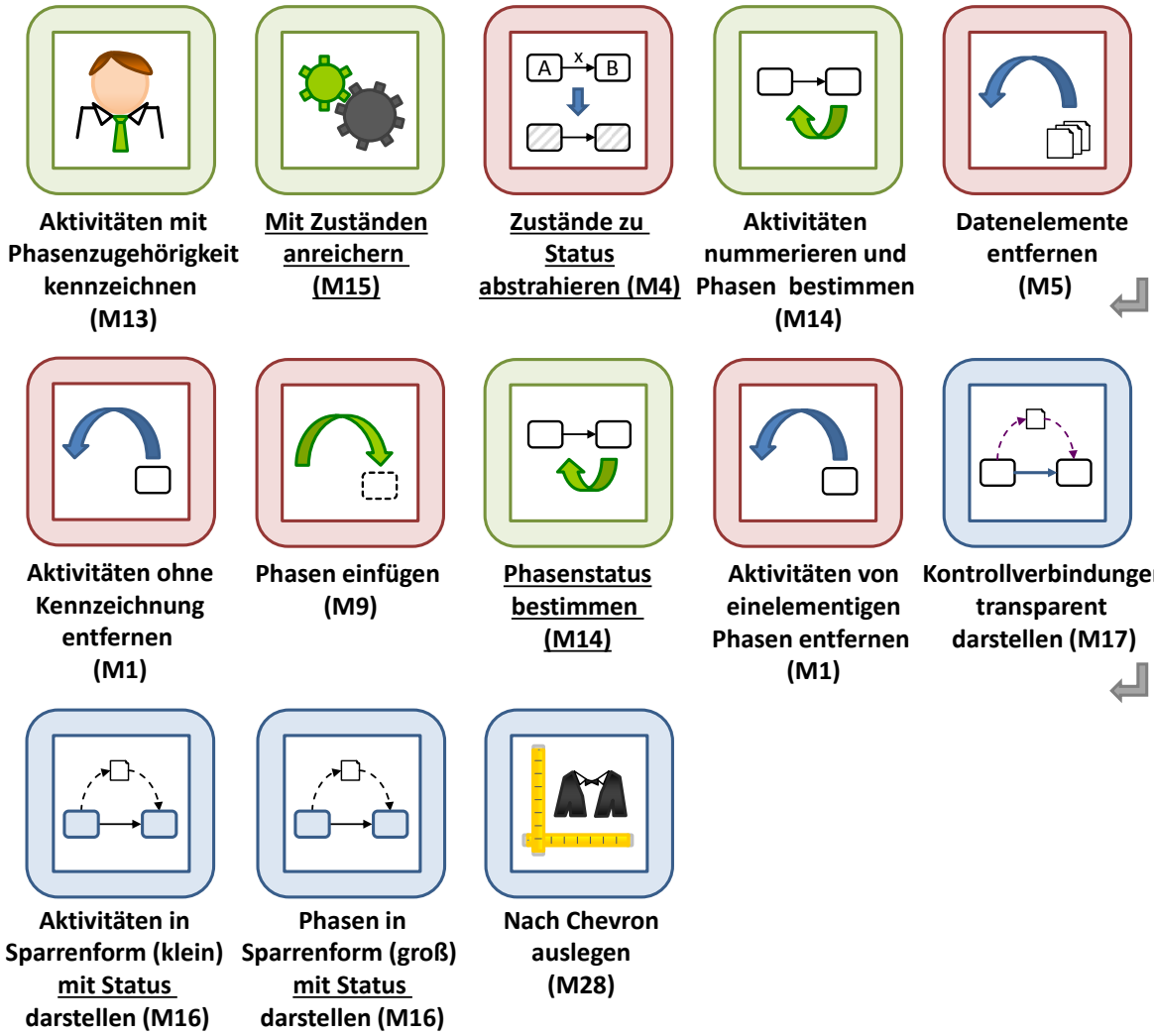


Abbildung 5.18: Mustersequenz für die Sprachemulation einer Prozessinstanz durch Chevron.

Die Anreicherung mit Zustandsinformationen (M15) und deren Abstraktion zu Chevron-Status (M4) kann unmittelbar nach der manuellen Kennzeichnung erfolgen und so den Beginn der erweiterten Mustersequenz bilden. Die Bestimmung der Status von Phasen (M14) erfolgt nach deren Einfügen in den Prozess (M9) und vor der Entfernung von Aktivitäten aus

einelementigen Phasen (M1). Die Anpassungen am Erscheinungsbild (M16) wurden sinngemäß zusammengeführt.

Skizze: Die Skizze in Abbildung 5.19 zeigt, wie sich die Erweiterung um Laufzeitinformationen (a) auf die Sprachemulation durch Chevron auswirkt. In Abbildung 5.19 (b) sind die Aktivitätszustände zu Chevron-Status abstrahiert. Aus diesen Status werden anschließend die Status der Phasen bestimmt (c). Das Erscheinungsbild der Aktivitäten und Phasen ist abschließend um ein Symbol für den Status erweitert worden (d).

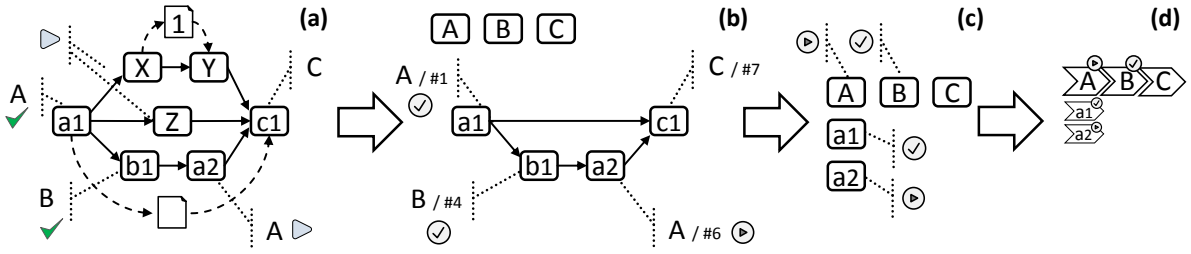


Abbildung 5.19: Skizze für eine Sprachemulation einer Prozessinstanz mit Chevron.

Diskussion: Die Skizze in Abbildung 5.19 offenbart ein Merkmal der Sprachemulation von Prozessinstanzen durch Chevron: Da Aktivität *b1* die einzige Aktivität von Phase *B* ist und nach *b1* eine weitere, noch *In Bearbeitung* befindliche Aktivität von Phase *A* folgt, ist die Phase *B* bereits *Abgeschlossen*, obwohl die vorangehende Phase *In Bearbeitung* ist. Dieses Phänomen, das auch als Zustandsinkonsistenz diskutiert wurde [Bob08] kann auftreten, wenn die Aktivitäten eines Prozess in nicht überlappungsfreie Phasen unterteilt werden.

Ein weiterer Aspekt dieser Sprachemulation ist die Modifikation der Semantik hinsichtlich der Ausnahmezustände *Unterbrochen* und *Fehlerhaft*, welche beide auf *In Bearbeitung* abgebildet wurden. Eine Unterscheidung von einer normalen Aktivitätsbearbeitung ist für den Betrachter so unmöglich. Die Festlegung einer geeigneten Zustandsabstraktion wird als Aufgabe des Feinentwurfs betrachtet. Im Feinentwurf müssen demzufolge prozess- oder domänenspezifische Statements wie „Fehlerhaftes wird nachgeholt und es bleibt nichts liegen“ interpretiert und in eine entsprechende Abbildung umgesetzt werden.

5.4 Instanzprojizierende Sichten

Zunächst wird das Prinzip der Instanzprojektion vertieft und in den musterbasierten Sichtenentwurf integriert. Anschließend werden verschiedene Formen der Zustandsübertragung vorgestellt, die bei der Analyse verschiedener Szenarien identifiziert wurden. Die Spezifikation von Zustandsübertragungsregeln und eine Vorgehensweise zur Durchführung einer Zustandsprojektion folgen im Realisierungsteil (Abschnitt 9.2).

5.4.1 Prinzip der Instanzprojektion

Bei der Projektion von Zuständen konnten durch das Studium verschiedener Szenarien wiederkehrende Formen der Zustandsübertragung identifiziert und als Muster zur

Zustandsübertragung beschrieben werden (engl. *state propagation patterns*). Durch deren Extraktion aus ihrem Kontext können diese Muster fortan auch in anderen Szenarien eingesetzt werden. Diese Muster sind nicht auf der Ebene der Sichtenentwurfsmuster des vorgestellten Musterkatalogs angesiedelt, sondern sie stellen feingranulare Techniken dar, die in der Anwendung des Hilfselements der Zustandsprojektion (S2) zum Tragen kommen.

Anwendungsszenarien. Es wurden zwei Szenarien untersucht, in denen der Bedarf nach einer Zustandsprojektion bestand, vgl. [SLL+11]. Diese Szenarien werden hier vorgestellt:

1. decidR [Stu10] ist eine Plattform zur einfachen Modellierung und Ausführung von Workflows. Diese Workflows werden in einer einfachen Sprache modelliert, der decidR Workflow Definition Language (DWDL). In DWDL können vordefinierte Aktivitätstypen verwendet werden, bei denen Details verborgen werden. So kann ein Nutzer eine Aktivität zum Versand einer E-Mail in einen Prozess einfügen. Das abstrakte Prozessmodell in DWDL wird zur Ausführung nach vordefinierten Regeln in einen BPEL-Prozess transformiert. Eine Aktivität zum Versand einer E-Mail wird bspw. in eine Sequenz transformiert, welche aus einer Aktivität zur Vorbereitung einer Nachricht, zum Aufruf eines Dienstes und zur Erfassung der Rückmeldung des Dienstaufrufes besteht. Allerdings sind Nutzer von DWDL nicht an der Prozessausführung auf Basis des komplexen Prozessmodells in BPEL interessiert, sondern nur am Zustand in DWDL.
2. Mit Mustern zur Integration von Unternehmensanwendungen (engl. *Enterprise Application Integration, EAI*) nach Hohpe und Woolf [HW03] können komplexe Integrationsszenarien durch Einsatz bewährter Techniken entworfen werden. Scheibler et al. [SML09] stellen mit „Parametrierbaren EAI Patterns“ (PEP) eine Erweiterung vor, mit der die Muster als Ausgangsmodell einer Codegenerierung verwendet werden können. Mit dem Werkzeug Generating Enterprise Integration Executable Scenarios (GENIUS) [SL09] besteht eine grafische Modellierungsoberfläche für PEP-Musterkompositionen. Nach dem Prinzip der modellgetriebenen Softwareentwicklung werden diese Modelle in ausführbaren Code transformiert. Als Ausführungsplattform unterstützt GENIUS bspw. BPEL-Umgebungen, wobei Modellelemente in PEP zu komplexen Strukturen in BPEL transformiert werden. Auch in diesem Szenario ist das Monitoring lediglich auf BPEL-Ebene möglich, welche aber nicht die Arbeitsebene des Modellierers darstellt. Eine Besonderheit an diesem Szenario ist, dass das Verhalten der Ausführung in BPEL teilweise eine andere Semantik auf Ebene von PEP hat. Ein Fehlerzustand in BPEL kann auf Basis von PEP mitunter ein normales Verhalten darstellen. Diese Besonderheit entsteht durch die unterschiedliche Ausdrucksstärke der beiden Sprachen, welche den Einsatz von Ersatzkonstrukten bei der Übersetzung von PEP zu BPEL bedingt.

Möglichkeiten der Instanzprojektion. Beide Szenarien wurden analysiert, um daraus Muster für die Übertragung von Zuständen von konkret-technischen zu abstrakt-fachlichen Modellen zu gewinnen. Die Szenarien haben gemein, dass zunächst ein Prozess fachlich modelliert wird und daraus ein ausführbarer Prozess generiert wird. Eine Zustandsprojektion ist insofern eine Art Rückführung zum Ursprung. Der Prozess, auf den ein Ausführungszustand projiziert wird, kann jedoch ebenso ein fachliches Prozessmodell darstellen, welches

im Zuge der Verfeinerung durch die IT manuell in ein ausführbares Modell überführt wurde. Das Ziel einer Projektion kann aber auch frei nach individuellen Informationsbedarfen modelliert werden. Dadurch wird das Monitoring eines komplexen Prozesses in interessengruppenspezifischer Sprache und individuellem Prozessverständnis möglich. Eine Äquivalenz des tatsächlich ausgeführten Prozesses und des Ziels der Projektion ist weder semantisch noch hinsichtlich der Prozesssprache erforderlich, um eine Projektion anwenden zu können. Es ist darüber hinaus nicht einmal erforderlich, dass das Ziel einer Projektion ein vollständiger Prozess ist, auch das Monitoring eines Prozessfragments ist damit möglich.

Datenprojektion. Eine Instanzprojektion umfasst eine Zustandsprojektion und eine Datenprojektion. Bei einer Datenprojektion werden Laufzeitwerte von Datenelementen auf Datenelemente eines anderen Prozessmodells übertragen. In einer Datenprojektion kommen Konzepte zur Format- und Typumwandlung (engl. *type casting*), wie auch Datenabfragesprachen wie bspw. SQL oder XQuery[5] zum Tragen. Aspekte einer Datenprojektion treten außerdem im Kontext der Nachrichtentransformation auf, die in der Integration von Anwendungssystemen eine wichtige Rolle spielt. Auf diesem Gebiet haben Hohpe und Woolf [HW03] verschiedene Muster identifiziert, die zum Entwurf einer Datenprojektion herangezogen werden können: Das Muster *Aggregator* beschreibt die Verbindung verschiedener, aber in Bezug stehender Nachrichten; *Splitter* beschreibt die Zerlegung einer Nachricht in mehrere Teile, die unabhängig voneinander verarbeitet werden können; *Content Filter* beschreibt die Reduzierung eines Datenelements auf einzelne Bestandteile; *Message Translator* beschreibt die Übersetzung einer Nachricht in ein anderes Datenformat.

In Anbetracht der bereits bestehenden Möglichkeiten, eine Datenprojektion abstrakt zu entwerfen und technisch zu implementieren, wird dieser Aspekt im Rahmen dieser Arbeit nicht weiterverfolgt, sondern der Fokus stattdessen auf die noch zu erschließende Zustandsprojektion gelegt. Zudem ist eine Datenprojektion für Chevron aus Mangel an Datenelementen im Metamodell von Chevron nicht anwendbar.

Zustandsprojektion im musterbasierten Sichtenentwurf. Abbildung 5.20 nimmt eine Einordung des Ansatzes der Zustandsprojektion in den Kontext des musterbasierten Sichtenentwurfs vor. Im Vorfeld einer Zustandsprojektion müssen die anzuwendenden Zustandsübertragungsregeln festgelegt werden (M13). Für die Durchführung einer Zustandsprojektion wird ein Eingabeprozess in einer Quellsprache mit Laufzeitinformationen angereichert (M15). Der Ausgabeprozess dieser Transformation stellt die Informationsquelle für die Abbildung von Zuständen auf das Zielmodell dar.

Die Anwendung einer Zustandsprojektion (S2) wertet festgelegte Zustandsübertragungsregeln aus und reichert das Zielmodell mit Laufzeitinformationen an.

[5] W3C: An XML Query Language (XQuery)

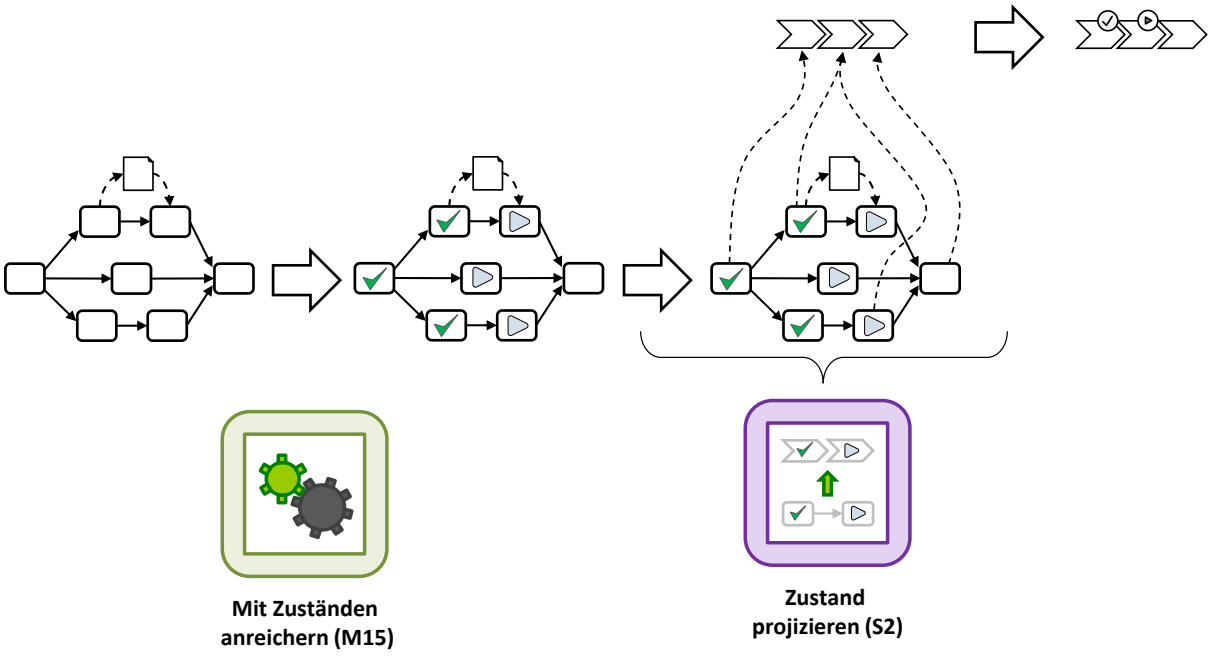


Abbildung 5.20: Mustersequenz für eine Zustandsprojektion mit beispielhafter Darstellung von Eingabeprozess und Ausgabeprozess.

5.4.2 Muster zur Zustandsübertragung

Die nachfolgenden Musterbeschreibungen, Beispiele und Skizzen basieren auf den von Schumm et al. [SLL+11] diskutierten Zustandsübertragungsmustern. Diese Muster beschreiben die grundlegenden Prinzipien zum Entwurf einer Zustandsprojektion.

Zustand direkt übertragen (Z1).

Abbildung 5.21: Zustand direkt übertragen (Z1).

Problem: Wie kann ein Zustand übertragen werden, wenn genau eine Aktivität aus einem ausführungsnahen Prozess genau einer Aktivität des Prozesses entspricht, auf den der Ausführungszustand projiziert werden soll?

Beschreibung: Bei der direkten Übertragung eines Zustands wird der Zustand ohne Änderung übertragen (bijektive Abbildung). Die direkte Übertragung eignet sich daher weniger für sprachübergreifende, sondern mehr für modellübergreifende Projektionen. Die Skizze in Abbildung 5.21 zeigt eine direkte Übertragung des PI-Graph-Zustands *Erledigt* auf eine Aktivität in Chevron-Darstellung ohne Änderung des Zustands und folglich ohne Berücksichtigung des Zustandsmodells der Zielsprache.

Beispiel: In einem fachlichen Prozessmodell in BPMN wird eine Aktivität als Task modelliert, aber nach der Implementierung des Prozesses in BPEL als `<invoke>` Aktivität in einer BPEL-basierten Umgebung ausgeführt. Die direkte Übertragung der `<invoke>`

Aktivität zu dem BPMN Task zeigt einem Nutzer den tatsächlichen Ausführungszustand auf Basis des ursprünglichen, fachlichen Prozessmodells an.

Zustand ändern (Z2).

Abbildung 5.22: Zustand ändern (Z2).

Problem: Wie kann ein Zustand übertragen werden, wenn sich Aktivitäten direkt entsprechen, aber die Mengen möglicher Zustände von Quellsprache und Zielsprache unterscheiden?

Beschreibung: Wenn sich die Mengen möglicher Zustände von Quellsprache und Zielsprache unterscheiden, dann ist eine Änderung des Zustands erforderlich. Die Änderung des Zustands erlaubt außerdem das Monitoring des Zustands einer Aktivität, bei der der Zustand im ausführungsnahen Prozessmodell und einem fachlichen Modell eine andere Bedeutung haben. Die Skizze in Abbildung 5.22 zeigt eine Änderung des PI-Graph-Zustands *Fehlerhaft* in den Chevron-Status *In Bearbeitung*.

Beispiel: Durch eingeschränkte Möglichkeiten zur Modellierung in einem ausführungsnahen Prozessmodell müssen ggf. Ersatzkonstrukte eingesetzt werden. Der Zustand dieser Ersatzkonstrukte hat auf fachlicher Ebene eine andere Bedeutung als in der tatsächlichen Ausführung. Ein Fehler, der während einer Ausführung in geplanter Form auftritt und abgefangen wird, könnte bspw. in dem fachlichen Modell als planmäßige Ausführung dargestellt werden.

Zustände kombinieren (Z3).

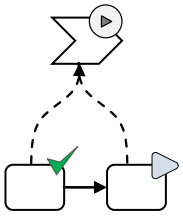


Abbildung 5.23: Zustände kombinieren (Z3).

Problem: Wie kann ein Zustand übertragen werden, wenn das ausführungsnahe Modell einen höheren Detaillierungsgrad aufweist?

Beschreibung: Wenn eine Aktivität in einem fachlichen Modell durch mehrere, nicht notwendigerweise direkt verbundene Aktivitäten in einem ausführungsnahen Modell repräsentiert ist, müssen mehrere Zustände kombiniert werden. Für die Kombination von Zuständen können beliebige Abbildungsfunktionen eingesetzt werden. Die Skizze in Abbildung 5.23 zeigt eine einfache Kombination der PI-Graph-Zustände *Erledigt* und *Aktiv* in den Chevron-Status *In Bearbeitung*. Die Kombination in dieser Skizze beinhaltet eine Änderung des Zustands aufgrund unterschiedlicher Zustandsmodelle in Quell- und Zielsprache (vgl. Z2).

Beispiel: Eine einfache Abfrage-Aktivität in einem fachlichen Prozessmodell erfordert in einer Implementierung auf Basis einer Dienstorchestrierung verschiedene Dienstaufrufe, um unterschiedliche Datenquellen abzufragen. Die Dienstaufrufe werden in BPEL durch parallel ausgeführte `<invoke>` Aktivitäten realisiert. Wenn mindestens einer der Dienstaufrufe erfolgreich abgeschlossen wurde, gilt die Aktivität in dem fachlichen Modell als *Erledigt*.

Zustand ableiten (Z4).

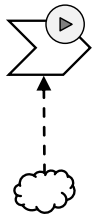


Abbildung 5.24:
Zustand ableiten (Z4).

Problem: Wie kann der Zustand einer Aktivität im fachlichen Modell bestimmt werden, wenn die Aktivität keine Entsprechung im ausführungsnahen Modell besitzt?

Beschreibung: Der Zustand dieser Aktivitäten muss von anderen Zuständen abgeleitet werden. Als mögliche Quellen für einen abgeleiteten Zustand können etwa Vorgängeraktivitäten aus dem fachlichen Modell oder der Zustand der Prozessinstanz dienen. Prinzipiell sind für eine Zustandsableitung aber beliebige Datenquellen denkbar, wie in Abbildung 5.24 angedeutet.

Beispiel: In einem Modell zum abstrakten Monitoring eines Prozesses sind mehrere Aktivitäten zur Repäsentation von Sicherheitsfunktionen wie Ver- und Entschlüsselung enthalten, die nicht von der Prozessausführungsumgebung durchgeführt werden, sondern durch eine andere Komponente realisiert werden. Der Zustand dieser Aktivitäten wird daher von Events abgeleitet, die diese Komponente ausgibt.

Übergang bilden (Z5).

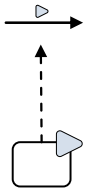


Abbildung 5.25:
Übergang bilden (Z5).

Problem: Wie kann der Zustand dargestellt werden, wenn es für Aktivitäten im Quellmodell keine Entsprechung im Zielmodell gibt?

Beschreibung: Ein ausführbares Modell kann Aktivitäten enthalten, die keine Entsprechung im fachlichen Modell haben. Wenn deren Ausführung eine gewisse Zeit benötigt, kann der Zustand der Prozessinstanz auf fachlicher Ebene eingefroren erscheinen, obwohl Aktivitäten ausgeführt werden. Um anzuzeigen, dass sich die weitere Durchführung in Vorbereitung befindet, kann der Zustand der unsichtbaren Aktivitäten auf Kontrollverbindungen übertragen werden wie in Abbildung 5.25 skizziert. Diese Form der Zustandsübertragung setzt Elemente zur Beschreibung von Übergängen und Beziehungen im Zielmodell voraus, wie bspw. Kontrollverbindungen. Daher kann dieses Muster in einer Zustandsprojektion auf Chevron nicht angewendet werden.

Beispiel: Ein fachliches Modell enthält die Aktivität *Bestellung erfassen* und unmittelbar danach die Aktivität *Bestellung prüfen*. In der technischen Realisierung des Prozesses müssen vor der Prüfung der Bestellung verschiedene Daten über den Kunden aus unterschiedlichen Quellen abgefragt und dem zuständigen Sachbearbeiter auf einem Netzlaufwerk bereitgestellt werden. Der kombinierte Zustand der dafür nötigen Dienstaufrufe kann auf die Kontrollverbindung oder die Datenverbindung zwischen den beiden Aktivitäten im fachlichen Modell übertragen werden, um diesen Sachverhalt darzustellen.

Zustände sammeln (Z6).

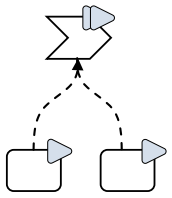


Abbildung 5.26:
Zustände sammeln (Z6).

Problem: Wie kann der Zustand bei mehrfacher Ausführung einer Aktivität übertragen werden?

Beschreibung: Die Sammlung von Zuständen beschreibt die Anreicherung einer Aktivität im Zielmodell mit mehreren Zuständen. Als Quelle für die Sammlung können verschiedene Aktivitäten eines Prozessmodells verwendet werden. Des Weiteren besteht die Möglichkeit, die mehrfache Ausführung einer Aktivität innerhalb einer Schleife als Quelle für die Sammlung von Zuständen zu verwenden. Alternativ kann ein und dieselbe Aktivität verschiedener Instanzen eines Prozesses zum Einsatz kommen. Im Unterschied zur Kombination von Zuständen (Z3), bei der mehrere Zustände auf einen Zustand abgebildet werden, bleiben die Zustände bei einer Zustandssammlung erhalten und werden einer einzigen Aktivität als Feld zugeordnet. Die Menge der Instanzen, die die Informationsgrundlage für die Übertragung bilden, kann durch eine Abfrage bestimmt werden. Die Skizze in Abbildung 5.26 zeigt die Zustandssammlung von zwei Aktivitäten aus unterschiedlichen Prozessinstanzen auf eine Aktivität in Chevron. Die Zustandssammlung in dieser Skizze umfasst keine Zustandsänderung (Z2).

Beispiel: Die Zustandssammlung kann dazu genutzt werden, einen Prozess mit dem Verständnis eines Fließbandes oder einer Pipeline darzustellen, anstelle des Verständnisses von einem Prozessmodell und einzelner Instanzen. Enthält das Feld der gesammelten Zustände einen Zustand *Aktiv*, so wird diese Aktivität gerade in einem der laufenden Prozessinstanzen ausgeführt. Eine Zustandssammlung aller aktiven Instanzen eines Prozesses könnte z.B. die Informationsbasis für eine Sicht zur Analyse von Ressourcenengpässen liefern.

Teil III

Compliance von Geschäftsprozessen

Überblick zu Teil III

In Teil III wird der Stand der Technik durch eine ganzheitliche Betrachtung der Compliance von Geschäftsprozessen, Regeln und regelrealisierenden Prozessfragmenten sowie von Sichten zur Unterstützung von Compliance erweitert. Compliance ist an dieser Stelle als beispielhaftes Themengebiet zu verstehen, das einen erheblichen Einfluss auf die Komplexität von Geschäftsprozessen und deren Management hat. Neben der Nutzung von Sichten als Mittel, um die Komplexität beherrschbar zu machen, werden in diesem Teil der Arbeit auch die für Compliance erforderliche Prüfung der Regelkonformität sowie die Nutzung von Techniken zur Wiederverwendung als komplexitätsreduzierendes Konzept eingesetzt.

In Kapitel 6 wird ein Modell zum Management von Compliance in Geschäftsprozessen vorgestellt. Dieses Modell berücksichtigt ein Vorhandensein mehrerer Abstraktionsebenen (A1) und die Verwendung unterschiedlicher Prozesssprachen (A2). Außerdem wird der Tatsache Rechnung getragen, dass je nach technischen und organisatorischen Gegebenheiten eines Unternehmens (A13) möglicherweise nur Teile eines Geschäftsprozesse durch IT-Systeme (A3) und davon wiederum nur ein Teil durch Workflows implementiert sind (A4). Um eine Lücke im Stand der Technik zu schließen, werden regelrealisierende Prozessfragmente (kurz rr-Fragmente) als wiederverwendbare Vorlagen regelkonformen Verhaltens konzipiert (A16) und in das Modell zum Management von Compliance integriert.

In Kapitel 7 werden Sichten auf Geschäftsprozesse mit Compliance-Bezug entworfen. Diese Sichten betreffen insbesondere den Umgang mit rr-Fragmenten. Dies umfasst konkret die Unterstützung der Wiederverwendung (A21), Nachvollziehbarkeit der Umsetzung von Compliance-Anforderungen in einem Prozess (A14, A20) als auch ein Entgegenwirken einer möglichen Belastung des Prozesses durch bestimmte Compliance-Anforderungen (A19).

In Kapitel 8 werden integrative Techniken entwickelt, die zum Management von Compliance mit rr-Fragmenten benötigt werden. Dies betrifft die Integration von rr-Fragmenten in einen Prozess wie auch die Verknüpfung des Ansatzes von rr-Fragmenten mit Ansätzen zur Regelkonformitätsprüfung, um der Anforderung nach einem ganzheitlichen Ansatz zur Unterstützung von Compliance nachzukommen, der sowohl die Prüfung als auch die Herstellung von Regelkonformität umfasst (A15).

KAPITEL 6

MANAGEMENT VON COMPLIANCE IN GESCHÄFTSPROZESSEN

Das Management von Compliance in Geschäftsprozessen muss ein breites Spektrum an Anforderungen abdecken und auf die organisatorischen und technischen Rahmenbedingungen eines Unternehmens abgestimmt werden. Je stärker ein Prozess automatisiert ist, desto mehr technische „Compliance-Instrumente“ können und müssen dazu eingesetzt werden, um eine Regelkonformität zu gewährleisten. Ein *Compliance-Instrument* ist ein abstrakter Begriff zur Beschreibung einer Einflussnahme, die sich auf Prozesse, auf die Organisation als auch auf IT-Systeme beziehen kann. Die Einflussnahme auf das Verhalten kann in Bezug auf den Prozesslebenszyklus präventiv in der Modellierungsphase oder reaktiv in der Ausführungs- und Monitoringphase erfolgen. Neben der Einflussnahme auf das Verhalten ist die Prüfung des Verhaltens essentiell, damit potentiell mögliche oder bestehende Verstöße gegen Regeln erkannt und in der Modellierungsphase oder sogar unmittelbar in der Ausführungsphase unterbunden werden können. Allerdings müssen auch wenig automatisierte Abläufe betrachtet und zur Einhaltung von Regeln angepasst und kontrolliert werden.

6.1 Ganzheitliche Herangehensweise

In der Industrie haben sich unterschiedliche Ansätze entwickelt, um Compliance umzusetzen. Moosmayer [Moo10] hat dazu einen umfassenden, juristisch fundierten Leitfaden für Unternehmen vorgestellt, mit der Compliance in der Praxis handhabbar gemacht werden kann. Dieser Leitfaden zeigt wie komplex die Rechtsgrundlagen, Risikoanalysen, Präventions- und Kontrollmaßnahmen sind. Auch ist die für Compliance erforderliche Organisationsstruktur in großen Unternehmen keinesfalls trivial. Zusammen mit Pohlmann benennt Moosmayer in einem konkreten Compliance-Programm aber auch eine geradlinige Strategie: „Vorbeugen (prevent) – Aufdecken (detect) – Reagieren (respond) und fortlaufende Verbesserung (Continous Improvement)“ [Moo10]. Andere Leitfäden erwähnen, dass Compliance im Wesentlichen darin besteht, das Richtige zu tun und die Dinge richtig zu tun: „do the right things and do things right“[6]. Dieser Gedanke findet sich bspw. in dem Regelwerk Sarbanes-Oxley Act (SOX) [Con02] in Abschnitt 404 in der Anforderung wieder, dass interne Compliance-Instrumente eingerichtet und deren Einhaltung überprüft werden müssen.

[6] Die primäre Quelle dieser Aussage konnte nicht eindeutig identifiziert werden.

Das eigentliche Management von Compliance beginnt allerdings einen Schritt zuvor, nämlich in der Erfassung und Prüfung, welche Compliance-Quellen und welche Teile daraus für ein Unternehmen und seine Geschäftsprozesse relevant sind und wie die darin enthaltenen Anforderungen zu interpretieren sind [DCD+09]. Erst nach diesem Schritt können Compliance-Instrumente ausgewählt oder definiert werden, um die relevanten Compliance-Anforderungen umzusetzen. Sackmann und Kähmer [SK08] unterscheiden zwischen automatisierbaren und manuellen Compliance-Instrumenten. Ferner beschreiben Namiri und Stojanovic [NS07], dass Anforderungen in der Umsetzung auf die vorhandenen IT-Systeme und die darin genutzten Technologien angepasst werden müssen. Komplexe Rahmenwerke stellen eine Nachvollziehbarkeit her indem Regelwerke, Compliance-Anforderungen, Compliance-Risiken, Compliance-Instrumente und Prozesse miteinander verknüpft werden [Elg12].

Compliance Management-Modell. Die verschiedenen Strategien und Ansätze im Stand der Technik und der industriellen Praxis zum Erreichen von Compliance haben überwiegend gemein, dass man (1) *prüft*, ob Erforderliches getan und Untersagtes unterlassen wurde und (2) präventiv oder reaktiv *agiert*. Schumm et al. [SAL+10a] haben ein Modell vorgeschlagen, dass diese Aspekte berücksichtigt, vgl. Abbildung 6.1. Auf Basis der Ausführungen von Schumm et al. [SAL+10a] wird dieses Modell nun vorgestellt.

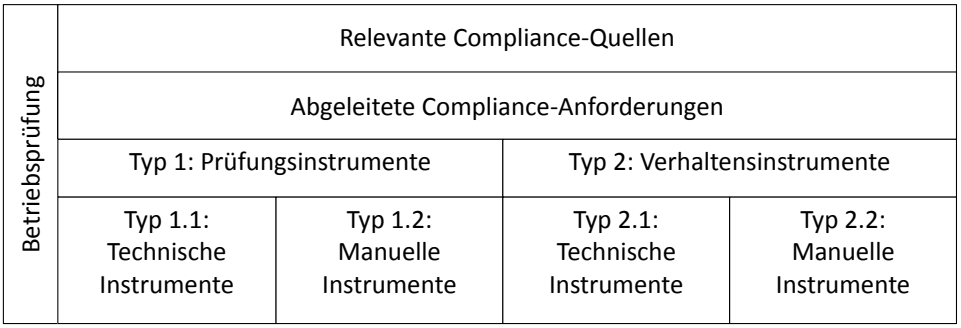


Abbildung 6.1: Compliance-Instrumente im Compliance Management-Modell, vgl. [SAL+10a].

Ausgehend von unternehmensrelevanten Compliance-Quellen werden die für einen Prozess relevanten Anforderungen ausgewählt und nach juristischen Erwägungen interpretiert. Diese Compliance-Anforderungen werden anschließend verfeinert und konkretisiert. Eine Anforderung kann einerseits eine Prüfung erfordern, welche zeigt, ob ein Ablauf oder ein IT-System regelkonform aufgebaut ist und sich regelkonform verhält. Andererseits kann eine Anforderung auch erfordern, dass erwünschtes Verhalten in einen Ablauf oder in ein IT-System integriert wird. Während Anforderungen auf das Handeln in Prozessen einwirken, geben Events aus der Prozessausführung und deren Abstraktion und Aggregation in Form von Compliance-Indikatoren und Berichten Aufschluss über die Wirkung einer Maßnahme. Diese Rückmeldung kann wiederum zu Reaktionen und geänderten Anforderungen führen.

Die *Betriebsprüfung*, obgleich intern in Form einer Revision oder extern veranlasst, wird als wichtiger übergreifender Aspekt betrachtet, der seinerseits Anforderungen an die Nachvollziehbarkeit, Transparenz und Prüfung der Umsetzung von Compliance-Anforderungen stellt.

Es werden zwei Haupttypen an Compliance-Instrumenten unterschieden: *Typ 1 umfasst Prüfungsinstrumente* und *Typ 2 umfasst Verhaltensinstrumente*. Diese beiden Typen sind ferner in technische und manuelle Instrumente untergliedert. Welche Compliance-Instrumente konkret zum Einsatz kommen, hängt von der konkreten technischen Infrastruktur einer Organisation bzw. eines Unternehmens und der Ausgestaltung eines Geschäftsprozesses ab.

Zu jedem dieser Typen gibt es bereits zahlreiche Techniken, Ansätze und praktische Beispiele, die der Zusammenstellung von Schumm et al. [SAL+10a] entnommen werden können.

Die Instrumenttypen sind wie folgt untergliedert:

- *Typ 1.1 (technische Prüfinstrumente)* umfasst neben Techniken zum Monitoring und der Verarbeitung von Events (engl. *Complex Event Processing, CEP*) u.a. formale Methoden zur Prüfung eines Prozessmodells (vgl. Abschnitt 2.3.3).
- *Typ 1.2 (manuelle Prüfinstrumente)* umfasst die Befragung von Menschen in Interviews oder mit Fragebögen als auch Akteneinsicht im Rahmen einer Revision.
- *Typ 2.1 (technische Verhaltensinstrumente)* wirkt auf das Verhalten von IT-Systemen, bspw. werden Anpassungen an Sicherheitseinstellungen und zusätzliche Software wie eine Firewall und ein Anti-Virenprogramm genutzt, um Sicherheitsanforderungen umzusetzen.
- *Typ 2.2 (manuelle Verhaltensinstrumente)* umfasst Instruktionen und Verhaltensregeln, die Mitarbeiter im Rahmen einer Vereinbarung, internen Richtlinien oder in Form eines Compliance-Kodex erhalten, etwa bzgl. der Annahme von Geschenken oder zur Regelung des Umgangs mit Geschäftspartnern.

Für eine effiziente Unterstützung von Compliance in Geschäftsprozessen sind alle Typen zu betrachten, um einerseits Fehlverhalten erkennen zu können und andererseits die Umsetzung von regelkonformen Verhalten zu unterstützen.

rr-Fragmente als Compliance-Instrument. Für den Bedarf, regelkonformes Verhalten in Form von wiederverwendbaren Prozessstrukturen in bestehende Geschäftsprozesse integrieren zu können (A16), wird das Konzept regelrealisierender Prozessfragmente, kurz rr-Fragmente, vorgeschlagen (engl. *Compliance fragment*). Es passt bestehende prozessbezogene Ansätze zur Wiederverwendung für die Nutzung im Kontext Compliance an und erweitert diese für eine Verwendung als Verhaltensinstrument, d.h. Typ 2.1 oder Typ 2.2.

Die in Abbildung 6.2 veranschaulichte Entscheidung zur Auswahl einer Realisierungsoption positioniert den Ansatz der rr-Fragmente zu bestehenden Techniken zur Wiederverwendung. Sofern möglich, sollte zur Realisierung von Compliance-Anforderungen gegenwärtig ein modularer, in sich abgeschlossener Sub-Prozess oder eine technisch implementierte Geschäftsregel (Business Rule) eingesetzt werden. Dieser Empfehlung liegt folgende Argumentation zugrunde: Das Konzept von Sub-Prozessen ist bereits in den meisten Sprachen und Werkzeugen etabliert und in Ausführungsumgebungen durch Standards unterstützt, vgl. bspw. BPEL-SPE [KKL+05]. Auch ist die Nutzung von Geschäftsregeln bereits in vielen Systemen etabliert und durch Standards und Technologien umfassend unterstützt. Sub-Prozesse und Geschäftsregeln haben aufgrund von deren Modularität den Vorteil der

einfachen Austauschbarkeit gegenüber von fest in einen Prozess integrierten rr-Fragmenten. rr-Fragmente erfordern zum gegenwärtigen Stand eine Anpassung der Modellierungsumgebung und es gibt weder für rr-Fragmente im Besonderen noch für Prozessfragmente im Allgemeinen einen etablierten Standard. Eine Architektur und prototypische Implementierung, welche die Nutzung von rr-Fragmenten dennoch ermöglicht, wird in Teil IV (technische Realisierbarkeit) vorgestellt.

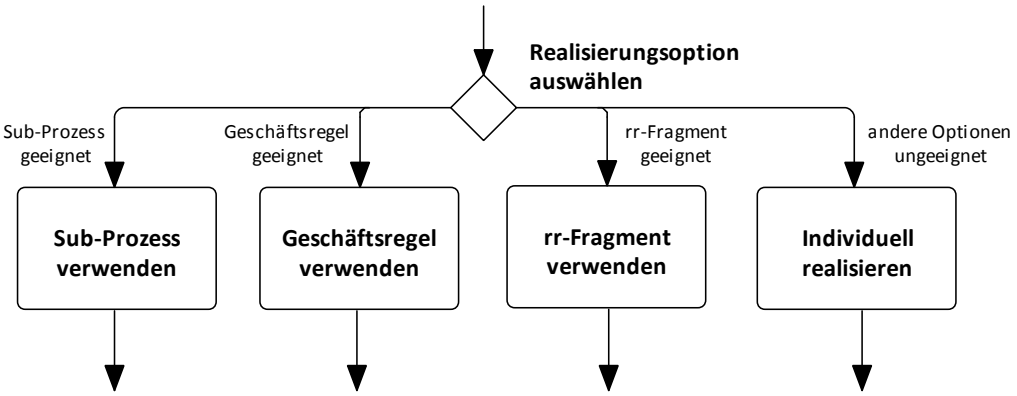


Abbildung 6.2: Auswahl einer Option zur Realisierung regelkonformen Verhaltens in einem Prozess.

Für verschiedene Compliance-Anforderungen ist allerdings ein Eingriff in den Prozessablauf erforderlich, der mit einem Sub-Prozess oder einer Geschäftsregel nicht abgebildet werden kann. Sollte die Umsetzung einer derartigen Anforderung auch durch ein rr-Fragment nicht realisiert werden können, ist eine individuelle Lösung erforderlich. Welche Option der Realisierung letztlich ausgewählt wird, liegt im Ermessen des Modellierers. Dieser entscheidet, welche der Optionen angemessen und am besten geeignet ist. Eine Prozessbelastung kann an dieser Stelle wohlgemerkt nicht nur durch rr-Fragmente, sondern ebenso durch Sub-Prozesse oder Geschäftsregeln entstehen. Eine Prozessbelastung besteht, wenn die verwendeten Strukturen lediglich Compliance-Anforderungen erfüllen, aber sonst keine wertschöpfende Funktion im Prozess erfüllen.

6.2 Erweiterungen des Lebenszyklus von Geschäftsprozessen

Für die Compliance-bezogenen Aufgaben sind mehrere Erweiterungen am Lebenszyklus von Geschäftsprozessen erforderlich. Der in Hinblick auf die Herstellung von Compliance zur Entwurfszeit erweiterte Lebenszyklus ist in Abbildung 6.3 dargestellt.

Eine zusätzliche Phase ist zur *Compliance-Begutachtung* und Interpretation der Compliance-Quellen erforderlich [DCD+09]. In dieser Phase wirkt ein Compliance-Beauftragter auf die Prozessmodellierung ein. Die Prozessmodellierung kann durch die *Modellierung regelkonformer Strukturen* in Form von Sub-Prozessen, Geschäftsregeln und rr-Fragmenten unterstützt werden. Diese Strukturen können bspw. von Beratungsunternehmen bereitgestellt werden. *Statische Verifikation* unterstützt den Prozesseigner in der präventiven Prüfung, ob die Anforderungen korrekt umgesetzt wurden. Der Lebenszyklus schließt mit der Phase der *Betriebsprüfung*. In dieser Phase kontrolliert ein Compliance-Beauftragter die ergriffenen Maßnahmen und bewertet deren Wirksamkeit. Ferner unterstützt er einen externen Prüfer, die

Umsetzung der Anforderungen nachzuvollziehen. Der Aufbau eines Compliance Office ist an dieser Stelle eine mögliche Maßnahme, um die umfassenden Aufgaben zur Steuerung, Kontrolle, Sanktion und Kommunikation effizient zu managen. Die Compliance Organisation bei Siemens bspw. umfasste im Jahr 2009 rund 600 Mitarbeiter, die teils in der Zentrale, teils in Bereichen und Regionalgesellschaften des Unternehmens verortet waren [Moo10].

Über die diskutierten Erweiterungen hinaus kann die Herstellung von Compliance auch zur Lauf- und Analysezeit unterstützt werden. In Abschnitt 2.3.3 wurden dazu verschiedene Herangehensweisen wie bspw. zum Compliance Enforcement oder zum Process Mining vorgestellt, die in Bezug auf den Lebenszyklus eine Ergänzung der Ausführungs-, Monitoring- und Analysephase bedeuten.

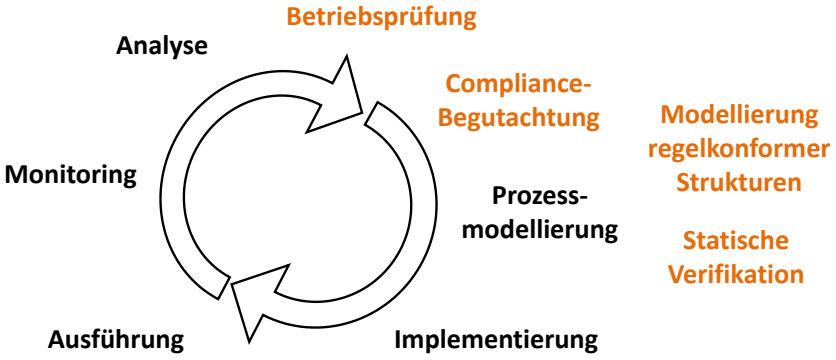


Abbildung 6.3: Lebenszyklus von Geschäftsprozessen mit besonderer Betrachtung von zusätzlichen Phasen für Compliance (in Orange).

Neben den Compliance-geprägten Phasen kann auch das *Informationsdesign* das Compliance Management unterstützen. Der effektiv resultierende Lebenszyklus von Geschäftsprozessen mit besonderer Betrachtung von Compliance ist in Abbildung 6.4 dargestellt. Das Informationsdesign unterstützt Compliance-Beauftragte, Prozesseigner und Betriebsprüfer durch die Bereitstellung von bedarfsorientierten Sichten. Durch Sichten können bspw. Compliance-relevante Strukturen hervorgehoben oder ausgeblendet werden [SLS10b]. Eine gesonderte Phase zum *Grafikdesign* kann dazu beitragen, dass das Erscheinungsbild von Aktivitäten und Datenelementen die für Compliance relevanten Informationen angemessen darstellt.

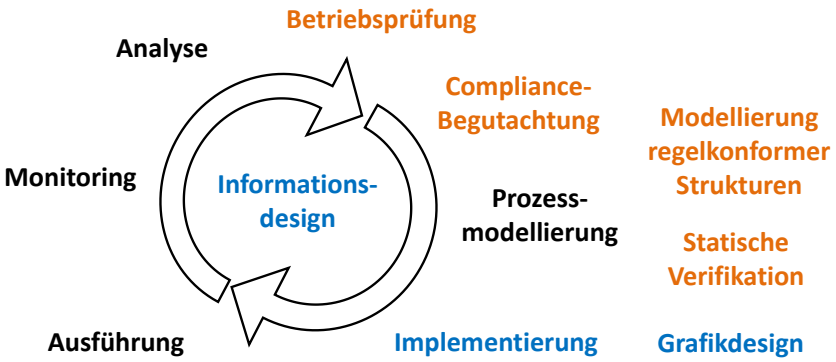


Abbildung 6.4: Informationsdesign im Lebenszyklus von Geschäftsprozessen (in Blau) mit besonderer Betrachtung von Compliance (in Orange).

6.3 Konzeption von rr-Fragmenten

Das Konzept wiederverwendbarer Prozessstrukturen wurde bereits in vielen Facetten diskutiert. Nach bestem Wissen wurde bisher allerdings kein Ansatz zur Wiederverwendung von Prozessstrukturen vorgeschlagen, der die Anforderungen des Themenkomplexes Compliance abdeckt. Der in diesem Abschnitt vorgestellte Ansatz wiederverwendbarer Prozessstrukturen für Compliance wurde von Daniel et al. [DCD+09] im Kontext des Projekts COMPAS skizzenhaft positioniert, von Schumm et al. [SLM+10] ausgearbeitet und von Schumm et al. [STK+10] weiterentwickelt.

Eine grundlegende Annahme des Ansatzes besteht darin, dass zum Management der Compliance-Anforderungen die wiederverwendbaren Einheiten manuell von einem Compliance-Experten identifiziert und ggf. aus einem bereits regelkonformem Prozess extrahiert oder auch speziell den Anforderungen entsprechend neu konstruiert werden, vgl. Schumm et al. [SLM+10]. Die Konzeption der rr-Fragmente [SLM+10] erweitert bestehende Ansätze zu wiederverwendbaren Prozessstrukturen um den Compliance-Aspekt. Zu nennen sind an dieser Stelle die Vorarbeiten von Eberle et al. [EUL09] sowie Ma und Leymann [ML08]. Die in [EUL09] vorgeschlagenen und in der Konzeption der rr-Fragmente ebenfalls verwendeten Prinzipien von nur teils verbundenen Kontrollverbindungen und Regionen zur Auszeichnung von Platzhaltern für Aktivitäten oder andere Prozessfragmente findet sich auch in Eberle [Ebe14] in verfeinerter Form wieder. Ma [Ma12] unterscheidet zudem verschiedene Stile der Wiederverwendung von Prozessfragmenten:

- Black Box: Die Logik eines Prozessfragments ist verborgen und kann nicht geändert werden.
- Grey Box: Die Logik eines Prozessfragments ist verborgen, aber das Verhalten kann durch die Festlegung von Parametern gesteuert werden.
- Glass Box: Die Logik eines Prozessfragments ist sichtbar, aber kann nicht geändert werden.
- Holey Box: Die Logik eines Prozessfragments ist sichtbar und kann an vordefinierten Stellen geändert werden.
- Open Box: Die Logik eines Prozessfragments ist sichtbar und kann beliebig geändert werden.

Das Konzept der rr-Fragmente kann mehreren dieser Stile zugeordnet werden. Ma schlägt die Nutzung des Glass Box Stils für die in Schumm et al. [SLM+10] konzipierten rr-Fragmente vor, damit das ursprüngliche Design bei der Wiederverwendung erhalten bleibt. Sofern ein rr-Fragment Platzhalter zur Anpassung enthält, kann es auch dem Holey Box Stil zugeordnet werden. Die Nutzung des Black Box oder Grey Box Stils eignen sich für rr-Fragmente nicht, da die Umsetzung von Compliance-Anforderungen für den Modellierer als auch für die Wirtschaftsprüfung transparent sein müssen. Die Nutzung des Open Box Stils ist möglich, birgt jedoch das Risiko, dass das ursprüngliche Design des rr-Fragments bei der Wiederverwendung zerstört wird.

Metamodell. Ein rr-Fragment ist ein zusammenhängender Graph, der geringere Kriterien bzgl. der Vollständigkeit und Konsistenz erfüllen muss als ein syntaktisch korrekter Prozessgraph. Ein rr-Fragment besteht aus *Aktivitäten*, *Aktivitätsplatzhaltern* (engl. *regions*) und *Kontrollverbindungen*, mit denen Kontrollabhängigkeiten ausgedrückt werden. Ein rr-Fragment kann einen *Kontext* wie z.B. Datenelemente definieren und es kann einen Knoten für den Prozess-Beginn und ein Prozess-Ende enthalten. Ein rr-Fragment kann mehrere *eingehende und ausgehende Kontrollverbindungen* besitzen, mit denen es in einen Prozess oder mit anderen rr-Fragmenten integriert werden kann. Ein rr-Fragment muss mindestens eine Aktivität enthalten und es muss außerdem möglich sein, das rr-Fragment zu einem ausführbaren Prozessgraphen zu vervollständigen. Demzufolge ist ein rr-Fragment nicht direkt ausführbar und teilweise undefiniert, vgl. [SLM+10].

In Abschnitt 6.1 wurde vorgeschlagen, rr-Fragmente als Compliance-Instrument zu betrachten, mit dem Einfluss auf die Regelkonformität eines Prozesses genommen werden kann. Drei Aspekte charakterisieren ein rr-Fragment als Compliance-Instrument und heben es damit neben seinem Einsatzzweck auch von einer herkömmlichen wiederverwendbaren Prozessstruktur ab: ein rr-Fragment kann (1) *parametrisiert* sein, um Freiheitsgrade anzuzeigen, in denen es angepasst werden darf, (2) die enthaltenen Platzhalter können mit *Constraints* annotiert sein und (3) die vorhandenen eingehenden und ausgehenden Kontrollverbindungen können in *verpflichtende und optionale* unterschieden werden. Dadurch kann der Konstrukteur eines rr-Fragments explizit ausweisen, welche Teile eines rr-Fragments bei der Integration in einen Prozess geändert werden können ohne seine bezweckte Funktion zu beinträchtigen. Um das Konzept flexibel zu halten, besteht keine Festlegung, in welcher Beschreibungssprache die Constraints zu formulieren sind. Zur Gewährleistung der Nachvollziehbarkeit sollten ein rr-Fragment und dessen Bestandteile mit einem eindeutigen Identifikationsmerkmal versehen sein.

Verknüpfungen. Anhand der eindeutigen Identifikation eines rr-Fragments kann eine Verknüpfung zwischen dem abstrakten, in Prosa beschriebenen Compliance-Instrument und dessen konkreter Implementierung durch rr-Fragmente hergestellt werden. Formale Regeln dienen hierbei zur Verhaltensprüfung zur Entwurfszeit, während rr-Fragmente das geforderte Verhalten realisieren.

Analog dazu können auch Sub-Prozesse oder Geschäftsregeln als konkrete Implementierung eines Compliance-Instruments verwendet werden, vgl. dazu Abbildung 6.5. Auch diese würden mit formalen Regeln verknüpft werden, damit deren korrekte Integration in einen Prozess im Nachgang geprüft werden kann.

Konzept. Abbildung 6.5 zeigt diese Beziehungen in Form eines konzeptionellen Modells, angepasst nach der Darstellung von Schumm et al. [STK+10]. Das Modell beinhaltet weitere Verknüpfungen zur Positionierung im Compliance-Kontext. Die Verknüpfung von Anforderungen zu den Quellen, von denen sie abgeleitet sind, ist in Rahmenwerken zum Management von Compliance gängig [COM10d][Elg12]. Ferner stehen Anforderungen in Bezug zu den Compliance-Risiken, die sich aus deren Nichteinhaltung ergeben. Es besteht ferner eine Verbindung zwischen einem Prozessmodell und Compliance-Anforderungen durch die

angewendeten Compliance-Instrumente. Diese Verknüpfungen unterstützen Änderungen an Prozessen, welche durch neue oder geänderte Compliance-Quellen erforderlich werden.

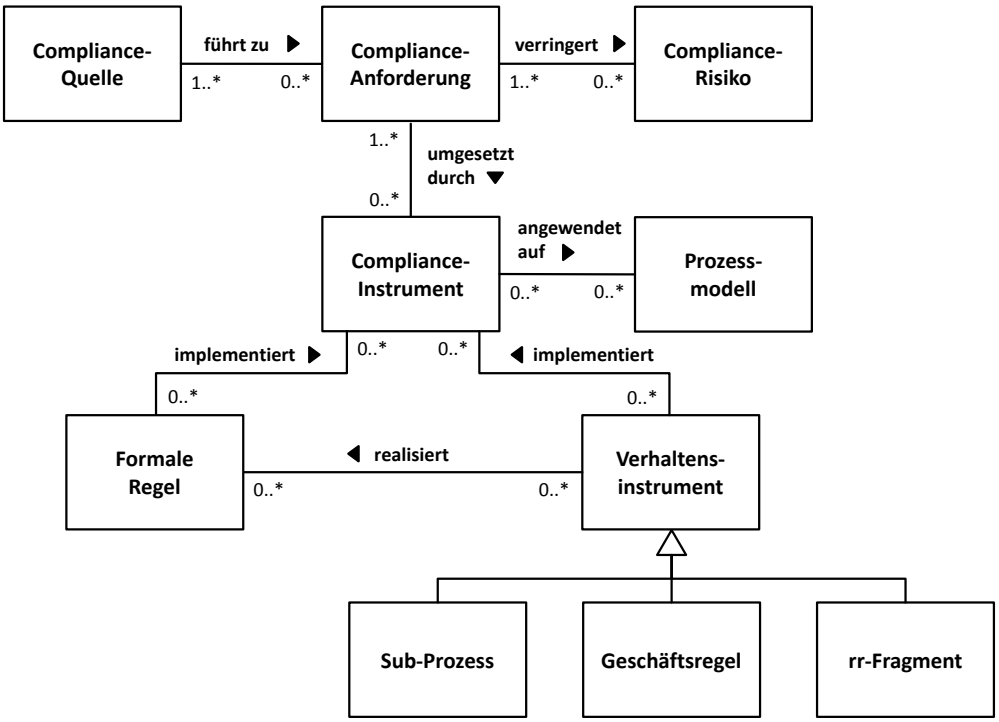


Abbildung 6.5: Konzeptionelles Modell zur Verknüpfung von Compliance-Quellen, Compliance-Anforderungen, Risiken und Mitteln zu deren Berücksichtigung in Prozessen, angepasst nach [STK+10].

Obgleich rr-Fragmente nach der Konzeption nach [SLM+10] nicht notwendigerweise komplexe Strukturen darstellen, erschließt sich ihr hoher Wiederverwendungswert durch die Gewährleistung, dass sie Compliance-Regeln treffend und angemessen realisieren und einem Prozessbesitzer dadurch Sicherheit bei seinem Handeln geben. Diese Sicherheit ist in Anbetracht der möglichen Risiken eines Compliance-Verstoßes ein wichtiges Gut.

Handhabung. Die Modellierung eines rr-Fragments muss sich an der Abstraktionsebene und Sprache der Prozesse orientieren, in denen es eingesetzt werden soll. Die Bereitstellung eines rr-Fragments auf unterschiedlichen Abstraktionsebenen und in unterschiedlichen Prozesssprachen ist daher u.U. erforderlich und sinnvoll. Zur Reduzierung der Komplexität können neben dem Modell eines rr-Fragments auch Transformationsregeln für dessen Abstraktion definiert werden.

Bibliotheken stellen eine wichtige und gängige Ausgangsbasis zur Wiederverwendung von Wissen dar. Unter dem Begriff Repository haben Habermann und Leymann [HL93] Konzepte von versionierten Bibliotheken technischer Artefakte diskutiert; Frakes und Terry [FT96] haben (Code-)Bibliotheken als wichtigen Bestandteil in Strategien zur Wiederverwendung in der Softwareentwicklung herausgestellt. Die Bereitstellung einer Bibliothek zur Wiederverwendung von rr-Fragmenten ist demzufolge für eine effiziente Handhabung naheliegend. Ferner ist die Verwaltung von rr-Fragmenten besonders in Hinblick auf Änderungen an Regelwerken und die erforderliche Nachvollziehbarkeit der Umsetzung von Compliance-

Anforderungen essentiell (A14). Zur Unterstützung der Verwaltung wird daher in Kapitel 10.3 eine Bibliothek für Prozessfragmente vorgestellt.

Neben den Sichten, die für rr-Fragmente in Abschnitt 7.1 entworfen werden, sind weitere Techniken erforderlich. Diese integrativen Techniken sind zum einen die Integration von rr-Fragmenten in einen Geschäftsprozess und zum anderen die Überprüfung der Integration durch Verifikationstechniken. Zur Vervollständigung des ganzheitlichen Ansatzes werden diese Techniken in Kapitel 8 diskutiert.

6.4 Beispiele von rr-Fragmenten

Ebenso wie Geschäftsprozesse, können auch rr-Fragmente in unterschiedlichen Beschreibungsformen und Automatisierungsstufen vorliegen. Dementsprechend kann ein rr-Fragment in einer eher abstrakten, geschäftsnahen Form vorliegen oder auch in einer eher technischen, ausführungsnahen Form. Geschäftsnahe Beispiele von rr-Fragmenten werden nachfolgend mit der grafischen Notation BPMN modelliert, um das Konzept mit einer standardisierten Darstellung zu vermitteln ohne auf technische Aspekte einzugehen. Technische Aspekte werden bei der Abbildung des Konzepts der rr-Fragmente auf die ausführungsnahe Prozesssprache BPEL betrachtet, welche in Abschnitt 9.3 geleistet wird.

Die nachfolgend diskutierten beispielhaften rr-Fragmente wurden für die vorliegende Dissertation neu konstruiert und erheben nicht den Anspruch, rechtlichen Anforderungen zu genügen. Die rr-Fragmente dienen lediglich der Veranschaulichung der Konzeption. Weitere Beispiele für rr-Fragmente können Beiträgen zu Bibliotheken für Prozessfragmente entnommen werden, darunter sind rr-Fragmente für revisionssichere Zeitstempel, für abgesicherte Dienstaufrufe, für Genehmigungen und zur Authentifizierung von Nutzern, vgl. dazu Schumm et al. [SKL+10] und Schumm et al. [SKK+11].

In den Beiträgen zu Bibliotheken für Prozessfragmente wurden die grafischen Elemente von BPMN um zusätzliche Symbole zur Darstellung von Platzhaltern und Constraints erweitert, vgl. [SKK+11]. Diese finden auch in den nachfolgenden Beispielen von rr-Fragmenten Verwendung. Die entsprechenden Symbole sind in Abbildung 6.6 aufgeführt.

Abbildung 6.6: Auswahl an Symbolen aus BPMN zu Darstellung von rr-Fragmenten mit Erweiterungen für Platzhalter und Constraints, angepasst und erweitert nach [OMG11b] und [SKK+11].

6.4.1 Revisionssichere Dokumentation

Die Dokumentation von Abläufen ist eine häufige Compliance-Anforderung, die in unterschiedlichen Bereichen zu Tage tritt. Eine Dokumentation eines Produktberatungsgesprächs kann bspw. im Bankenumfeld für einen Nachweis herangezogen werden, dass bestehende Risiken angemessen offengelegt wurden. Auch kann eine Dokumentation im Rahmen der Produkthaftung im Schadensfall von großer Bedeutung sein. Diese Dokumentationen müssen je nach Domäne und Prozess teils über mehrere Jahre aufbewahrt werden. Das rr-Fragment in Abbildung 6.7 setzt die Anforderung nach einer revisionssicheren Ablaufdokumentation um.

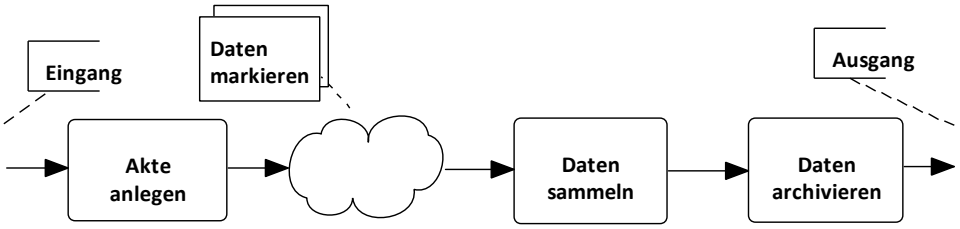


Abbildung 6.7: rr-Fragment für revisionssichere Ablaufdokumentation.

Das rr-Fragment besitzt einen Eingang und einen Ausgang zur Integration in einen Prozess, in dem während des Ablaufs anfallende Daten revisionssicher dokumentiert werden sollen. Zunächst wird eine Akte oder Vergleichbares angelegt. Das rr-Fragment beschreibt nur den Aspekt der Dokumentation; der eigentliche, zu dokumentierende Ablauf ist prozessspezifisch und wird durch einen Platzhalter repräsentiert. Die Bedingung, die für die Befüllung des Platzhalters besteht, ist die Kennzeichnung derjenigen Daten, die aufbewahrungsrelevant sind. Nach dem Ablauf, wie bspw. im Anschluss an ein Produktberatungsgespräch, werden diese Daten gesammelt und archiviert. Da nach der Archivierung ggf. weitere Schritte folgen können, wird der Prozess an dieser Stelle nicht beendet, sondern anschließend fortgesetzt.

6.4.2 Einbindung eines Compliance-Beauftragten

Compliance-Beauftragte müssen in Prozesse eingebunden werden können, um Vorgänge bzgl. der Regelkonformität zu bewerten und um Maßnahmen zur Wahrung der Regelkonformität zu veranlassen. Das rr-Fragment in Abbildung 6.8 setzt diese Anforderung um.

Zur Einbindung eines Compliance-Beauftragten wird zunächst ein Bericht zusammengestellt, der die Entscheidungsbasis für die Einschätzung eines Sachverhalts darstellt. Dieser Bericht muss an die zuständige Körperschaft zugestellt werden. Der Compliance-Beauftragte ist ggf. prozessspezifisch festzulegen, es gilt an dieser Stelle jedoch das Constraint einzuhalten, dass der Empfänger des Berichts dem Compliance Office zugeordnet sein muss. Das rr-Fragment benötigt ein (in Abbildung 6.8 nicht dargestelltes) Gegenstück im Bereich des Compliance Office, etwa einen gesonderten Prozess zur Risikobewertung, in dem eingehende Berichte nach internen Richtlinien bewertet werden und anschließend eine Reaktion formuliert und dem Anfragenden zugestellt wird. Die Zustellung der Reaktion ist in dem rr-Fragment durch das Ereignis „Reaktion empfangen“ abgebildet. Entsprechend der Reaktion des Compliance-Beauftragten sind ggf. Maßnahmen erforderlich und entsprechend einzuleiten. In diesem Fall

wird der Prozess auf besondere Weise fortgesetzt, modelliert durch Ausgang 1. Sofern der Compliance-Beauftragte keinen Regelverstoß bei dem gemeldeten Sachverhalt sieht, kann der Prozess normal fortgesetzt werden, modelliert durch Ausgang 2. Die Prozesse, in denen dieses rr-Fragment eingesetzt wird, können unterschiedlich zeitkritisch sein und nach einer bestimmten Dauer eine prozessspezifische Ausnahmebehandlung wie bspw. eine Eskalation erfordern. Um diesem Umstand Rechnung zu tragen, kann das Zeitereignis an einen konkreten Prozess angepasst werden.

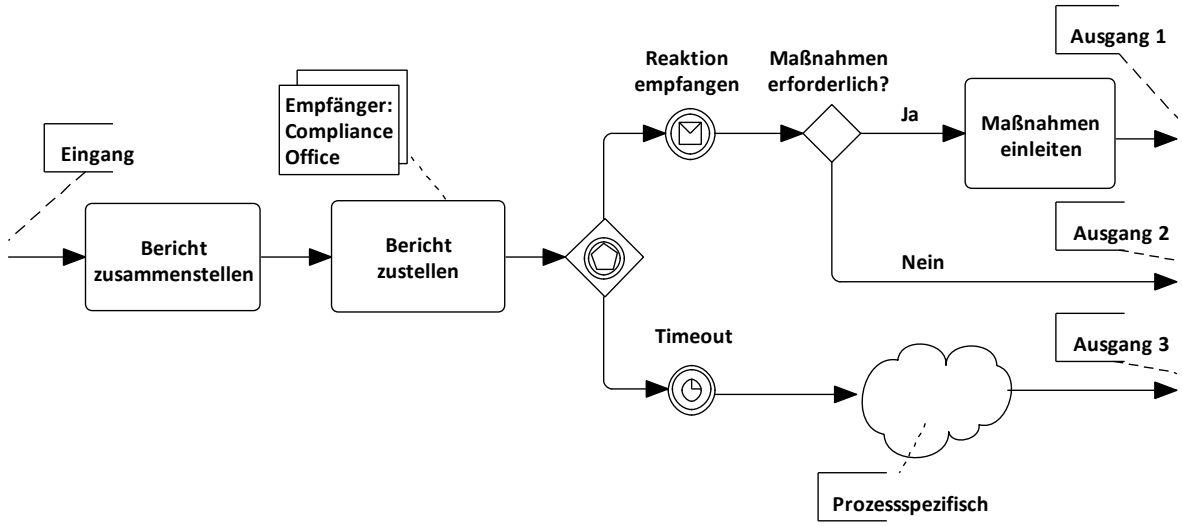


Abbildung 6.8: rr-Fragment für die Einbindung eines Compliance-Beauftragten in einen Prozess.

KAPITEL 7

SICHTENENTWÜRFE FÜR COMPLIANCE

Durch das Konzept regelrealisierender Prozessfragmente (rr-Fragmente) ergeben sich Anforderungen und Informationsbedarfe, die teilweise durch Sichten auf Prozesse abgedeckt werden können. Wesentliche Bedarfe sind das Ausschneiden, Ausblenden, Integrieren und Hervorheben von rr-Fragmenten, vgl. Schumm et al. [SLS10b]. In Abschnitt 7.1 werden Sichtenentwürfe für das Ausschneiden, das Ausblenden und das Hervorheben vorgestellt, um diesen Bedarfen nachzukommen. Diese Entwürfe erweitern und verfeinern die vorgestellten Konzepte [Sch10][SLS10b]. Die Integration von rr-Fragmenten in einen Prozess wird als weitere Technik zum Management von Compliance in Abschnitt 8.1 diskutiert.

Die bisher vorgestellten Sichtenentwürfe wurden nicht unter der expliziten Berücksichtigung von Compliance entworfen, können aber auch in diesem Kontext eingesetzt werden. Der Informationsschutz (Abschnitt 4.1.4) ist etwa eine gängige Compliance-Anforderung. Das Hervorheben bestimmter Aspekte (Abschnitt 4.1.3) kann außerdem dazu eingesetzt werden, von einem Prüfungswerkzeug festgestellte Regelverletzungen des Prozessmodells herauszustellen. Strukturreduzierende Sichten bieten allerdings auch die Möglichkeit, problematische Strukturen zu verbergen. Sichten zur Unterstützung einer Compliance-Prüfung, die keinen unmittelbaren Bezug zu rr-Fragmenten haben, werden in Abschnitt 7.2 gezeigt. Diese Sichten betreffen im Besonderen die Nachvollziehbarkeit von durchgeführten Sichtentransformationen und die Analyse von Verantwortlichkeiten.

7.1 Sichten für rr-Fragmente

7.1.1 Ausschneiden eines rr-Fragments

Problem: Die Wiederverwendung von bereits in Prozessen enthaltenen Strukturen erfordert einen Mechanismus, um diese aus einem Prozess auszuschneiden [GSL+05][ML08][Ma12]. Während dafür bereits komplexe Algorithmen entwickelt wurden, stellen Sichten eine alternative Herangehensweise dar [ACF96][SAL+10b]. Wie kann eine solche Sicht aufgebaut sein?

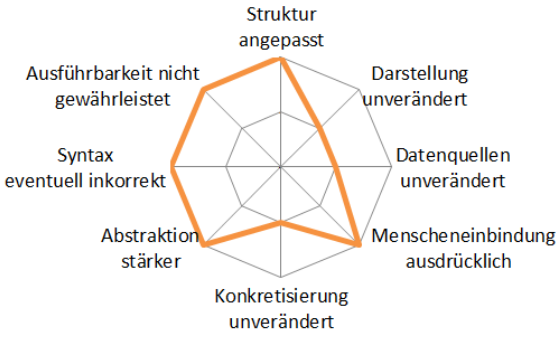


Abbildung 7.1: Klassifikation des Entwurfs.

Lösungsansatz: Das Prinzip des Ausschneidens eines rr-Fragmentes mit einer Sicht besteht darin, dessen Aktivitäten in einem Prozess zu kennzeichnen und alles andere zu entfernen. Die entsprechende Mustersequenz ist in Abbildung 7.2 dargestellt. Zunächst werden die Bestandteile des rr-Fragments als solche gekennzeichnet (M13). Die gekennzeichneten Aktivitäten sollen in den nachfolgenden Reduzierungsschritten bewahrt werden (M31) und die Kontrollabhängigkeiten zwischen den Bestandteilen des rr-Fragments durch ein Zusammenhalten des Prozesses erhalten bleiben (M29). Unter diesen Vorgaben werden alle Aktivitäten soweit möglich entfernt (M1, ohne Einhaltung der Atomicity-Eigenschaft). Die relevanten Aktivitäten und deren Zusammenhang bleiben dadurch übrig. In dem rr-Fragment nicht verwendete Datenelemente und redundante Kontrollverbindungen werden durch ein Aufräumen (M8) entfernt. Daher müssen für das rr-Fragment relevante Datenelemente nicht explizit gekennzeichnet werden. Abschließend findet eine manuelle Nachbearbeitung des Ausgabeprozesses statt, bei der prozessspezifische Aspekte generalisiert werden und das Fragment hinsichtlich veränderlicher Bestandteile parametrisiert wird.

Die manuelle Nachbearbeitung ist für den Abstraktionsgrad des rr-Fragments entscheidend. Dieser Abstraktionsgrad reicht von einer konkreten Kopie über eine parametrisierte Kopie bis zu einem abstrakten Muster. Das resultierende rr-Fragment wird abschließend persistiert. Die reduzierenden Transformationen werden daher auf einer Prozesskopie durchgeführt (vgl. Durability, Abschnitt 3.4.4).

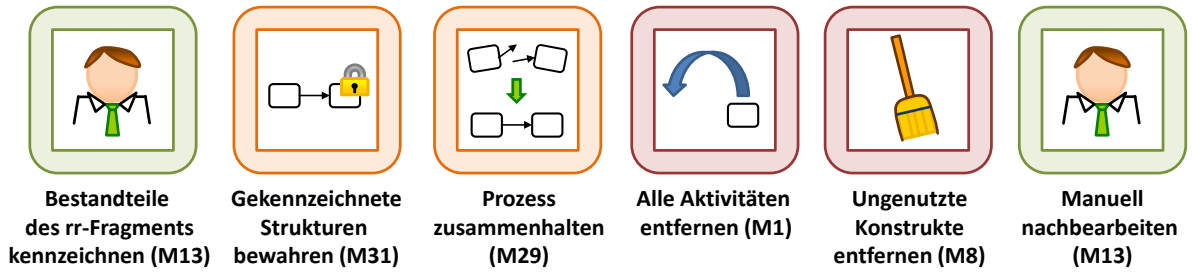


Abbildung 7.2: Mustersequenz für das Ausschneiden eines rr-Fragments.

Skizze: In der Skizze in Abbildung 7.3 werden die Aktivitäten *B* und *D* als Bestandteile eines rr-Fragments gekennzeichnet (b) und beim anschließenden Entfernen aller Aktivitäten bewahrt (c). Bei der manuellen Nachbearbeitung werden prozessspezifische Bezeichnungen und Eigenschaften angepasst, wodurch *B* im resultierenden rr-Fragment zu *B'* und *D* zu *D'* wird. Das rr-Fragment wird mit einer passenden Bezeichnung (F_1) und Schlagwörtern versehen und anschließend mit Compliance-Anforderungen und -Regeln verknüpft.

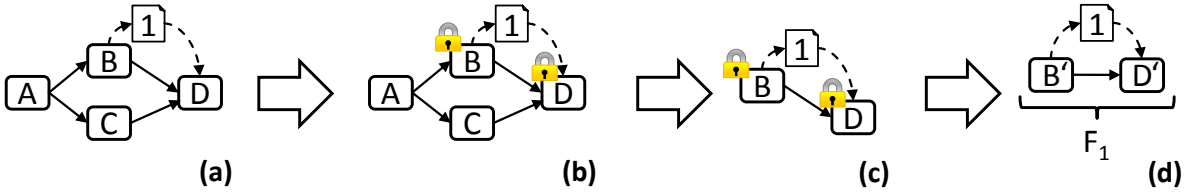


Abbildung 7.3: Skizze für ein Ausschneiden von rr-Fragmenten.

Beispiele: Der Verantwortliche für einen Kleinkreditvergabeprozess soll sicherstellen, dass in dem Prozess eine revisionssichere Dokumentation der Kundenkontakte stattfindet. Er weiß, dass diese Anforderung zwar bereits in einem anderen Prozess realisiert wurde, aber noch nicht als rr-Fragment bereitgestellt wurde. In Abstimmung mit dem Prozessverantwortlichen des anderen Prozesses werden die Strukturen, die die revisionssichere Dokumentation betreffen gekennzeichnet und das rr-Fragment anschließend extrahiert. Nach einer manuellen Nachbereitung wird das rr-Fragment in den Kleinkreditvergabeprozess integriert und zusätzlich in einer Fragment-Bibliothek zur weiteren Wiederverwendung abgelegt.

7.1.2 Ausblenden von rr-Fragmenten

Problem: Durch zahlreiche Anforderungen ist mitunter eine große Zahl von rr-Fragmenten in einem Prozess enthalten. Sofern die rr-Fragmente keine wertschöpfende Funktion, sondern Kontrollfunktionen, Protokollierungen oder Sicherheitsaspekte betreffen, resultiert daraus eine Prozessbelastung [SLM+10] oder eine semantische Verschmierung [Bun12]. Wie kann einer solchen Prozessbelastung entgegengewirkt werden?

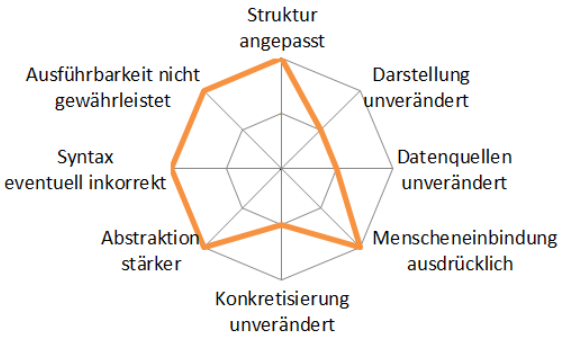


Abbildung 7.4: Klassifikation des Entwurfs.

Lösungsansatz: Um dennoch das einfache Verständnis der wertschöpfenden Strukturen in einem Geschäftsprozess zu ermöglichen, können prozessbelastende rr-Fragmente temporär entfernt, d.h. ausgeblendet werden. Dazu kann eine Mustersequenz verwendet werden, die der Mustersequenz zum Ausschneiden eines rr-Fragments ähnelt und eine dazu identische Klassifikation aufweist. Eine entsprechende Mustersequenz ist in Abbildung 7.5 dargestellt. Zuerst werden die im Prozess enthaltenen rr-Fragmente gekennzeichnet und klassifiziert, ob sie als wertschöpfend oder als nicht relevant betrachtet werden (M13). Für weitere Transformationen gilt die Vorgabe, dass der Prozess zusammengehalten werden soll (M29). Danach werden diejenigen rr-Fragmente entfernt, die als nicht wertschöpfend betrachtet werden (M1). Abschließend werden ungenutzte Datenelemente entfernt (M8).

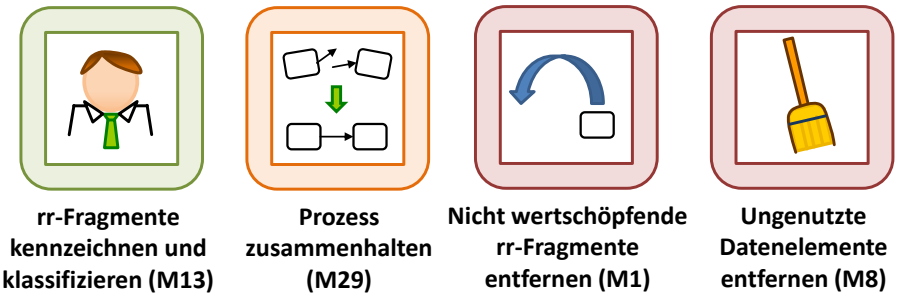


Abbildung 7.5: Mustersequenz für ein Ausblenden von rr-Fragmenten.

Das Kennzeichnen und Klassifizieren von rr-Fragmenten setzt ein zu definierendes Annotationsschema voraus. Ein Annotationsschema stellt den Bezug zwischen Prozessstrukturen und Compliance her und kann zur Auswahl auszublendender rr-Fragmente genutzt werden.

Skizze: In Abbildung 7.6 werden die im Prozess enthaltenen rr-Fragmente F_1 und F_2 manuell gekennzeichnet (b). Das rr-Fragment F_1 wird von einer bestimmten Interessengruppe als nicht wertschöpfend betrachtet. Es wird daher ausgeblendet, während rr-Fragment F_2 in der Sicht erhalten bleibt (c). Die überflüssige Kontrollverbindung $A \rightarrow G$ wird beim Aufräumen entfernt.

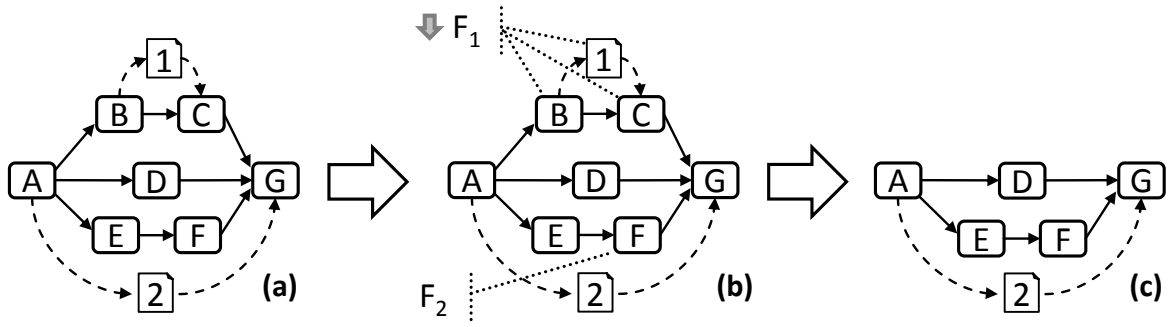


Abbildung 7.6: Skizze für ein Ausblenden von rr-Fragmenten.

Beispiele: Ein Workflow kann bspw. mehrere rr-Fragmente zum Aufruf eines Dienstes zur revisionssicheren Dokumentation von Ein- und Aufgabedaten mit gesicherten Zeitstempeln enthalten. Diese rr-Fragmente sind für die technische Umsetzung der Regelkonformität wesentlich, aber sie tragen nicht zum Verständnis der eigentlichen Ablauflogik des Workflows bei und können bspw. beim Monitoring einer Prozessinstanz ausgeblendet werden, wenn deren Ausführung erfolgreich abgeschlossen wurde.

7.1.3 Hervorheben von rr-Fragmenten

Problem: Zur Handhabung von rr-Fragmenten bestehen weiterführende Anforderungen an das Hervorheben von Strukturen. In einem Prozess können mehrere, u.U. nicht zusammenhängende rr-Fragmente enthalten sein. Eine Hervorhebung muss daher all diese Strukturen umfassen, aber eine Differenzierung ermöglichen. Wie kann diese differenzierende Hervorhebung erfolgen?

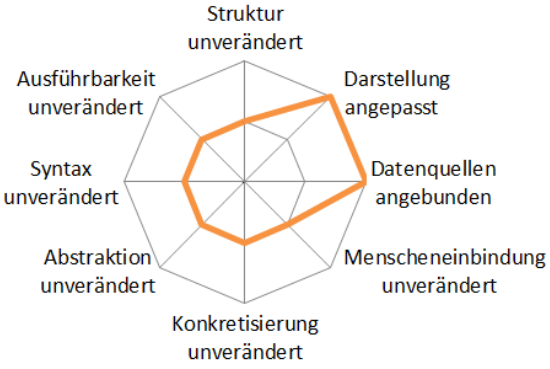


Abbildung 7.7: Klassifikation des Entwurfs.

Lösungsansatz: Abbildung 7.8 zeigt eine Mustersequenz zur Hervorhebung von rr-Fragmenten mit Farbcodes. Zuerst werden bestehende Kennzeichnungen der in einem Prozess enthaltenen rr-Fragmente geladen. Diesem Laden kann eine manuelle Kennzeichnung vorausgehen (M13). Ebenfalls kann eine automatische Erkennung von im Prozess enthaltenen Prozessfragmenten [MLL09] oder von rr-Fragmenten erfolgen [Pos10b]. In der anschließenden Anpassung des Erscheinungsbildes werden alle Aktivitäten und Datenelemente mit einem unscheinbaren hellgrau besetzt, um den Kontrast zu den anderen Prozessstrukturen zu erhöhen (M16, M18). Zur eigentlichen Hervorhebung wird jedem rr-Fragment eine Farbe zugewiesen und dessen Aktivitäten und Datenelemente entsprechend eingefärbt (M16, M18).

Die Definition einer Farbpalette kann als Bestandteil des Informationsdesigns betrachtet werden. Anhand der Farbe kann ein Betrachter unmittelbar erfassen, zu welchem rr-Fragment eine Aktivität oder ein Datenelement gehört.

Im Feinentwurf kann eine Parametrierung durch Auswahl im Annotationsschema oder durch Auswahl einzelner rr-Fragmente erfolgen. Dadurch können z.B. alle rr-Fragmente, die mit der Compliance-Quelle Basel II [Bas06] in Bezug stehen, ausgewählt und hervorgehoben werden. Die Nutzung von Farbpaletten kann zur Gliederung eingesetzt werden, z.B. können rr-Fragmente mit Bezug zu Gesetzen in Nouancen von orange hervorgehoben werden.

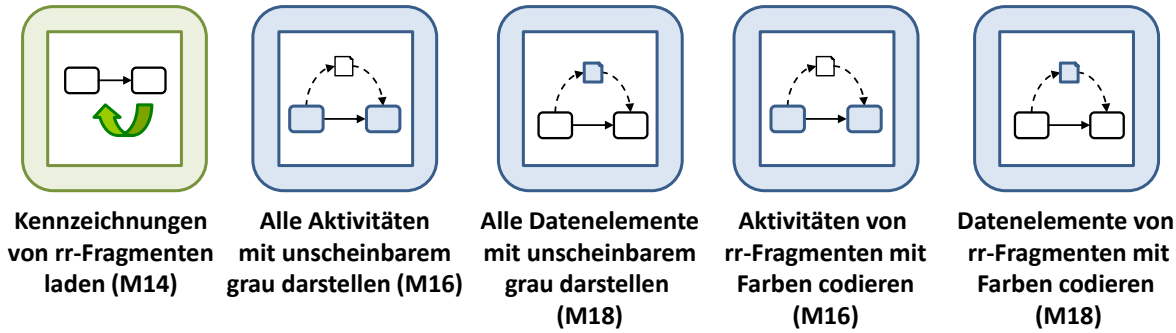


Abbildung 7.8: Mustersequenz zur Hervorhebung von rr-Fragmenten.

Skizze: Die Skizze in Abbildung 7.9 zeigt einen Eingabeprozess (a), der zwei rr-Fragmente enthält, für die eine Kennzeichnung vorliegt (F_1, F_2). Die Farbcodierung stellt die Bestandteile von F_1 gelb dar, die Bestandteile von F_2 blau. Nicht zu rr-Fragmenten gehörende Strukturen treten durch unauffälliges hellgrau in den Hintergrund und erhöhen so den Kontrast.

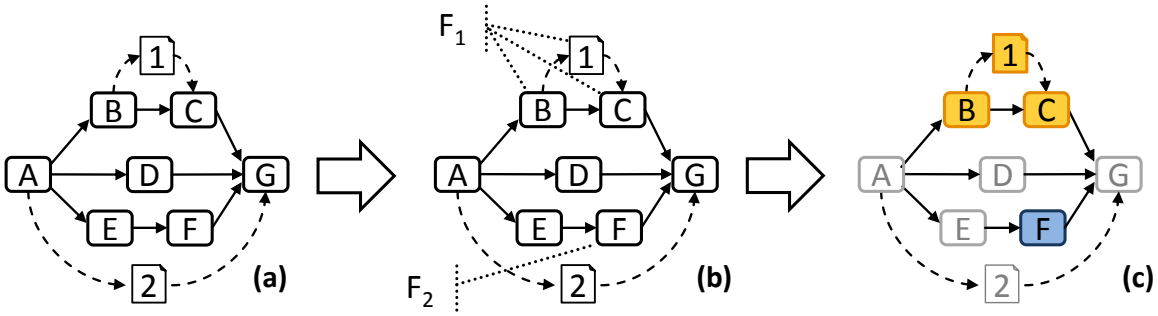


Abbildung 7.9: Skizze für das Hervorheben von rr-Fragmenten.

Beispiele: Ein Kreditvergabeprozess kann ein rr-Fragment enthalten, welches gewährleistet, dass ein potentieller Kreditnehmer eine bestimmte, gesetzlich vorgeschriebene Bedenkzeit hat bevor er einen Kreditvertrag abschließen kann. Wenn eine Betriebsprüfung stattfindet, können die Aktivitäten und Datenelemente des in den Kreditvergabeprozess integrierten rr-Fragments hervorgehoben werden, um dem Prüfer die Realisierung darzustellen. Der Prüfer kann dann nach eigenem Ermessen beurteilen, ob die Compliance-Anforderung angemessen umgesetzt wurde.

7.2 Sichten für Compliance-Prüfungen

7.2.1 Nachvollziehbarkeit einer Sicht

Problem: Eine Sicht kann dazu missbraucht werden, um potentiell regelverletzende oder aus der Perspektive einer bestimmten Interessengruppe fragwürdige Strukturen eines Prozesses bewusst zu verbergen. Dadurch wird die Einsetzbarkeit der Sichtentechnologie im Kontext von Compliance infrage gestellt. Wie könnte eine derartige Manipulation aufgedeckt werden?

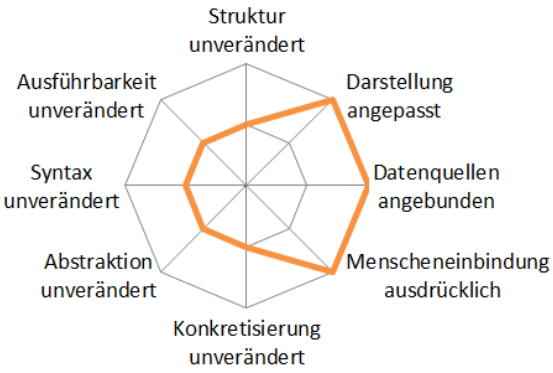


Abbildung 7.10: Klassifikation des Entwurfs.

Lösungsansatz: Um eine bewusste Manipulation durch die Anwendung einer Sicht zu verhindern, müssen die in einer Sicht durchgeführten Transformationen nachvollziehbar sein und deren Auswirkung transparent gemacht werden. Eine Sicht zur Überprüfung von Transformationen kann neben der Nutzung im Compliance-Kontext außerdem zum Testen der Wirkweise einer Sicht oder einer bestimmten Implementierung genutzt werden. Eine Voraussetzung für diese Sicht ist eine Anwendung der Vorgabe zur Nachvollziehbarkeit (M33) in der zu prüfenden Sicht, wie in Abbildung 7.11 dargestellt. Dadurch soll ein revisionssicheres Protokoll über die durchgeführten Transformationen erzeugt werden.

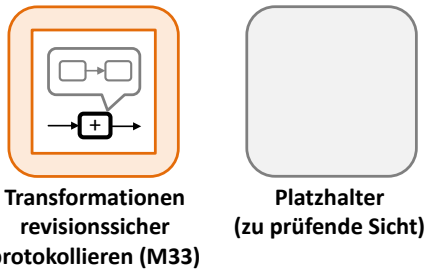


Abbildung 7.11: Voraussetzung für eine Überprüfung von Transformationen.

Die Mustersequenz in Abbildung 7.12 zeigt die Musteranwendungen des Sichtenentwurfs zur Überprüfung von Transformationen. Vorab kann ein Prüfer Strukturen des Prozesses kennzeichnen, die aus seiner Wahrnehmung besondere Relevanz haben (M13) und daher unangetastet bleiben sollten. Eine Anreicherung des Eingabeprozesses mit einem revisionssicheren Transformationsprotokoll (M15) stellt die Information bereit, welche Aktivitäten und Datenelemente in der zu prüfenden Sicht ausgeblendet, aggregiert oder anderweitig modifiziert wurden. Diese angereicherte Information wird in einem angepassten Erscheinungsbild von Aktivitäten (M16), Verbindungen (M17) und Datenelementen (M18) grafisch dargestellt.

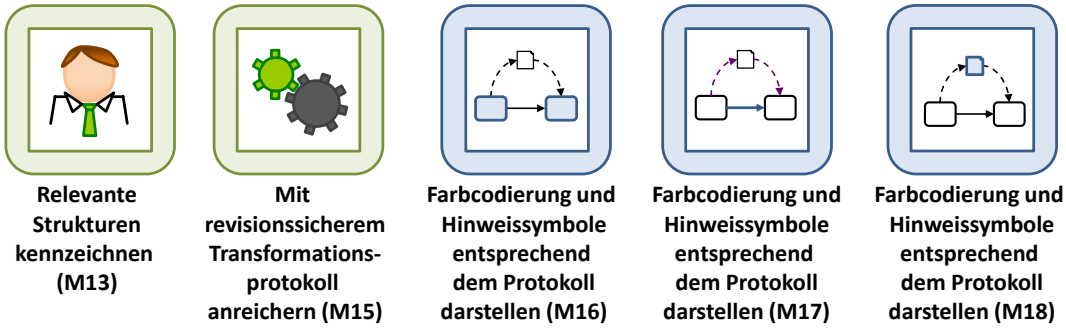


Abbildung 7.12: Mustersequenz für eine Sicht zur Überprüfung von Transformationen.

Eine Spezifikation der vorzunehmenden Anpassungen am Erscheinungsbild wird im Feinentwurf definiert. Tabelle 7.1 zeigt eine beispielhafte Grobspezifikation, die reduzierende Strukturveränderungen abdeckt (M1-M8).

Tabelle 7.1: Grobspezifikation zur Anpassung des Erscheinungsbildes von Aktivitäten, Datenelementen und Verbindungen für eine Sicht zur Überprüfung von Transformationen.

Element	Ausgeblendet	Aggregiert	Modifiziert
Aktivität	Oranger Hintergrund	Blauer Hintergrund; Hinweissymbol für Aggregatnamen	Hinweissymbol zur Warnung
Datenelement			
Verbindung			
Gekennzeichnetes Element	Roter Rahmen	Roter Rahmen	Roter Rahmen

Skizze: Als Informationsgrundlage dient ein Protokoll über durchgeführte Transformationen einer Sicht. Abbildung 7.13 skizziert eine zu überprüfende Sicht. Im Zuge der Erzeugung der Sicht (b) wurden folgende Transformationen protokolliert:

- eine Eigenschaft von Aktivität *A* wurde geändert;
- eine Eigenschaft von Aktivität *B* wurde entfernt;
- die Bedingung der Kontrollverbindung *A*→*B* wurde entfernt;
- Aktivität *E* wurde entfernt;
- Datenelement *1* wurde entfernt;
- die Aktivitäten *G* und *H* wurden aggregiert.

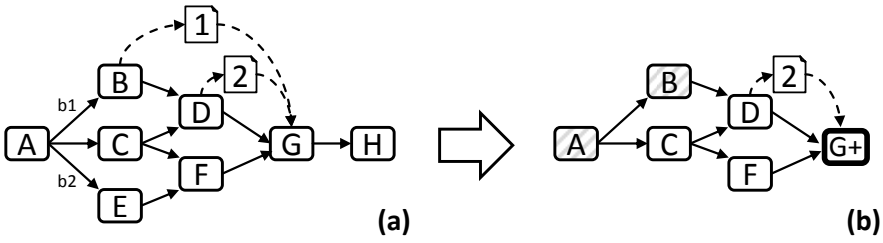


Abbildung 7.13: Skizze für eine Sicht, deren Transformationen überprüft werden sollen.

Die Skizze in Abbildung 7.14 (a) zeigt den mit dem Transformationsprotokoll angereicherten Eingabeprozess. Eine Kennzeichnung Compliance-relevanter Strukturen ist ebenfalls erfolgt (Aktivität *E*; Datenelement *2*). Entsprechend der festgelegten Anpassung des Erscheinungsbildes werden die Aktivitäten, Datenelemente und Verbindungen farblich codiert und mit Hinweissymbolen versehen (b).

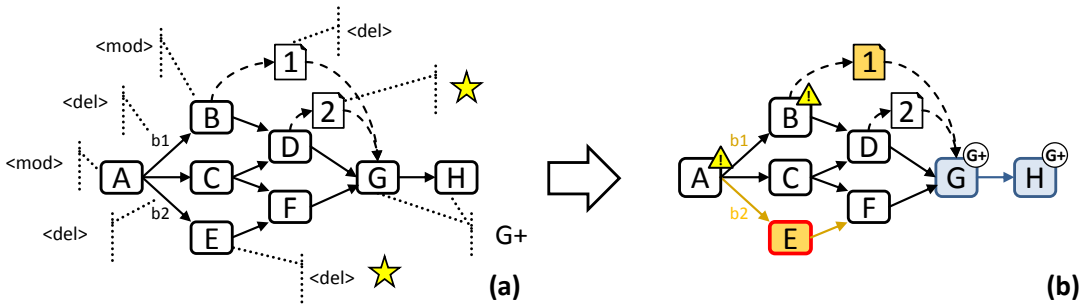


Abbildung 7.14: Skizze für eine Sicht zur Überprüfung von Transformationen.

Beispiele: Wenn die Compliance-Prüfung eines Prozesses auf Basis einer Sicht stattfindet, muss der Prüfer bewerten können, ob die angewendeten Transformationen den Prozess auf unangemessene Weise verändern. Um den Prüfer bei dieser Bewertung zu unterstützen, kann ein schrittweises Vorgehen angewendet werden. In einem ersten Schritt werden alle in der Sicht entfernten Aktivitäten hervorgehoben. Sind diese Aktivitäten nach Ermessen des Prüfers unproblematisch, so können sie für die weitere Betrachtung ausgeblendet werden. Im nächsten Schritt wird analog mit entfernten Datenelementen verfahren, danach mit abstrahierten Aktivitäten und so weiter.

7.2.2 Analyse von Verantwortlichkeiten

Problem: Verantwortlichkeiten sind ein zentraler Aspekt von Compliance. Rollen- und personenbezogene Compliance-Anforderungen umfassen u.a. das 4-Augen-Prinzip, Pflichtenbindung von Aktivitäten an einen Akteur und die Bindung von Aktivitäten an eine Rolle. Die Ansätze zur Analyse von Verantwortlichkeiten im Stand der Technik kommen diesen Fragestellungen durch Modell- und Instanzanalyse nach. Wie könnte dieser Informationskomplex integriert werden?

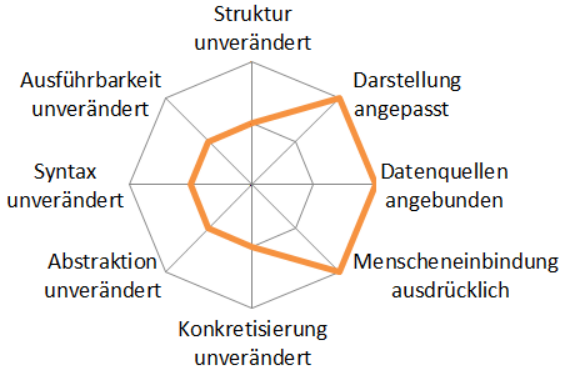


Abbildung 7.15: Klassifikation des Entwurfs.

Lösungsansatz: Die Mustersequenz in Abbildung 7.16 beschreibt den Entwurf einer Sicht für einen Betriebsprüfer zur Analyse der konkreten Verantwortlichkeiten in einer ausgewählten Prozessinstanz. Die Sicht setzt eine Anreicherung des Modells mit personen- und rollenbezogenen Regeln durch den Betriebsprüfer voraus (M13). Vor einer Prüfung müssen die tatsächlichen Akteure einer Prozessinstanz identifiziert und diese Information mit dem Prozessmodell in Verbindung gebracht werden (M15). Diese Information kann aus einer

Automatisierungsumgebung oder aus anderen Informationsquellen bezogen werden, je nachdem wie ein Prozess in dem Unternehmen technisch und organisatorisch realisiert ist.

Um die Betriebsprüfung zu beschleunigen, werden zusätzlich Kontaktinformationen zu den Akteuren und deren Vorgesetzten bereitgestellt (M15). Auf Basis des angereicherten Modells können anschließend automatisierte Regelüberprüfungen durchgeführt und Verletzungen von Regeln aufgezeichnet werden (M14). Sofern Regelverletzungen festgestellt wurden, werden die betreffenden Strukturen hervorgehoben (M16). Ferner werden Kontaktmöglichkeiten zu den Akteuren im Kontext dargestellt (M16).

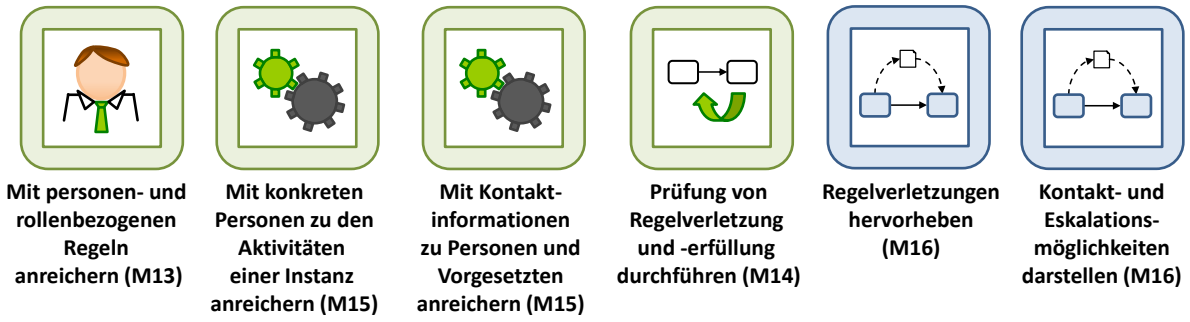


Abbildung 7.16: Mustersequenz für die Analyse von Verantwortlichkeiten.

Skizze: Eine Skizze für die Analyse von Verantwortlichkeiten ist in Abbildung 7.17 dargestellt. Zunächst wird das Prozessmodell mit Vorgaben angereichert (a). In der Skizze werden eine Pflichtenbindung von *A* und *B* (engl. *Binding of Duties, BoD*) sowie eine Pflichtentrennung von *C* und *D* gefordert (engl. *Segregation of Duties, SoD*). Nach dem Anreichern mit den konkreten Akteuren einer Instanz werden diese Vorgaben überprüft und Regelerfüllungen (#t für True) und Regelverletzungen (#f für False) gekennzeichnet, vgl. Abbildung 7.17b. Abschließend werden Regelverletzungen hervorgehoben (SoD bei *C*, *D*) und Kontaktinformationen zu den Akteuren bereitgestellt. In der Skizze wird dies in Form einer Legende dargestellt (c).

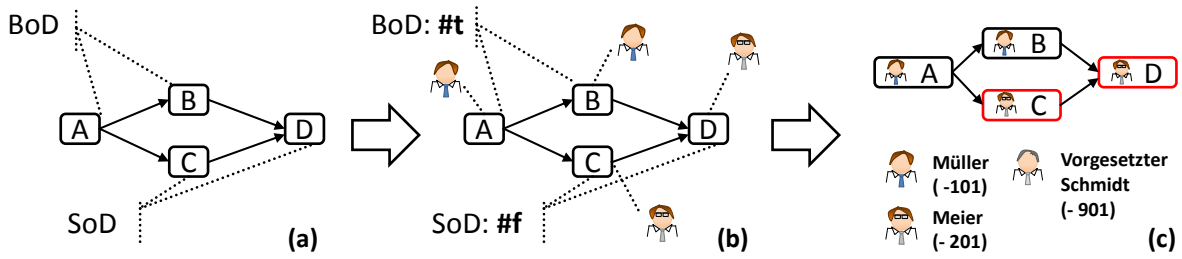


Abbildung 7.17: Skizze für die Analyse von Verantwortlichkeiten.

Beispiele: Die Ansätze von Wolter und Schaad [WS07] und Sadiq et al. [SGN07] ermöglichen eine Modellierung und Visualisierung von rollen- und personenbezogenen Regeln. Diese Ansätze schlagen außerdem eine Visualisierung der Compliance-Regeln im Prozess vor. Auch wurde die Visualisierung von Regelverletzungen im Prozess diskutiert [AW10]. Allerdings stellen diese Arbeiten keinen Bezug zum Ansatz von Sichten auf Prozesse her.

Eine ähnliche Sicht mit konkretem Personenbezug haben Schumm et al. [SLS10a] diskutiert: „Who is in Charge?“ dient dazu, schnell den Zustand einer Prozessinstanz zu erfassen und den Verantwortlichen kontaktieren zu können, wenn Probleme auftreten.

7.3 Übersicht der Verwendung der Sichtenentwurfsmuster

Tabelle 7.2 gibt eine Übersicht, welche Sichtenentwurfsmuster in den Sichtenentwürfen für Compliance eingesetzt werden. Was die Verwendungshäufigkeit der einzelnen Muster in den vorgestellten beispielhaften Sichtenentwürfen angeht, zeichnet sich ein ähnliches Bild ab wie in den Sichten ohne Bezug zu Compliance (vgl. Abschnitt 4.4).

Die Verwendung der Muster zum Entfernen von Strukturen unter Einhaltung der Konsistenz ist ein gängiger Bestandteil in den Sichtenentwürfen. Ebenfalls findet die Anreicherung mit Informationen und die Anpassung des Erscheinungsbildes mehrfach Verwendung. Die übrigen Sichtenentwurfsmuster sind damit nicht obsolet, sondern lediglich im Kontext dieser Entwürfe nicht relevant. Die Einsatzmöglichkeiten jedes einzelnen Musters werden durch die in der Musterbeschreibung aufgeführten Verwendungsbeispiele deutlich.

Tabelle 7.2: Verwendung der Sichtenentwurfsmuster in den Sichtenentwürfen mit Bezug auf Compliance.

			1. Ausschneiden eines rr-Fragments	2. Ausblenden von rr-Fragmenten	3. Hervorheben von rr-Fragmenten	4. Nachvollziehbarkeit einer Sicht	5. Analyse von Verantwortlichkeiten	**Anzahl an Verwendungen**
Strukturveränderung	Reduzierung	1. Aktivitäten entfernen	x	x				2
		2. Verbundene Aktivitäten aggregieren						0
		3. Unverbundene Aktivitäten aggregieren						0
		4. Eigenschaften abstrahieren						0
		5. Datenelemente entfernen						0
		6. Datenelemente aggregieren						0
		7. Pfade aggregieren						0
		8. Strukturen aufräumen	x	x				2
	Erweiterung	9. Aktivitäten einfügen						0
		10. Aktivitäten zerlegen						0
		11. Datenelemente zerlegen						0
		12. Pfade zerlegen						0
Informations-anreicherung		13. Nutzergestützt anreichern	x	x		x	x	4
		14. Statisch anreichern			x		x	2
		15. Dynamisch anreichern				x	x	2
Darstellungsanpassung	Erschei-nungsbild	16. Erscheinungsbild von Aktivitäten anpassen			x	x	x	3
		17. Erscheinungsbild von Verbindungen anpassen				x		1
		18. Erscheinungsbild von Datenelementen anpassen			x	x		2
	Schema	19. Graphschema verwenden						0
		20. Baumschema verwenden						0
		21. Blockschema verwenden						0
		22. Codeschema verwenden						0
		23. Tabellenschema verwenden						0
	Layout	24. Orientierung festlegen						0
		25. Phasenweise auslegen						0
		26. Organisatorisch auslegen						0
		27. Räumlich auslegen						0
		28. Maßgeschneidert auslegen						0
Vorgabe		29. Konsistenz bewahren	x	x				2
		30. Ausführbarkeit bewahren						0
		31. Strukturen bewahren	x					1
		32. Transformationen protokollieren						0
		33. Nachvollziehbarkeit gewährleisten				x		1
		Anzahl verwendeter Muster	**5**	**4**	**3**	**6**	**4**	

KAPITEL 8

INTEGRATIVE TECHNIKEN

Für den Themenkomplex Compliance wurden Sichtenentwürfe vorgestellt, die das Konzept der rr-Fragmente unterstützen und auch weitere Fragestellungen betreffen. Sichten alleine reichen für die Handhabung von rr-Fragmenten allerdings nicht aus. Neben den Sichten zum Ausschneiden, Ausblenden und Hervorheben von rr-Fragmenten sind zwei weitere Techniken erforderlich: die Wiederverwendung (A16) erfordert neben einem Ausschneiden auch die *Integration* eines rr-Fragments in einen bestehenden Prozess; ferner muss nach einer Integration eine *Validierung*, ob die Regelkonformität durch die Integration tatsächlich erreicht wurde (A15), durchgeführt werden.

8.1 Integration von rr-Fragmenten in einen Geschäftsprozess

In Schumm et al. [SLM+10] wurden zwei Möglichkeiten zur Integration von rr-Fragmenten in einen Geschäftsprozess gegenüberstellt: zum einen das *kopierende Einfügen* (engl. *glueing*) und zum anderen das *Einweben* (engl. *weaving*). Das Einweben macht vom Konzept der Aspekt-orientierten Programmierung (AOP) nach Kiczales et al. [KLM+97] Gebrauch. Ein Aspekt wird demnach als übergreifende Funktion verstanden, wie bspw. Protokollierung. Aspekte werden zur Kompilierungszeit oder zur Laufzeit in den Programmcode, bzw. analog in einen Prozess eingewoben. Der Vorteil gegenüber dem kopierenden Einfügen besteht darin, dass der Originalprozess nicht modifiziert wird. Bestehende Einwebetechniken setzen i.d.R. voraus, dass der zu integrierende Aspekt modular aufgebaut ist und an einer definierten Position im Prozess eingewoben werden kann.

Allerdings erfordern rr-Fragmente nach der praktischen Erfahrung aus deren Anwendung im Projekt COMPAS häufig einen tieferen Eingriff in ein Prozessmodell, bspw. wenn das rr-Fragment mehrere Ausgänge enthält. rr-Fragmente sind außerdem nicht zwingend in sich abgeschlossen, bspw. bei Vorhandensein von Platzhaltern. Des Weiteren setzen die Techniken zur formalen Prüfung der Regelkonformität voraus, dass das zu prüfende Prozessmodell vollständig vorliegt und nicht nachträglich modifiziert wird. Aufgrund dieser Rahmenbedingungen ergibt sich die Notwendigkeit für ein kopierendes Einfügen als Integrationstechnik. Die negativen Effekte der Komplexitätssteigerung können durch ein Ausblenden von rr-Fragmenten reduziert werden (vgl. Abschnitt 7.1.2). Das nachfolgend diskutierte kopierende Einfügen von rr-Fragmenten in einen Prozess basiert auf der Arbeit Schumm et al. [SAL+10a].

8.1.1 Vorgehensweise

Eberle [Ebe14] hat verschiedene Ansätze zur Integration von sogenannten Prozessbausteinen skizziert, darunter die Integration auf Basis von Constraints, von Nutzervorgaben und von Zielen. In diesem Abschnitt wird eine Vorgehensweise vorgeschlagen, die als nutzerbasierte Integration betrachtet werden kann, da der Nutzer, d.h. der Modellierer, das in einen Prozess zu integrierende rr-Fragment, den Integrationszeitpunkt und die Einfügepositionen selbst bestimmt. Der Integrationszeitpunkt ist in der verschlagenen Vorgehensweise wohlgemerkt auf die Modellierungsphase beschränkt.

Abbildung 8.1 zeigt die Schritte, die bei der vorgeschlagenen Vorgehensweise zur Integration durchgeführt werden müssen. Die Vorgehensweise ist abstrakt beschrieben und dient zur Diskussion des Prinzips. Die Spezifikation eines konkreten Algorithmus hängt von der Prozesssprache, der Abstraktionsebene wie auch von der Automatisierungsstufe ab. Je höher der Abstraktionsgrad, desto einfacher lässt sich eine Integration technisch unterstützen. Je näher sich Prozess und rr-Fragment an der Ausführung befinden, desto umfangreicher und aufwändiger ist die Integration und dessen technische Unterstützung.

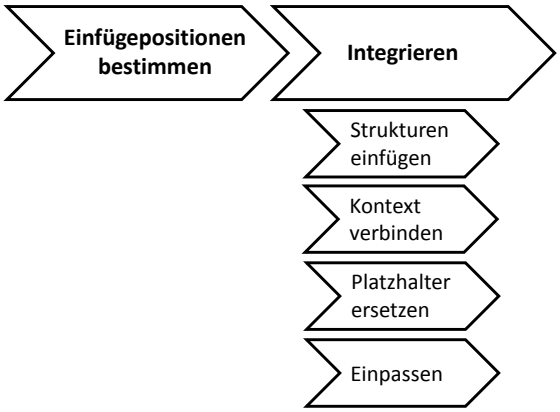


Abbildung 8.1: Vorgehensweise zur Integration von rr-Fragmenten in einen Prozess.

Einfügepositionen bestimmen. Die Integration beginnt mit der Auswahl der Positionen im Prozess, an denen die ein- und ausgehenden Kontrollverbindungen des rr-Fragments mit dem Prozess verbunden werden sollen. Für jede verpflichtend zu verwendende eingehende Kontrollverbindung eines rr-Fragments (vgl. Abschnitt 6.3), d.h. für jeden verpflichtenden Eingang, muss eine Vorgängeraktivität im Prozess bestimmt werden. Analog muss für jeden verpflichtenden Ausgang eine nachfolgende Aktivität bestimmt werden. Entsprechend der Wahl der Positionen müssen die im Prozess bestehenden Kontrollverbindungen neu arrangiert und ggf. aufgebrochen werden. Wenn ein rr-Fragment genau einen Eingang und einen Ausgang besitzt (SESE), kann zur Positionsbestimmung auch eine Kontrollverbindung zwischen zwei Aktivitäten ausgewählt werden, welche durch das rr-Fragment ersetzt werden soll. Um jede freie Kontrollverbindung des rr-Fragments sinnvoll mit dem Prozess zu verbinden, muss der Prozess evtl. angepasst werden. Einfügepositionen ergeben sich daher möglicherweise erst im Verlauf der Einpassung in den Prozess.

Integrieren. Das rr-Fragment wird als Ganzes in den Prozess eingefügt und die Ein- und Ausgänge des rr-Fragments mit den ausgewählten Positionen verbunden. Nicht benötigte optionale Ein- und Ausgänge können entfernt werden.

Beim Integrieren muss der Kontext des rr-Fragments in den Kontext des Prozesses überführt und auf diesen abgestimmt werden. Welche Informationen der Kontext eines rr-Fragmentes genau umfasst, hängt von der angelegten Abstraktionsebene und der verwendeten Prozesssprache ab. Während auf abstrakter Ebene möglicherweise nur Bezeichnungen von Datenelementen zu berücksichtigen sind, können bei einer ausführungsnahen Sprache wie BPEL u.a. Schnittstellenbeschreibungen in WSDL [W3C01], Datenstrukturen von Variablen und Nachrichten in XML Schema, Event Handler, Endpunktreferenzen von Diensten, Mitarbeiterzuweisungen sowie Informationen zur Nachrichtenkorrelationen im Kontext enthalten sein. Ein Teil des Kontextes kann nur lokal für dieses rr-Fragment erforderlich sein, vergleichbar mit dem Kontext eines Sub-Prozesses. Dieser Teil muss lediglich in den Kontext des Prozesses übernommen werden. Konflikte bei bereits bestehenden Bezeichnern wären in diesem Fall durch ein Umbenennen aufzulösen. Allerdings kann sich ein anderer Teil des Kontextes auf Informationen aus dem Prozesskontext beziehen und muss im Zuge der Integration an die im Prozesskontext verwendeten Bezeichner und Typen angeglichen werden. Wenn unterschiedliche Typen im rr-Fragment und im Prozess zur Darstellung derselben Informationen verwendet werden, so muss eine Anpassung am Prozesskontext oder am rr-Fragment vorgenommen werden. Bspw. kann eine Entscheidung als Boolean mit `true` und `false` oder als String mit `yes` und `no` abgebildet werden.

Nach dem Einfügen des rr-Fragments kann der Prozess Platzhalter enthalten, die im weiteren Verlauf der Integration durch konkrete Prozessstrukturen ersetzt werden müssen. Die Integration eines rr-Fragments kann verschiedene Anpassungen des Prozesses erfordern. Sind bspw. mehrere Ausgänge in einem rr-Fragment enthalten, kann dies eine Änderung der bisherigen Ablauflogik erfordern. Bspw. kann eine neue Ausnahmebehandlung notwendig sein, wenn ein Compliance-Beauftragter eingebunden werden muss. Das Einpassen des rr-Fragments in den Prozess umfasst des Weiteren die prozessspezifische Bindung freier Parameter und die Behebung von Konflikten, die nach der Kontextverbindung bestehen.

Nachbedingungen. Aus der Anwendung dieser Vorgehensweise resultiert ein konsistenter Prozess, in den ein rr-Fragment integriert wurde. Alle Ein- und Ausgänge des rr-Fragments sind mit Prozessstrukturen verbunden und optionale Ein- und Ausgänge sind verbunden oder entfernt. Der Kontext des rr-Fragments ist im Prozesskontext aufgegangen und freie Parameter wurden gesetzt. Vorhandene Platzhalter wurden durch konkrete Prozessstrukturen ersetzt. Das rr-Fragment wurde damit vollständig integriert und kann nur durch die Identifikationsmerkmale an seinen Bestandteilen von den anderen Prozessstrukturen unterschieden werden. Diese Vorgehensweise steht im Kontrast zu der von Eberle et al. [ELS+10] diskutierten Kompositionstechnik, bei der unverbundene Ein- und Ausgänge weiterbestehen können, um daran weitere Fragmente anzuhängen. Eberle et al. konstruieren einen ganzen Prozess aus einzelnen Fragmenten und führen auch einen nur teilweise ausspezifizierten Geschäftsprozess aus. Dieser Ansatz erfordert allerdings eine erweiterte Laufzeitumgebung und ist zudem nicht mit einem präventiven Management von Compliance vereinbar.

8.1.2 Beispiel

Die Auflage, für einen bestimmten Vorgang eine Genehmigung einzuholen, ist eine häufig auftretende Compliance-Anforderung [SLM+10]. Genehmigungen sind aber auch unabhängig von Compliance häufig in Geschäftsprozessen zu finden [TRI09]. Eine Genehmigung kann als rr-Fragment konstruiert werden, wie in Abbildung 8.2 dargestellt. Das rr-Fragment wird durch einen Eingang und zwei Ausgänge charakterisiert. Ein weiterer Eingang ist optional. Die Ausgänge können semantisch in eine Genehmigung und eine Ablehnung unterschieden werden. Der Kontext des rr-Fragments besteht aus zwei Datenelementen, die den zu prüfenden Sachverhalt und das Ergebnis der Prüfung repräsentieren. Als Parameter ist die Rolle des Prüfers ausgewiesen.

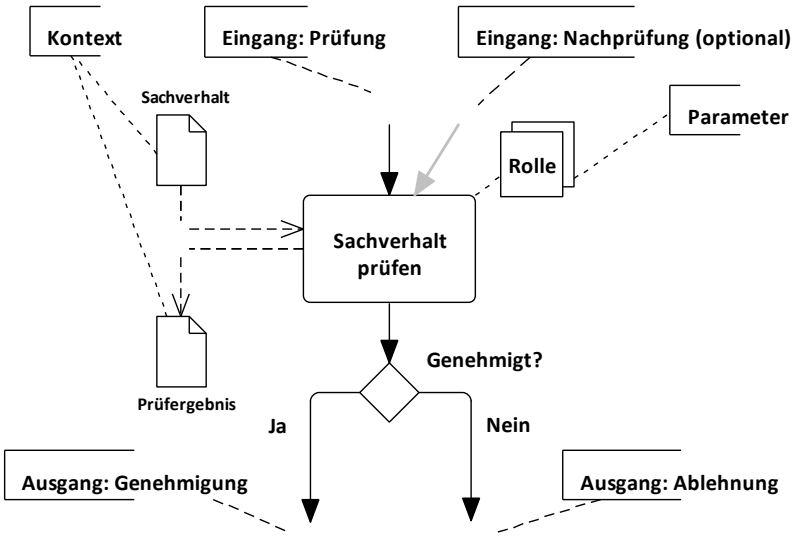


Abbildung 8.2: Bestandteile eines rr-Fragments zur Realisierung von Genehmigungsanforderungen, angepasst nach [SAL+10a].

Als Anwendungsbeispiel dient ein einfacher Prozess zur Kleinkreditvergabe, vgl. Abbildung 8.3 (b). Der Prozess sieht vor, einen Kreditantrag zu erfassen und zu veranlassen, wenn eine Wirtschaftsauskunftei den Antragsteller als kreditwürdig einstuft. Andernfalls wird der Antrag abgelehnt.

Angenommen es wird eine unternehmensinterne Richtlinie erlassen, die eine Genehmigung durch einen internen Sachbearbeiter vorschreibt. Für diese Compliance-Anforderungen kann ein rr-Fragment als Compliance-Instrument eingesetzt werden, vgl. Abbildung 8.3 (a). Die Integration erfolgt nach den zuvor beschriebenen Schritten.

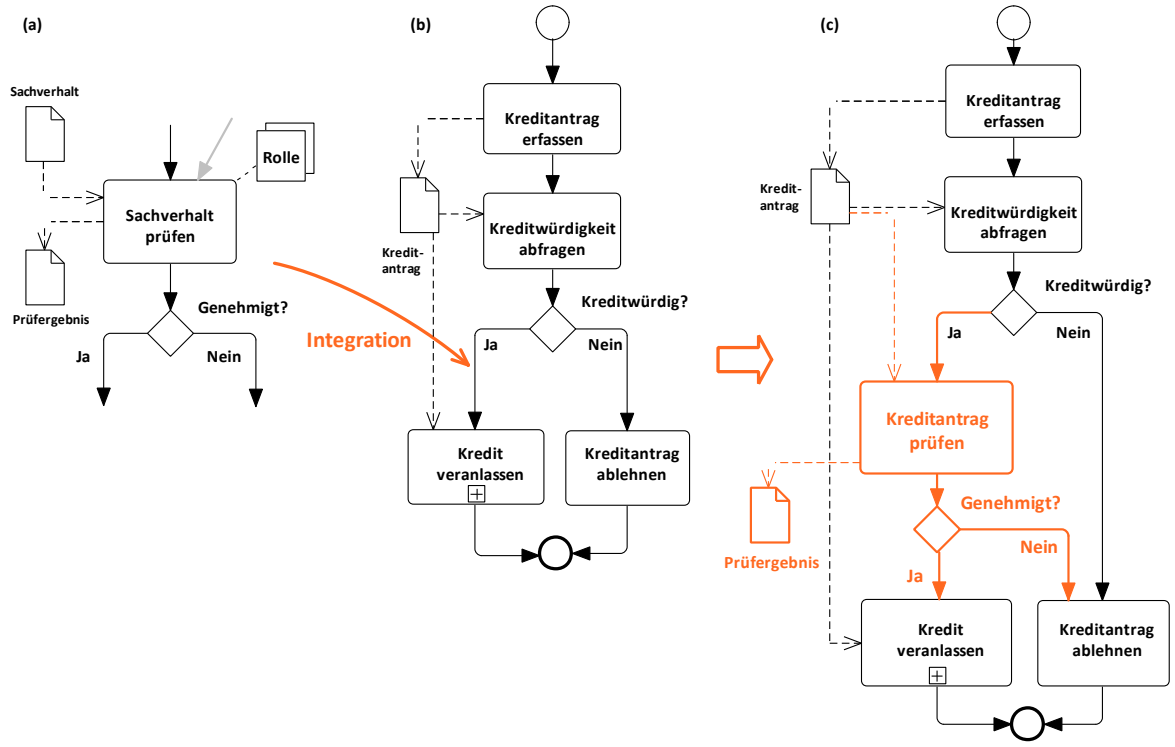


Abbildung 8.3: Ein rr-Fragment (a) wird in einen Geschäftsprozess (b) integriert und dieser Geschäftsprozess somit um regelrealisierendes Verhalten erweitert (c), angepasst nach [SAL+10a].

Einfügepositionen bestimmen. Als Einfügeposition kommen zwei Möglichkeiten in Betracht: vor dem Abfragen der Kreditwürdigkeit bei der Wirtschaftsauskunftei oder danach. Angenommen diese Abfrage verläuft automatisiert und stellt einen zu vernachlässigenden Kostenfaktor dar. Um die Personalkosten gering zu halten, wird die Einfügeposition daher so gewählt, dass nur bei vorermittelter Kreditwürdigkeit des Antragstellers eine Genehmigung durch einen internen Sachbearbeiter erfolgt.

Integrieren. Die ursprüngliche Kontrollverbindung zwischen der zuvor abgefragten Kreditwürdigkeit und dem Veranlassen des Kredits wird aufgebrochen und der Eingang des rr-Fragments entsprechend verbunden. Der auf die Genehmigung des Sachverhalts folgende Ausgang stellt die Verbindung zum Veranlassen des Kredites her. Der andere Ausgang wird mit der Ablehnung des Kreditantrags verbunden. Die optionale Nachprüfung ist nicht erforderlich und wird entfernt. Der Rollenparameter wird entsprechend der Compliance-Anforderung als Sachbearbeiter gesetzt. Die Beschriftung der Aktivität *Sachverhalt prüfen* wird durch eine konkretisierte Beschriftung ersetzt. Durch die Nutzung eindeutiger Identifikationsmerkmale bleibt die Nachvollziehbarkeit trotz dieser Anpassung erhalten. Bei der Verbindung der Kontexte wird das Datenelement *Sachverhalt* aus dem rr-Fragment durch den bereits im Prozess bestehenden *Kreditantrag* ersetzt. Das neue Datenelement *Prüfergebnis* wird in den Prozesskontext übernommen.

Die Integration kann mit einer Anpassung der Darstellung unterstützt werden. Eine temporäre Hervorhebung, wie in Abbildung 8.3 (c) verwendet, hilft bei der Unterscheidung der Bestandteile des rr-Fragments von dem ursprünglichen Prozess. In einer anschließenden Einpassung könnte der Prozess dahingehend erweitert werden, auch das Prüfergebnis

weiterzuverarbeiten oder abzulegen. Der resultierende Prozess ist in Abbildung 8.3 (c) mit hervorgehobenem rr-Fragment dargestellt.

8.1.3 Diskussion

Aus verschiedenen Gründen besteht die Möglichkeit, dass ein rr-Fragment nach dessen Integration in einen Prozess nicht den Zweck der Regelrealisierung erfüllt, für den es ursprünglich konstruiert wurde. Es kann z.B. an einer falschen Stelle im Prozess eingefügt worden sein, annotierte Constraints wurden vom Modellierer nicht beachtet, Parameter falsch gesetzt, die Verbindung des Kontextes war fehlerhaft oder nachfolgende Anpassungen am Prozess haben den ursprünglichen Aufbau des rr-Fragments zerstört. Um dem entgegenzuwirken, ist eine Verbindung zur formalen Prüfung der Regelkonformität erforderlich.

8.2 Prüfung der Integration durch Verifikationstechniken

Ein korrektes Design eines rr-Fragments kann für sich alleine nicht gewährleisten, dass ein Prozess regelkonform ist nachdem das rr-Fragment in den Prozess integriert wurde. Daher ist eine Prüfung der Integration durch Verifikationstechniken erforderlich.

Die in diesem Abschnitt beschriebene Vorgehensweise zur Prüfung der Integration eines rr-Fragments basiert auf einer Zusammenarbeit im Rahmen des Forschungsprojekts COMPAS[7], deren Ergebnisse von Schumm et al. [STK+10] vorgestellt wurden. In dieser Zusammenarbeit wurde der Ansatz der rr-Fragmente als Verhaltensinstrument mit einem Ansatz zur formalen Beschreibung von Anforderungen in Verbindung gebracht und durch den Einsatz von Verifikationstechniken als Prüfungsinstrument abgesichert. Die Arbeit [STK+10] beschreibt außerdem eine Werkzeugkette, die diese Vorgehensweise unterstützt. rr-Fragmente dienen der Anreicherung mit regelkonformem Verhalten, während die Formalisierung und die Verifikation den Compliance-Nachweis einer Prozessanpassung liefern. Diese Kombination stellt eine nach bestem Wissen neuartige und schlüssige Herangehensweise zur Unterstützung der Herstellung von Regelkonformität zur Entwurfszeit dar (engl. *Compliance by design*).

8.2.1 Vorgehensweise

Die vorgeschlagene Vorgehensweise zur geprüften Integration von rr-Fragmenten in einen Geschäftsprozess ist in Abbildung 8.4 dargestellt. Auf die Darstellung in Chevron wurde hier verzichtet, um größere Präzision bei der Beschreibung zu ermöglichen. Dies betrifft die beiden Verzweigungen und die Schleife, die in der Vorgehensweise enthalten sind.

[7] http://www.future-internet.eu/activities/fp7-projects.html

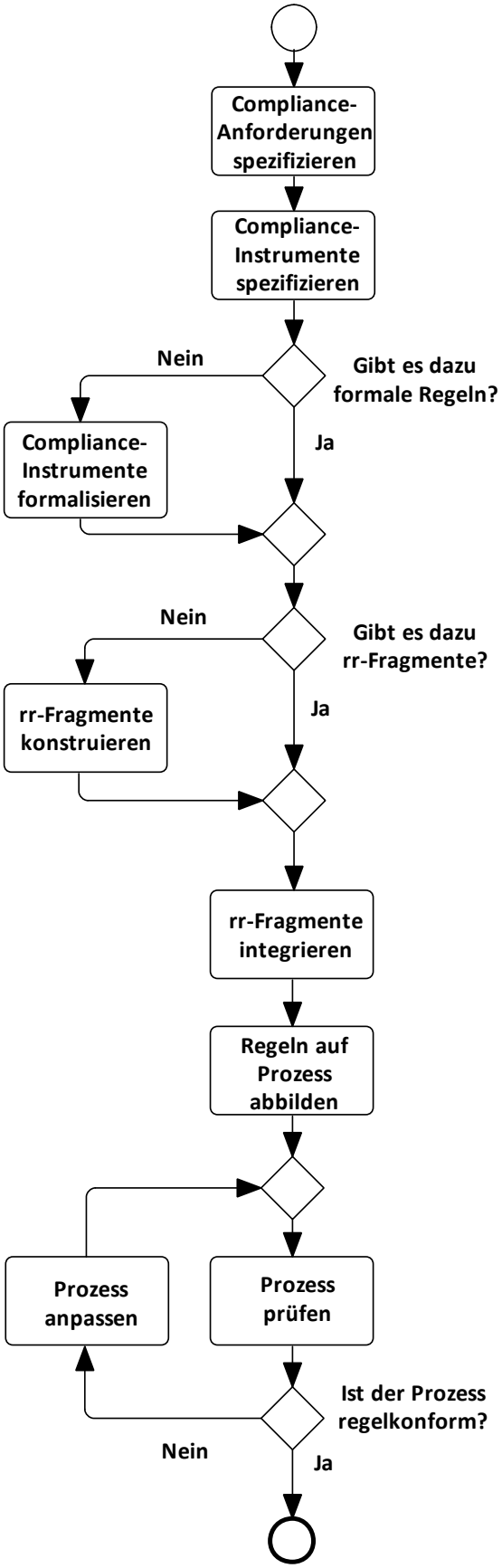


Abbildung 8.4: Vorgehensweise zur Prüfung der Integration eines rr-Fragments, modelliert nach [STK+10].

Compliance-Anforderungen spezifizieren. In der Vorgehensweise zur Unterstützung der Regelkonformität eines Geschäftsprozesses wird vorausgesetzt, dass eine Menge von widerspruchsfreien Compliance-Anforderungen besteht (vgl. Voraussetzung V2). Allerdings ist die Art und Weise wie diese Compliance-Anforderungen beschrieben sind von großer Bedeutung. Compliance-Anforderungen sollten ebenso wie andere Anforderungen so klar und präzise wie möglich formuliert sein und auf subjektive oder vage Umschreibungen verzichten. Wenn Freiheiten und Interpretationsmöglichkeiten in der Beschreibung der Compliance-Anforderungen bestehen, kann dies eine unwirksame Umsetzung zur Folge haben. Andererseits erlauben Freiheitsgrade, die Anforderungen gemäß der vorhandenen IT-Infrastruktur und den organisatorischen Möglichkeiten entsprechend umzusetzen.

Compliance-Instrumente spezifizieren. Ein Compliance-Instrument (engl. *Compliance control*) ist ein Begriff zur Beschreibung einer Einflussmöglichkeit, mit der eine Compliance-Anforderung umgesetzt werden kann. Wie in der Beschreibung der Compliance-Terminologie in Abschnitt 2.3.1 erwähnt, sind die Einflussnahme auf Prozesse und die Durchführung von Überprüfungen gängige Arten von Compliance-Instrumenten. Ein Compliance-Instrument wird in Prosa beschrieben und lässt damit ggf. signifikanten Freiraum bei der konkreten Realisierung. Die spezifizierten Compliance-Anforderungen werden in ein oder mehrere Compliance-Instrumente zur Erfüllung der gestellten Anforderungen überführt. Zur Gewährleistung der Nachvollziehbarkeit von der Compliance-Quelle bis hin zur Implementierung in einem Geschäftsprozess ist eine Dokumentation der getroffenen Entscheidungen und Interpretationen essentiell. Diese Aktivität kann ebenso wie die Spezifikation von Compliance-Anforderungen juristische Fachkenntnisse erfordern. Aufgrund der Vielfalt an möglichen Compliance-Quellen werden diese Fachkenntnisse jedoch nicht immer benötigt.

Compliance-Instrumente formalisieren. Für neu spezifizierte Compliance-Instrumente liegen noch keine formalen Regeln zur Wiederverwendung vor und daher müssen diese neu erstellt werden. Im Stand der Technik wird häufig Lineare Temporale Logik (LTL) eingesetzt, um zu prüfende Aussagen mit Bezug auf den Kontrollfluss eines Geschäftsprozesses formal zu beschreiben [Sch14]. LTL erweitert die Aussagenlogik um temporale Operatoren, die zur Vorgabe einer Reihenfolge von Aktivitäten eingesetzt werden können. Compliance-Anforderungen können allerdings in komplexen Compliance-Instrumenten resultieren, deren formale Beschreibung umfassende Kenntnisse in dem eingesetzten Formalismus erfordern. Dieser Umstand unterstreicht die neuartigen Aufgaben des Modellierens regelkonformer Strukturen und der statischen Verifikation im Prozesslebenszyklus (vgl. Abschnitt 6.2), welche die nötige Expertise abdecken.

rr-Fragmente konstruieren. Liegen für ein Compliance-Instrument keine rr-Fragmente vor, müssen diese neu konstruiert werden. Nach der Darstellung des Lebenszyklus eines rr-Fragments von Schumm et al. [SLM+10] kann dies auf zwei Arten erfolgen. Entweder wird ein rr-Fragment aus einem regelkonformen Geschäftsprozess ausgeschnitten; für diesen Ansatz wurde in Abschnitt 7.1.1 ein Sichtenentwurf diskutiert. Die andere Möglichkeit ist die Modellierung von Grund auf, basierend auf der Beschreibung des Compliance-Instruments.

Das rr-Fragment wird anschließend parametrisiert, mit entsprechender Beschreibung versehen, mit dem Compliance-Instrument und der formalen Regel verknüpft und zur Wiederverwendung in eine Bibliothek eingestellt. Wie Holschke et al. [HRL09] allerdings feststellen, hat die Granularität einzelner Teile Einfluss auf die Effektivität der Wiederverwendung von Prozesswissen. Daher sollte die Granularität eines rr-Fragments zur Granularität der Prozesse passen. Die Abbildung einer formalen Regel auf ein rr-Fragment setzt zwingend voraus, dass beides im Abstraktionsgrad zusammenpasst.

rr-Fragmente integrieren. Die Integration eines rr-Fragments in einen Prozess umfasst die Bestimmung von Einfügepositionen, das Einfügen der Strukturen des rr-Fragments, die Verbindung der Kontexte, das Parametrieren und das Ersetzen von Platzhaltern. Im Zuge der Einpassung wird das eingefügte rr-Fragment an seinen Verwendungszweck eingepasst, d.h. konkretisiert und verfeinert. Die Schritte der Integration sind in Abschnitt 8.1 beschrieben.

Regeln auf Prozess abbilden. Die Aussagevariablen der Regeln müssen auf die Strukturen des Prozesses abgebildet werden, damit die Gültigkeit der Regeln geprüft werden kann. Die im Stand der Technik häufig verwendete Zuordnung auf Basis von Beschriftungen (engl. *label-based matching*) kann in komplexen Prozessen zu mangelnder Eindeutigkeit führen, bspw. wenn mehrere Schritte zur Genehmigung vorliegen. Dem Mapping von Aussagevariablen auf Prozessstrukturen wird daher die Annahme zugrunde gelegt, dass den Elementen des rr-Fragments und den Aussagenvariablen der formalen Regel eindeutige Identifikationsmerkmale zugeordnet sind, wodurch eine eindeutige Korrelation möglich ist.

Prozess prüfen. Die Prüfung der formalen Regeln gegen den mit rr-Fragmenten angereicherten Prozess erfolgt durch Model Checking, dem im Stand der Technik gängigen Ansatz der Compliance-Prüfung. Dazu müssen der Prozess und die Regeln in die Formate überführt werden, die von dem spezifischen Model Checking System verarbeitet werden können. Wie Kokash [STK+10] beschreibt, umfassen Prozessmodelle nur selten alle Informationen, die für eine vollautomatische Überprüfung erforderlich sind. Daher wird die Annahme getroffen, dass ein Spezialist die erforderliche Verfeinerung vornimmt und anschließend die Prüfung durchführt.

Diesem Umstand ist im angepassten Lebenszyklus durch die zusätzliche Aufgabe der statischen Verifikation Rechnung getragen (vgl. Abschnitt 6.2). Sind alle geprüften Regeln erfüllt, ist der Prozess zur Entwurfszeit regelkonform. Ist eine Regel nicht erfüllt, liefert ein Model Checker ein Gegenbeispiel, d.h. einen möglichen Ausführungszustand in dem die Regel verletzt wird. Das Gegenbeispiel kann dem Modellierer als Hilfestellung zum Verständnis der Regelverletzung dienen. Eine Hervorhebung regelverletzender Pfade ist an dieser Stelle bereits Stand der Technik. Eine Hervorhebung der Bestandteile der integrierten rr-Fragmente ist eine weiterführende Unterstützung (vgl. Abschnitt 7.1.3).

Prozess anpassen. Ergibt die Prozessprüfung, dass Regeln nicht erfüllt sind, muss der Prozess angepasst werden. Die Anpassung kann sowohl die Bestandteile der eingefügten rr-Fragmente als auch andere Strukturen im Prozess betreffen. Anpassungen können bspw. ein Verschieben von Prozessstrukturen umfassen, etwa wenn die Bestandteile eines rr-Fragments

falsch platziert wurden. Auch kann eine Belegung von Parametern mit falschen Werten zu einer Regelverletzung führen, wie bspw. wenn eine andere Rollenzuweisung verwendet wird als in einer Regel gefordert ist. Die Integration von Compliance in einen Geschäftsprozess kann durch die neuen Prozessstrukturen und erforderlichen Einpassungen tiefgreifende Auswirkungen auf die ursprüngliche Struktur der Ablaufplanung haben.

8.2.2 Beispiel

Die einzelnen Schritte werden nachfolgend am Beispiel des Kleinkreditvergabeprozesses erläutert, der auch bei der Integration von rr-Fragmenten verwendet wurde, vgl. Abschnitt 8.1.

Compliance-Anforderungen spezifizieren. Für den Prozess der Kleinkreditvergabe ergibt sich aus der *Compliance-Quelle* unternehmensinterner Beschlüsse die neue *Compliance-Anforderung*, eine Genehmigung eines Kleinkreditantrags durch einen internen Sachbearbeiter durchzuführen. Diese Anforderung entschärft das *Compliance-Risiko*, dass Kredite vergeben werden, die nicht zurückgezahlt werden.

Compliance-Instrumente spezifizieren. Da die Compliance-Anforderung bereits sehr spezifisch und auf einen bestimmten Prozess beschränkt ist, unterscheidet sie sich praktisch nicht von dem *Compliance-Instrument* der Genehmigung eines Kleinkreditantrags durch einen internen Sachbearbeiter. Eine Compliance-Anforderung, die bspw. eine Pflichtentrennung bei jeglicher Kreditvergabe fordert, würde hingegen in zahlreichen Compliance-Instrumenten resultieren, die zudem in verschiedenen Prozessen berücksichtigt werden müssten.

Compliance-Instrumente formalisieren. Eine Regel für das Compliance-Instrument der Genehmigung durch einen Sachbearbeiter kann wie folgt formal beschrieben werden. Zunächst kann man interpretieren, dass ein Kredit nur dann veranlasst werden darf, wenn dafür die notwendigen Voraussetzungen erfüllt sind. Dazu kann eine Implikation $A \rightarrow B$ aufgestellt werden, wobei *A* der Aktivität *KreditVeranlassen* entspricht und *B* einer Konjunktion aus den Voraussetzungen *KreditantragPrüfen*, *Sachbearbeiter* und *Genehmigt*. Mit anderen Worten soll ein Kredit nur dann veranlasst werden, wenn die Prüfung durch einen Sachbearbeiter durchführt und genehmigt wurde.

Durch eine Äquivalenzumformung können die Aussagenvariablen in dieser Regel in die antizipierte Reihenfolge der Ausführung gebracht werden. Dies ist für den Ansatz der formalen Beschreibung nicht erforderlich, dient aber bei Abbildung von Aussagen auf Prozesselemente dem zeitlichen Verständnis:

$$A \rightarrow B \equiv \neg A \vee B \equiv B \vee \neg A$$

Dadurch ergibt sich folgende formale Regel:

$$(\text{KreditantragPrüfen} \wedge \text{Sachbearbeiter} \wedge \text{Genehmigt}) \vee \neg \text{KreditVeranlassen}$$

Neben aussagenlogischen Operatoren können temporale Operatoren verwendet werden, um einen Zeitbezug zu beschreiben. In LTL etwa stehen dazu die Operatoren *G* (globally /

immer), *F* (finally / irgendwann) und *U* (until / bis) zur Verfügung. *G* sagt im Kontext von dem Beispiel aus, dass die Regel in jedem Zustand des Prozesses erfüllt sein muss:

$$G((\text{KreditantragPrüfen} \wedge \text{Sachbearbeiter} \wedge \text{Genehmigt}) \vee \neg \text{KreditVeranlassen})$$

Ebenso wie ein rr-Fragment kann auch eine Regel zur Formalisierung eines Compliance-Instruments auf verschiedenen Abstraktionsebenen spezifiziert werden, um deren Wiederverwendbarkeit zu erhöhen. Zu der konkreten Regel kann eine abstraktere Regel formuliert werden, die sich auf derselben Abstraktionsebene wie das in Abbildung 8.2 gezeigte rr-Fragment befindet:

$$G((\text{SachverhaltPrüfen} \wedge \text{Rolle} \wedge \text{Genehmigt}) \vee \neg \text{Veranlassen})$$

rr-Fragmente konstruieren. Zur Genehmigung von Vorgängen wurde in Abbildung 8.2 bereits ein rr-Fragment vorgestellt. Die Abbildung der Aussagen der abstrakten Regel auf Strukturen dieses rr-Fragments ist in Abbildung 8.5 dargestellt. Man beachte, dass die Abbildung unvollständig ist, da die in der Regel referenzierte Aktivität *Veranlassen* der zu genehmigenden Maßnahme erst noch auf den konkreten Prozess abgebildet werden muss, in den das rr-Fragment integriert wird.

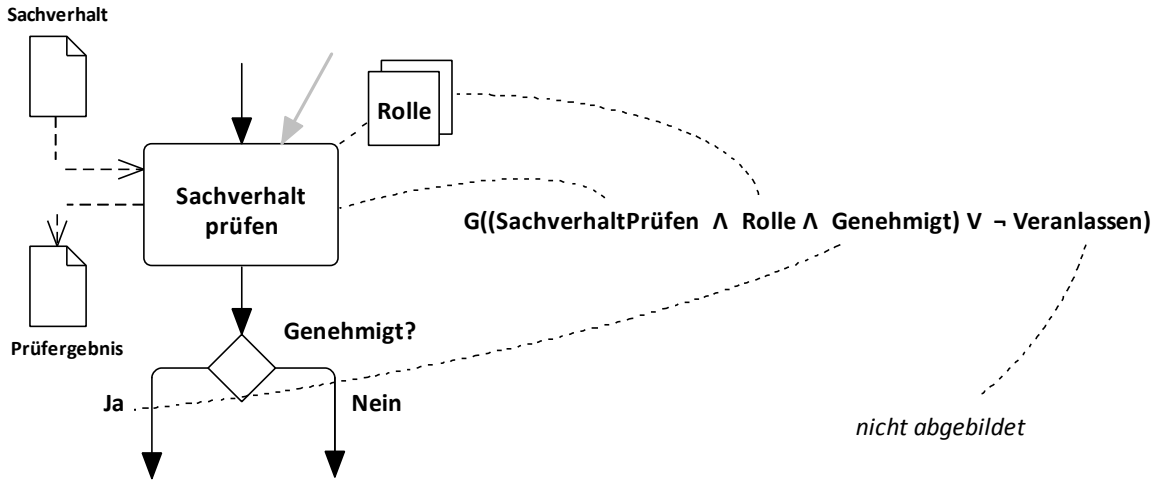


Abbildung 8.5: Abstraktes rr-Fragment zur Realisierung von Genehmigungsanforderungen mit den Beziehungen zu den Aussagenvariablen der abstrakten Regel.

rr-Fragmente integrieren. Das rr-Fragment wird wie in Abschnitt 8.1.2 beschrieben in den Prozess integriert und entsprechend den Nutzungsanforderungen konkretisiert.

Regeln anwenden. Entsprechend der Konkretisierung des rr-Fragments bei der Integration muss die Regel in ihrer konkreten Form angewendet werden. Bevor die Regel gegen den Prozess geprüft werden kann, müssen außerdem nicht abgebildete Aussagenvariablen zu bestehenden Prozessstrukturen in Bezug gesetzt werden. Im Beispiel wird die in der formalen Regel negierte Aussagenvariable *KreditVeranlassen* auf den Sub-Prozess *Kredit veranlassen* abgebildet. Der Prozess mit der angewendeten Regel ist in Abbildung 8.6 dargestellt.

Man beachte an dieser Stelle, dass die entworfene Regel keine Aussage darüber macht, ob ein Kreditantrag abgelehnt werden soll, wenn die Voraussetzungen nicht erfüllt sind. Dazu müsste die Regel entsprechend eine Aussagenvariable *KreditantragAblehnen* einbeziehen,

die auf die Aktivität *Kreditantrag ablehnen* im Prozess abgebildet werden kann. Dieser Umstand zeigt beispielhaft den Interpretationsspielraum, der bei der Umsetzung einer Compliance-Anforderung und deren Berücksichtigung in einem Prozess bestehen kann.

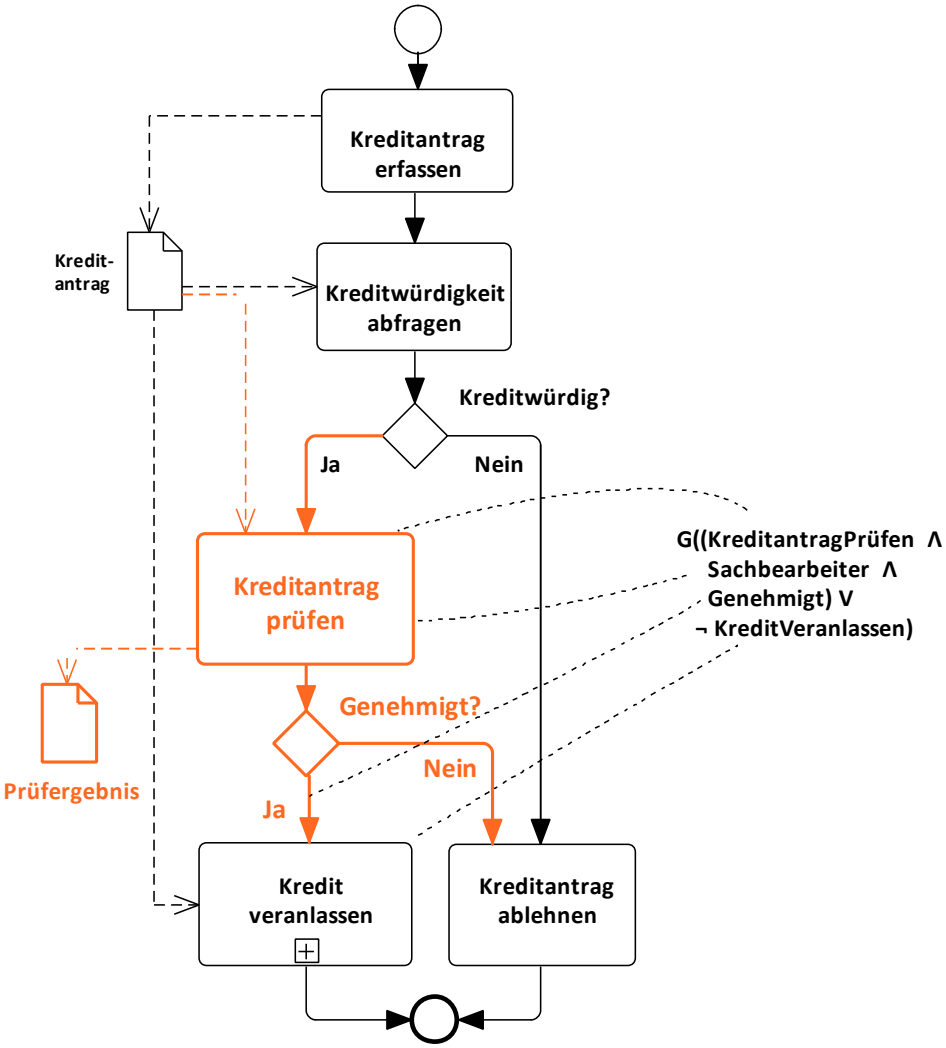


Abbildung 8.6: Beispielprozess nach Integration und Konkretisierung des rr-Fragments mit den Beziehungen zu den Aussagenvariablen der konkreten Regel.

Prozess prüfen. Zur konkreten Ausgestaltung der Prozessprüfung schlägt Kokash [STK+10] die Nutzung von Reo vor, einer Sprache zur formalen Beschreibung von komplexen Abläufen [Arb04]. Mit bestehenden Werkzeugen können Prozesse aus den Sprachen BPMN, BPEL oder UML AD in Reo Networks überführt werden [CKA10]. Die Eclipse Coordination Tools [AKM+08] dienen als Plattform zur Verfeinerung überführter Prozesse. Diese Plattform bietet weiterhin die Möglichkeit, verschiedene Werkzeuge zum Model Checking einzubinden. Dazu gehört u.a. Vereofy [Ver10], mit dem in LTL formulierte Regeln überprüft werden können. Weitere Details zur Prozessprüfung können in dem Artikel von Schumm et al. [STK+10] nachgelesen werden.

Prozess anpassen. Eine weitere Anpassung des Prozesses ist zur Erfüllung der beispielhaften Anforderung nicht erforderlich, da die Regel bereits erfüllt ist.

8.2.3 Diskussion

Änderungsmanagement. Neben der Berücksichtigung neuer Compliance-Anforderungen muss auch eine Handhabungsmöglichkeit für deren Änderung oder Entfall geschaffen werden (engl. *change management*). Bei einer Änderung der Compliance-Instrumente müssen Anpassungen an den formalen Regeln als auch an den rr-Fragmenten vorgenommen werden. Die angepasste Regel wird auf den Prozess angewendet und im Prozess müssen die Konstruktionsänderungen des rr-Fragments entsprechend nachgezogen werden. Um Änderungen zu unterstützen, können die Elemente des rr-Fragments im Prozess hervorgehoben und diejenigen Elemente grafisch gekennzeichnet werden, die sich im Vergleich zur Vorversion geändert haben (M16, M17, M18). Ansätze zur Synchronisation können an dieser Stelle ebenfalls von Nutzen sein, wie etwa auf Basis einer Änderungswarteschlange [WAK+11]. Entfällt eine Anforderung, können die entsprechenden Strukturen aus dem Prozess entfernt werden. Allerdings müssen nach der Anpassung *alle* bestehenden Regeln geprüft werden, da deren Gültigkeit durch die Änderungen am Prozess nicht mehr gewährleistet sein kann.

Ausdrucksstärke. Der Ansatz ist durch die Ausdrucksstärke seiner Komponenten in seiner Mächtigkeit eingeschränkt. Zum einen können mit rr-Fragmenten nur solche Anforderungen erfüllt werden, die sich auf den Kontrollfluss, den Datenfluss und die Datenelemente des Prozesses beziehen. Anforderungen an die Systeme und Menschen, die in einen Prozess involviert sind, wie auch nicht-funktionale Anforderungen erfordern weitere Compliance-Instrumente. Weiterhin kann der verwendete Formalismus zur Definition von Regeln eine Einschränkung darstellen. LTL ist bspw. für die formale Beschreibung von Regeln hinsichtlich des Kontrollflusses geeignet, nach Schleicher et al. [SGL+11] allerdings unpassend für datenbezogene Anforderungen. Der Mächtigkeit des Formalismus ist jedoch auch die Einfachheit seiner Anwendung entgegenzusetzen, um in der Praxis Akzeptanz zu erfahren.

Teil IV

Technische Realisierbarkeit

Überblick zu Teil IV

In Teil IV wird die technische Realisierbarkeit der vorgestellten Ansätze diskutiert, indem wesentliche Konzepte auf die Prozesssprache BPEL abgebildet und in Werkzeugen umgesetzt werden.

Kapitel 9 umfasst eine dienstbasierte Realisierung von Sichtenentwürfen unter Berücksichtigung der organisatorischen Dimension (A8). Des Weiteren wird die Realisierbarkeit von Sichten zum sprachübergreifenden Monitoring von BPEL mit Chevron-Diagrammen gezeigt (A1, A2). Für die praktische Einsetzbarkeit des Konzepts der rr-Fragmente werden Spracherweiterungen von BPEL zur Serialisierung vorgeschlagen.

In Kapitel 10 wird gezeigt, welche neuartigen Werkzeuge, Erweiterungen und Anwendungsarchitekturen für die Umsetzung der Konzepte erforderlich sind. Im Rahmen der Monitoring-Plattform wird insbesondere auf Sichten mit Bezug zu Compliance eingegangen (A17).

KAPITEL 9

ABBILDUNG WESENTLICHER KONZEPTE AUF DIE PROZESSSPRACHE BPEL

Der musterbasierte Sichtenentwurf, die Zustandsprojektion und rr-Fragmente stellen zentrale Innovationen dar, die im Rahmen der vorliegenden Dissertation entwickelt wurden. In diesem Kapitel werden diese Themen auf die Prozesssprache BPEL übertragen. Sichten werden im Stand der Technik i.d.R. in isolierten Werkzeugen mit einer festen Integration der Transformationsfunktionen realisiert. Mit einer *dienstbasierten Realisierung* von Sichtenentwürfen wird ein neuer Weg der Sichtenrealisierung aufgezeigt, bei dem Sichtentransformationen als modulare und komponierbare Dienste bereitgestellt werden, welche flexibel zu komplexen Sichtentransformationen integriert werden können. BPEL dient bei diesem Ansatz sowohl als Quellsprache der Eingabe- und Ausgabeprozesse der Sichtentransformationen als auch als Sprache zur Orchestrierung von Transformationsdiensten. Mit einer *Zustandsprojektion von BPEL zu Chevron* wird gezeigt, wie das Monitoring von Prozessinstanzen vereinfacht und durch individuell gestaltete Chevron-Modelle verschiedenen Interessengruppen zugänglich gemacht werden kann. Die technische Realisierbarkeit des Konzeptes regelrealisierender Prozessfragmente wird anhand von *Erweiterungen von BPEL für rr-Fragmente* diskutiert. Durch die Diskussion dieser drei Themenkomplexe wird gezeigt wie die abstrakten Muster, Mustersequenzen und Prozessstrukturen zu einer konkreten Implementierung unter der Nutzung aktueller Technologien überführt werden können.

9.1 Dienstbasierte Realisierung von Sichtenentwürfen

In diesem Abschnitt wird der Ansatz des musterbasierten Sichtenentwurfs auf eine dienstorientierte Infrastruktur abgebildet (A5). Der Einsatz des Architekturstils SOA ist wohlgemerkt eine von vielen Möglichkeiten einer Sichtenimplementierung. In Kapitel 10 wird auf weitere Realisierungsmöglichkeiten eingegangen wie bspw. die Integration von Transformationsfunktionen in Werkzeuge zur Modellierung und für das Monitoring. Die Stärke der dienstbasierten Realisierung gegenüber der festen Integration liegt in der Kombinierbarkeit und Austauschbarkeit von modularen, lose gekoppelten Transformationsfunktionen. Allerdings sind die Sichten im dienstbasierten Ansatz vom Quellmodell vollständig abgekoppelt und dadurch die Möglichkeit der Übertragung von Änderungen eingeschränkt. Der Anwendungsschwerpunkt der dienstbasierten Realisierung liegt daher bei Sichten zur Prozessdarstellung und -analyse.

Dieser Abschnitt basiert auf verschiedenen Arbeiten: Die Sichtenfunktionen wurden in studentischen Arbeiten realisiert, darunter sind reduzierende Strukturveränderungen [Str09], die Einbindung von Menschen [Den12][Sun12], die Prozessdarstellung [Lat10][Rap11] und die Komposition dieser Funktionen durch einen Dienst [Cai10][Cai11]. Die für die Handhabung von rr-Fragmenten relevanten Transformationsfunktionen wurden in Schumm et al. [SLS10b] diskutiert. Die dienstbasierte Einbindung von Menschen in automatisierte Prozesse wurde von Schumm und Karastoyanova [SK11] skizziert und deren technische Realisierung von Schumm et al. [SFK+12] berichtet und im Kontext wissenschaftlicher Workflows von Karastoyanova et al. [KDS+12] diskutiert. Die Untersuchung der konfigurierbaren grafischen Darstellung wurde von Nowak et al. [NKL+12] vorgestellt, welche auf dem Konzept des Informationsdesigns für Geschäftsprozesse nach Schumm [Sch11b] basiert. Die dienstbasierte Komposition von Transformationsfunktionen haben Schumm et al. in [SCF+11] vorgestellt.

9.1.1 Bereitstellung von Transformationsfunktionen als Dienst

In der Konzeption des musterbasierten Sichtenentwurfs (vgl. Abschnitt 3.2) wurde ein Begriffsmodell eingeführt, das in Abbildung 9.1 erweitert wird und daraus die Eingaben und Ausgaben einer modularen Transformationsfunktion abgeleitet werden.

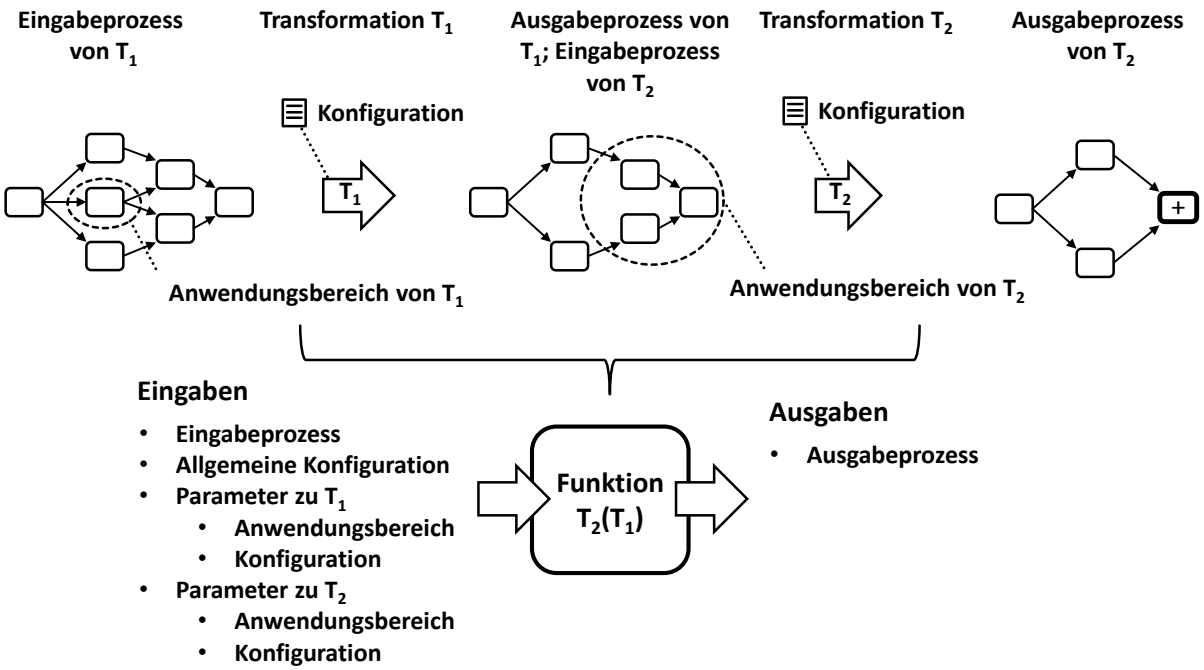


Abbildung 9.1: Eingaben und Ausgaben einer modularen Transformationsfunktion.

Ein- und Ausgaben. Ein Eingabeprozess wird verschiedenen Umformungen unterzogen, die auf einen bestimmten *Anwendungsbereich* angewendet werden. Das Begriffsmodell wird dahingehend erweitert, dass eine Transformationsfunktion anhand einer *Konfiguration* gesteuert wird. Eine allgemeine Konfiguration kann bspw. steuern, ob Vorgaben einzuhalten sind (M29-M33). Eine spezifische Konfiguration kann für eine einzelne Transformation

vorgeben, wie evtl. auftretende Mehrdeutigkeiten aufzulösen sind. Aus den sequentiell angewendeten Transformationen resultiert ein *Ausgabeprozess*.

Der initiale Schritt, eine Sichtentransformation durch einen Dienst zu realisieren, besteht in der Identifikation der Ein- und Ausgaben einer Funktion. Daraus kann eine generische Struktur von Anweisungen zur Steuerung von zusammengesetzten Transformationsfunktionen abgeleitet werden, wie in Abbildung 9.2 dargestellt.

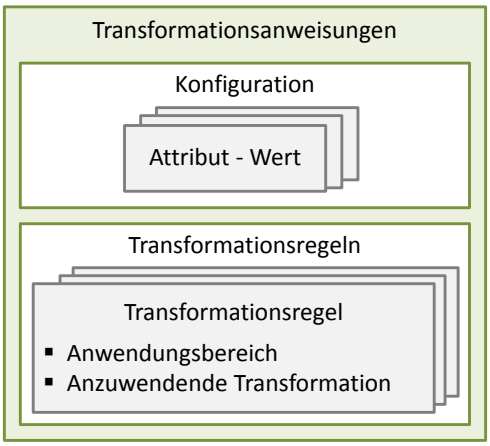


Abbildung 9.2: Generische Struktur von Transformationsanweisungen, vgl. [SCF+11].

Domänenspezifisches Eingabeformat. Wie Schumm et al. [SCF+11] anmerken, sind Transformationsfunktionen heterogen, und daher folgen die Eingabeparameter im Detail keiner einheitlichen Struktur. Eingabeparameter können bspw. einfache skalare Werte wie ein Relevanzwert zur Steuerung einer Filterfunktion sein oder auch komplexe Strukturen umfassen wie ein rr-Fragment, dessen Vorhandensein in einem Prozess analysiert werden soll. Auch der Anwendungsbereich kann je nach Transformationen auf unterschiedliche Weisen eingegeben werden, z.B. in Form eines regulären oder logischen Ausdrucks oder auch in Form einer Liste von betroffenen Prozessstrukturen. Bobrik [Bob08] verwendet bspw. eine XML-basierte Syntax zur Auflistung und Spezifikation der anzuwendenden Transformationen. Bobrik bezeichnet diese Syntax auch als View-Definitionssprache.

Daher kann kein abschließendes Format für Transformationsanweisungen festgelegt werden. Stattdessen wird das Konzept domänenspezifischer Sprachen (DSL) angewendet, um eine auf die Transformationsfunktion zugeschnittene, aber ausreichend ausdrucksstarke Beschreibungsmöglichkeit zu schaffen. Zur einfachen Nutzung einer DSL können Regeleditoren entwickelt werden, mit denen Parameter auf einfache Weise modelliert werden können. Rapoports [Rap11] hat bspw. einen Editor entwickelt, der Anweisungen für einen Dienst zur grafischen Darstellung erzeugt. Eine Standardisierung von Anweisungsformaten und der Funktionsweise gängiger Transformationen kann allerdings im Rahmen dieser Arbeit nicht geleistet werden.

Bereitstellung als Dienst. Transformationsfunktionen können als Codemodule oder Codebibliotheken in Werkzeuge eingebunden werden, um diese durch Sichtenfunktionalität zu erweitern. Um eine einfache oder auch eine zusammengesetzte Transformationsfunktion auch als Dienst nutzen zu können, muss die Funktion zu einer Komponente zusammengefasst,

durch eine Schnittstelle beschrieben und an einem definierten Endpunkt zur Verfügung gestellt werden. Damit ein Dienst in einen BPEL-Prozess eingebunden werden kann, ist die Schnittstellenbeschreibung durch WSDL [W3C01] obligatorisch. Die entsprechende Kapselung einer Transformationsfunktion als Dienst ist in Abbildung 9.3 (a) dargestellt.

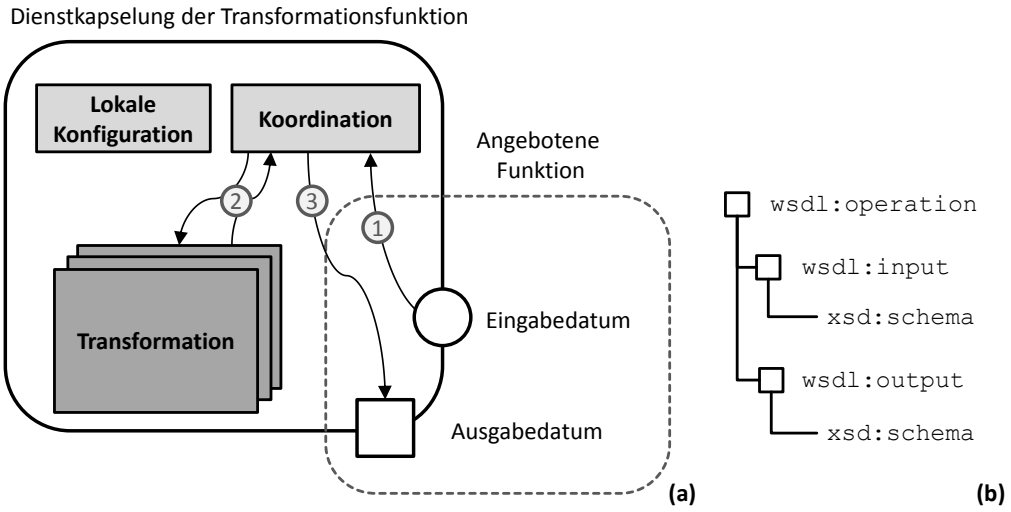


Abbildung 9.3: Konzeptionelle Kapselung einer Transformationsfunktion als Dienst (a) mit Bezug zu den Elementen einer Schnittstellenbeschreibung in WSDL (b).

Ein Dienstaufruf triggert die Funktion (1). Die gekapselte Transformationsfunktion enthält Logik zur Koordination der internen Transformationsroutinen (2). Der umgeformte Eingabeprozess wird abschließend an den Aufrufer retourniert (3). Die lokale Konfiguration umfasst Parameter, welche nicht von außen übergeben werden können. Nach außen wird lediglich offengelegt, welche Funktionen zur Verfügung stehen (`wsdl:operation`), welche Eingaben diese erwarten (`wsdl:input`), welche Ausgaben erzeugt werden (`wsdl:output`) und welche Formate für Ein- und Ausgabedaten erwartet werden (`xsd:schema`), vgl. Abbildung 9.3 (b). Formate werden in Form von XML Schema definiert. Nach außen abstrahiert der Dienst von seiner inneren Implementierung. Ein Dienst kann dementsprechend eine einfache oder zusammengesetzte Transformationsfunktion kapseln oder eine komplexe Orchestrierung mehrerer Dienste rekursiv als weiteren Dienst anbieten.

Im Zusammenhang mit musterbasierten Ansätzen ist es sinnvoll, ein Implementierungsartefakt mit dem Muster zu verknüpfen, für das es eine beispielhafte konkrete Lösung implementiert [FBB+11]. Entsprechend kann ein Transformationsdienst zur Realisierung von Sichtenfunktionalität mit den Sichtenentwurfsmustern verknüpft werden, die er realisiert.

9.1.2 Beispielhafte Transformationsdienste

Dienst 1: Reduzierende Strukturveränderungen. Der Dienst bietet mehrere Operationen unterschiedlicher Granularität zur Strukturreduzierung eines Eingabeprozesses an.

Die feingranularen Operationen unterstützen Transformationen zur Modifikation von Eigenschaften von Prozesselementen (M4), das Entfernen von Aktivitäten ohne die Einhaltung der Atomicity-Eigenschaft (M1), das Bewahren des Zusammenhalts (M29), das Bewahren ge-

kennzeichneter Strukturen (M31) und die Protokollierung von Transformationen über die Kommandozeile (M32). Das Aufräumen des Prozesses (M8) wurde in der ersten Implementierung nach Streule [Str09] durch eine Konfiguration gesteuert und in einer Kapselung als Dienst nach Cai [Cai10] auch als separate Operation angeboten.

Der Dienst stellt eine DSL zur Beschreibung von Transformationsanweisungen bereit. Der Anwendungsbereich einer Anweisung `<rule>` wird durch einen Ausdruck `<targets>` festgelegt, der eine Verschachtelung der logischen Operatoren `<and>`, `<or>` und `<not>` erlaubt. Zur Auswahl, ob sich ein Prozesselement im Anwendungsbereich befindet, sind die Vergleichsoperatoren `tag`, `attribute` und `type` verfügbar. Der Anwendungsbereich einer Transformation kann somit auf Basis einer Annotation, einem beliebigen Attribut wie bspw. eines Namens oder einer ID oder auf Basis des Elementtyps wie z.B. `invoke`, `variable` oder `import` erfolgen. Welche Transformationen auf die Prozesselemente im Anwendungsbereich angewendet werden sollen, wird im Ausdruck `<actions>` festgelegt. Als Transformationen stehen `actionOmit` zum Entfernen von Aktivitäten (M1), `actionOpaque` zur Abstraktion von Aktivitäten (M4) und `actionSetAttributeTo` zur Abstraktion beliebiger Eigenschaften (M4) zur Verfügung. Der Dienst bietet außerdem die Möglichkeit, einen ganzen Satz an Transformationsregeln als Liste zu übergeben, die sequentiell auf einen Eingabeprozess angewendet werden.

Listing 9.1 zeigt eine einelementige Liste mit einer Regel zum Ausschneiden eines rr-Fragments nach der Darstellung von Schumm et al. [SLS10b]. Die Regel `extractFragment` besagt, dass die Transformation `actionOmit` angewendet werden soll. Die spezifische Konfiguration `preserveChildren` gibt an, dass vom Nutzer gekennzeichnete Aktivitäten bewahrt werden sollen (M31), auch wenn deren übergeordnete Struktur nicht gekennzeichnet ist. Der Ausdruck in Listing 9.1 gibt an, dass alle Aktivitäten im Anwendungsbereich `<targets>` liegen, die nicht die Kennzeichnung `preserve` tragen und somit von Nutzer als zum rr-Fragment zugehörig gekennzeichnet wurden.

```
<rules>
    <rule name="extractFragment" apply="true">
        <actions>
            <actionOmit preserveChildren="true"/>
        </actions>
        <targets>
            <not>
              <tag tagName="preserve" />
            </not>
        </targets>
    </rule>
</rules>
```

Listing 9.1: Transformationsanweisung zum Ausschneiden eines rr-Fragments [SLS10b].

Der Dienst bietet außerdem eine grobgranulare Operation zum Entfernen eines rr-Fragments an. Diese Operation erwartet als Parameter einen Eingabeprozess und das zu entfernende rr-Fragment. Ein interner Regelgenerator erzeugt eine Regelliste, in der es für jedes Element im rr-Fragment eine Regel für dessen Entfernen enthalten ist. Eine Prüfung, ob das Fragment in dieser Form in einem Prozess tatsächlich enthalten ist, findet demnach bei dieser Operation

nicht statt. Diese Funktion könnte bei Bedarf durch einen anderen Dienst erbracht werden. Die Regelvorlage zur Generierung von Entfernungsregeln ist in Listing 9.2 gezeigt.

```
<rule name="omit%CONSTRUCT-NAME%" apply="true">
    <actions>
        <actionOmit preserveChildren="true"
                    preserveTransitionConditions="false"/>
    </actions>
    <targets>
        <attribute attributeName="name" value="%CONSTRUCT-NAME%" />
    </targets>
</rule>
```

Listing 9.2: Regelvorlage als Grundlage zum Entfernen eines rr-Fragments [SLS10b].

Weitere Details zu dieser DSL können in dem Artikel [SLS10b] nachgelesen werden.

Dienst 2: Einbindung von Menschen. Die Erzeugung von Sichten auf Geschäftsprozesse erfordert die Einbindungen von Menschen an verschiedenen Stellen. Menschen können bspw. in Form nutzergestützter Anreicherung (M13) Informationen zur Semantik und individueller Relevanz zur Verfügung stellen. Derlei Informationen stellen einen generellen, langfristig zu erhaltenden Wert dar und daher findet diese Form der Informationsbereitstellung i.d.R. zeitlich vor der Durchführung von anderen Transformationen in einem Modellierungswerkzeug statt. Soll hingegen eine Einschätzung eines Nutzers ad hoc erfasst werden oder eine Entscheidung zur Auflösung einer mehrdeutigen Situation getroffen werden, findet diese Abfrage inmitten einer Transformationssequenz statt.

In einer dienstbasierten Realisierung muss eine derartige Abfrage durch einen Dienstaufruf getätigt werden. Für BPEL stehen dazu neben einem Dienstaufruf mit `<invoke>` bereits die Erweiterungen BPEL4People [OAS09a] und WS-HumanTask [OAS09b] zur Verfügung. Diese erlauben den Aufruf einer Komponente zur Aufgabenverwaltung (engl. *human task manager*). Diese Komponente erstellt nach einem Aufruf eine Aufgabe, die anschließend von einem Nutzer angenommen und bearbeitet werden kann. Das Ergebnis kann nach Abschluss der Aufgabe an den Aufrufer retourniert werden. Neben dieser engen Kopplung von Aufgaben an eine Aufgabenliste (engl. *task list*) wurde von Schumm und Karastoyanova [SK11] eine lose Kopplung vorgeschlagen, nach der ein Nutzer über das Kommunikationsmedium eingebunden wird, über das er am besten verfügbar ist. Der technische Bericht von Schumm et al. [SFK+12] spezifiziert die dazu nötigen Prozesse und Nachrichtenformate, die eine entsprechende Kommunikationskomponente unterstützen muss, um zwischen einem Aufrufer und einem Nutzer zu vermitteln (Human Communication Manager, HCM). Zur Einbindung des HCM in einen Workflow als auch zum einfachen Aufruf eines E-Mail-Kommunikationsdienstes zur Einbindung eines Nutzers (Request Response) wurden entsprechende BPEL-Fragmente und Kommunikationsdienste implementiert [Den12][Sun12]. Weitere Details zur Architektur und Implementierung können in einer Zusammenfassung und Diskussion der Erkenntnisse im Kontext von Simulationstechnologien nachgelesen werden [KDS+12].

Dienst 3: Konfigurierbare Darstellung. Die Konfiguration der grafischen Darstellung eines Prozesses erfordert nach dem Stand der Technik die Nutzung einer komplexen Skriptsprache, mit der die Verknüpfung von grafischen Vorlagen, Prozesselementen und analytischen Daten vorgenommen wird, vgl. dazu [Bob08]. Auf Basis der von Latuske [Lat10] entwickelten Funktionalität zur Darstellung einer BPEL-Prozessinstanz durch skalierbare Vektorgrafiken (SVG) wurde von Rapoports [Rap11] ein Dienst entwickelt, der auf Eingabe eines Prozesses in BPEL und einer sogenannten grafischen Konfiguration eine maßgeschneiderte Repräsentation in SVG retourniert (M16, M17, M19, M24). Die grafische Konfiguration kann in einem eigens dafür entwickelten Editor modelliert werden. Die erzeugte SVG-Repräsentation kann in einem SVG-fähigen Webbrowser dargestellt werden. Dieser Editor stellt damit die DSL dieses Dienstes auf einfach zugängliche Weise bereit. Die grafische Konfiguration unterstützt eine lose Kopplung von Geschäftsprozessen, grafischen Elementen und in die Darstellung einzubeziehender Daten (M15), die nach Schumm [Sch11b] einen wichtigen Aspekt im Informationsdesign darstellen. Die Verfeinerung und technische Diskussion dieses Ansatzes wurde von Nowak et al. [NKL+12] geleistet. Auf Basis der durchgeführten Arbeiten wurde ein konzeptionelles Modell einer DSL zur grafischen Konfigurationen einer Sicht abgeleitet, vgl. das in Abbildung 9.4 dargestellte UML Klassendiagramm.

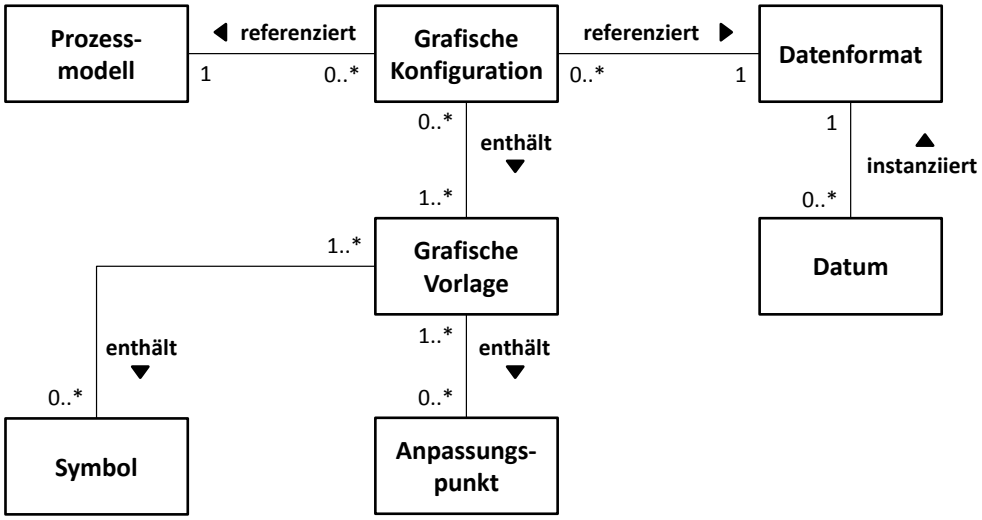


Abbildung 9.4: Konzeptionelles Modell einer DSL für die grafische Konfiguration einer Prozesssicht, angepasst nach [NKL+12].

Der Editor implementiert dieses Modell, indem er dem Nutzer erlaubt, individuelle grafische Vorlagen zur Darstellung von Aktivitäten zu importieren und diese in einer Konfiguration zu verwenden. Eine grafische Vorlage enthält verschiedene Symbole wie bspw. zur Darstellung des Ausführungszustands oder des Aktivitätstyps. Die Anpassungspunkte einer Vorlage machen dem Editor die Information explizit, wie die Vorlage angepasst werden kann, bspw. können verschiedene Beschriftungsfelder als Anpassungspunkt zur Verfügung stehen. Die Konfiguration gibt außerdem an, welche Vorlage für welche Aktivitäten eines Eingabeprozesses verwendet werden soll. Ein Betriebsprüfer könnte bspw. eine Standardvorlage nutzen, in der alle für seine Fragestellungen relevanten Informationen einheitlich dargestellt werden. Die Konfiguration stellt zudem die Verbindung zu analytischen Daten anhand des Datenformats her, in welchem die Daten bereitgestellt werden können. Konkrete Daten, die

dieses Datenformat instanziieren, werden erst zur Aufrufzeit des Dienstes bereitgestellt. Weitere technische Details und grafische Beispiele können in dem Artikel [NKL+12] und der studentischen Arbeit [Rap11] nachgelesen werden.

9.1.3 Von der Mustersequenz zur Dienstkomposition

Eine geradlinige Vorgehensweise zur Umsetzung einer Mustersequenz würde darin bestehen, die Orchestrierung dafür erforderlicher Dienste als BPEL-Prozess zu modellieren. Durch eine freie Modellierung in BPEL könnten auch komplexe Transformationsszenarien realisiert werden, die etwa Verzweigungen und Schleifen beinhalten. So könnten etwa Eingabeprozesse mit mehr als 50 Aktivitäten anders behandelt werden als Eingabeprozesse mit einer geringeren Anzahl an Aktivitäten. Die in der vorliegenden Dissertation vorgestellten Sichtenentwürfe machen von derartigen Konstrukten zwar keinen Gebrauch, sinnvolle Einsatzszenarien sind aber durchaus denkbar. Einer freien Modellierung ist entgegenzuhalten, dass dieses Programmieren im Großen [DK75] wiederum viel technisches Know-how erfordert und damit die Komplexität einer Realisierung steigt.

Eine mögliche Herangehensweise besteht daher darin, eine schlanke und einfach zu verwendende Möglichkeit der Dienstkomposition anzubieten, die auf Transformationssequenzen fokussiert. Dieser Ansatz stellt eine Anwendung des Architekturstils *Pipes and Filters* dar. Hohpe und Woolf [HW03] beschreiben diesen Architekturstil als Muster: Das Problem besteht darin, wie eine komplexe Verarbeitung einer Nachricht flexibel und ohne Abhängigkeiten durchgeführt werden kann. Die vorgeschlagene Lösung dieses Problems ist, umfassende Verarbeitungsschritte in eine Sequenz kleinerer, voneinander unabhängiger Schritte (Filter) zu zerlegen und diese Schritte über Kanäle zu verbinden (Pipes). Die Übertragung dieser Herangehensweise auf den Kontext der Erzeugung von Sichten auf Prozesse wurde von Schumm et al. [SCF+11] diskutiert und wird im Folgenden vorgestellt.

Die Vorgehensweise besteht darin, zunächst eine Reihe von Transformationsdiensten auszuwählen, deren angebotene Operationen die in der zu realisierenden Mustersequenz angewendeten Muster unterstützen. Entsprechend der Abhängigkeiten der Musteranwendungen werden die Dienstaufrufe in eine Sequenz gebracht und der Datenfluss zwischen den Dienstaufrufen festgelegt. Das so entstehende Resultat wird in der vorliegenden Dissertation als (dienstbasierte) Sichtendefinition bezeichnet. Bildlich fließt der Eingabeprozess durch die einzelnen Dienste hindurch, daher liegt der Vergleich mit einer Pipeline oder auch mit einem Fließband mit verschiedenen Bearbeitungsstationen nahe. Aus einer Sichtendefinition kann anschließend ausführbarer BPEL-Code generiert werden, der die Sicht als Dienstkomposition realisiert und diese als Dienst bereitstellt. Nachfolgend wird auf die Modellierung von Sichtendefinitionen und die Generierung einer Dienstkomposition aus einer Sichtendefinition eingegangen.

Modellierung von Sichtendefinitionen. Eine Sichtendefinition setzt die im feinen Sichtenentwurf formulierten Anforderungen in Form einer Sequenz von Dienstaufrufen um. Der erste Schritt bei der Erstellung einer Sichtendefinition ist die Abfrage eines Verzeichnisses verfügbarer Transformationsdienste. Die Abfrage dient dem Abgleich, welche Dienste aktuell

zur Verfügung stehen und welche Muster oder Mustersequenzen diese Dienste implementieren. Werden bestimmte Muster in der Mustersequenz des zu realisierenden Sichtenentwurfs nicht durch einen Dienst unterstützt, muss ein weiterer Dienst entwickelt oder zugekauft werden. Sind diese Möglichkeiten nicht gegeben, ist im Einzelfall zu klären, ob eine Anpassung bzw. eine unvollständige Realisierung der Mustersequenz sinnvoll ist. Derartige Realisierungsschwierigkeiten können im Vorfeld durch eine Teilmengenbildung des Musterkatalogs abgefangen werden.

Entsprechend den Anforderungen aus dem Feinentwurf werden von den Diensten angebotene Operationen ausgewählt und in einer Sequenz geordnet. Der nächste Schritt besteht in der Festlegung des Datenflusses, der die einzelnen Dienstaufrufe verbindet. Die Ausgaben der einzelnen Dienstaufrufe können nur in nachfolgenden Dienstaufrufen als Eingaben verwendet werden. Man kann außerdem zwischen dynamischen und statischen Parametern zur Konfiguration der Sicht unterscheiden. Dynamische Parameter erlauben die Konfiguration zur Aufrufzeit der entstehenden Dienstkomposition, während mit statischen Parametern eine Konfiguration vorgenommen werden kann, die über die Aufrufe hinweg konstant ist.

Beispiel. Die in Abbildung 9.5 skizzierte Sichtendefinition realisiert eine Sicht, um (1) durch den dynamischen Parameter A als unerwünscht charakterisierte Prozessfragmente zu erkennen, (2) gekennzeichnete Prozessfragmente aus dem Eingabeprozess zu entfernen und (3) den daraus resultierenden Prozess in SVG umzuwandeln, das in einem gängigen Webbrowser dargestellt werden kann. Parameter A kann für einen konkreten Aufruf bspw. Schwellenwerte enthalten, mit denen wenig relevante Aktivitäten von relevanten unterschieden werden. Der statische Parameter B enthält Transformationsanweisungen zum Entfernen (M1) und Aufräumen (M8), die in einer dienstspezifischen DSL formuliert sind. Der statische Parameter C kann eine grafische Standard-Konfiguration enthalten, die im Vorfeld mit einem Editor erstellt wurde. Die einzelnen Dienstaufrufe haben lediglich eine Ausgabe: den transformierten Eingabeprozess. Diese Ausgabe wird dem Nachfolger als Eingabe weitergereicht.

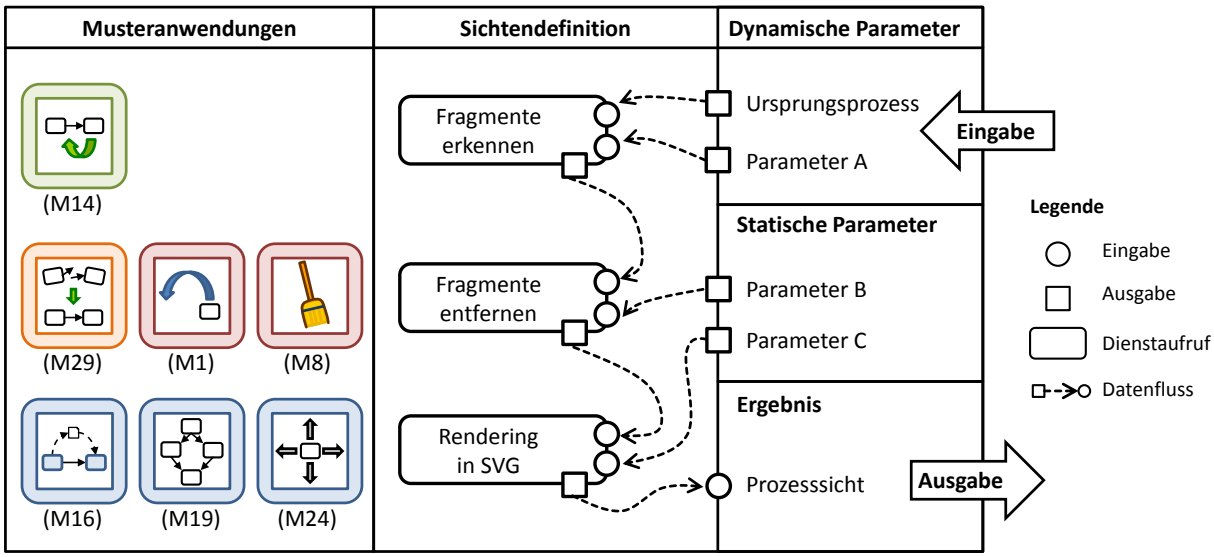


Abbildung 9.5: Festlegung des Datenflusses bei einer Komposition von Transformationsschritten durch Dienstaufrufe, angepasst und erweitert nach [SCF+11].

Generierung einer Dienstkomposition in BPEL. Eine Sichtendefinition ist gültig, wenn der Datenfluss keine Vorwärtsreferenzen enthält und jeder Dienstaufrufeingabeparameter an einen dynamischen Parameter, einen statischen Parameter oder an eine Ausgabe von vorangehenden Dienstaufrufen gebunden wurde und diese Eingabe dem erwarteten Format entspricht. Eine gültige Sichtendefinition kann zur Generierung eines ausführbaren BPEL-Prozesses verwendet werden, der die Transformationsdienste orchestriert und gesamthaft als Dienst zur Verfügung stellt.

Bei der Generierung eines ausführbaren BPEL-Prozesses werden verschiedene Artefakte erzeugt und zusammengetragen. Dazu gehören neben der eigentlichen BPEL-Datei ein WSDL-Dokument, welches die Schnittstellen des BPEL-Prozesses als Web Service beschreibt, die WSDL-Dokumente der im Prozess aufgerufenen Dienste sowie ein Deployment Descriptor, der Laufzeiteigenschaften für einen Prozessserver festgelegt.

Zur Veranschaulichung zeigt Abbildung 9.6 die Struktur eines automatisch aus einer Sichtendefinition generierten BPEL-Prozesses zur Komposition von Transformationsdiensten. In dieser Darstellung wird eine BPMN-Sprachemulation verwendet, vgl. dazu Abschnitt 5.3.

Die Dienstkomposition wird durch eine `<receive>` Aktivität gestartet. Dieser Aktivität werden alle dynamischen Parameter übergeben. Die statischen Parameter werden bei der Instanziierung des Prozesses gesetzt. Der Aufruf eines Transformationsdienstes wird durch eine `<assign>` Aktivität zur Zuweisung der erforderlichen Eingabeparameter vorbereitet. Der nachfolgende synchrone Dienstaufruf wird durch eine `<invoke>` Aktivität implementiert.

Die Struktur in Abbildung 9.6 zeigt zwei Dienstaufrufe und deutet weitere Dienstaufrufe durch einen Platzhalter an. Nach den Dienstaufrufen wird die Rückgabe des transformierten Prozesses durch eine `<assign>` Aktivität vorbereitet. Eine `<reply>` Aktivität liefert die erzeugte Sicht an den Aufrufer der Dienstkomposition zurück. Nach der Generierung des BPEL-Prozesses kann seine Schnittstellenbeschreibung in Form eines generierten WSDL-Dokuments mit der von ihm implementierten Mustersequenz in das Verzeichnis verfügbarer Transformationsdienste aufgenommen werden. Die Architektur und Implementierung eines Werkzeugs, das eine dienstbasierte Realisierung von Sichtenentwürfen unterstützt, wird in Abschnitt 10.1 vorgestellt.

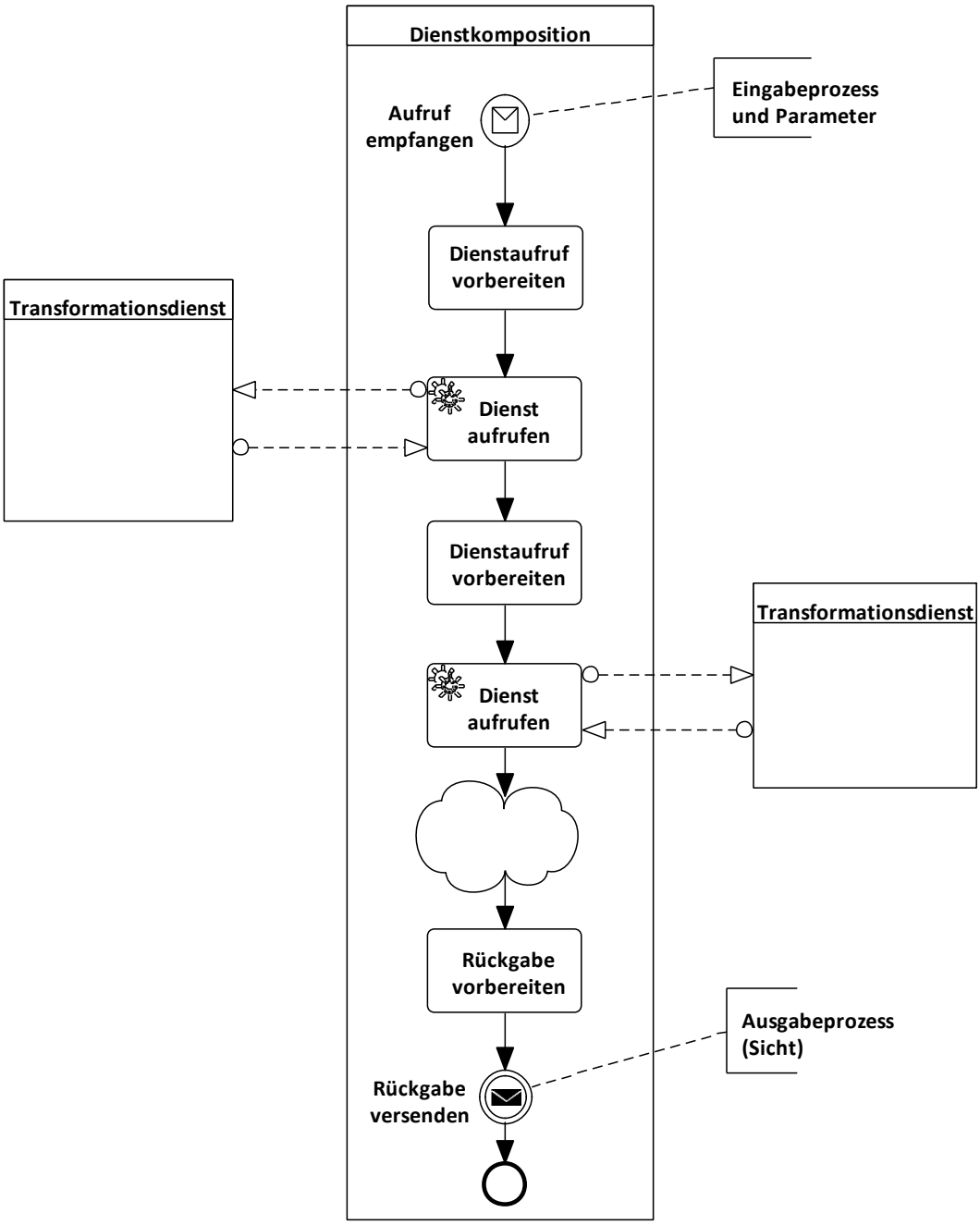


Abbildung 9.6: Struktur eines aus einer Sichtendefinition generierten BPEL-Prozesses zur Komposition von Transformationsdiensten nach [SCF+11] in einer Sprachemulation von BPEL durch BPMN.

9.2 Zustandsprojektion von BPEL zu Chevron

Das Ziel einer Zustandsprojektion kann sich sowohl in der Semantik als auch in der Prozesssprache vollkommen vom Quellmodell unterscheiden. Die Zustandsprojektion erlaubt damit im Kontrast zur Sprachemulation (vgl. Abschnitt 5.3.2) den Einsatz eines grundlegend unterschiedlichen Prozessmodells für das Monitoring wie zur tatsächlichen Ausführung. In diesem Abschnitt werden die technischen Grundlagen vorgestellt, mit denen eine Zustandsprojektion von Prozessinstanzen in BPEL zu Chevron-Diagrammen realisiert werden kann. Die Ausführungen basieren auf dem technischen Bericht von Schumm et al. [SKL+11] und

der vorangehenden Implementierung einer Zustandsprojektion im Rahmen einer studentischen Arbeit [Lie11]. Zunächst wird die Spezifikation von Zustandsübertragungen auf konzeptioneller Ebene diskutiert (Abschnitt 9.2.1) und anschließend auf die Vorgehensweise zur Durchführung einer Zustandsprojektion eingegangen (Abschnitt 9.2.2). Danach wird der konkrete Bezug zu BPEL hergestellt (Abschnitt 9.2.3). Der Abschnitt schließt mit einem Vorschlag zur Serialisierung und Darstellung von Modellen in Chevron (Abschnitt 9.2.4).

9.2.1 Konzeptionelles Modell zur Spezifikation einer Zustandsprojektion

Die identifizierten Muster zur Zustandsübertragung (vgl. Abschnitt 5.4.2) stellen zwar wiederverwendbare Lösungen dar, diese müssen allerdings für eine praktische Anwendung konkretisiert und an die Beschaffenheit von konkreten Prozesssprachen angepasst werden. Nachfolgend wird ein konzeptionelles Modell vorgestellt (siehe Abbildung 9.7), mit dem Zustandsübertragungen von BPEL zu Chevron spezifiziert werden können. Das konzeptionelle Modell bietet eine Unterstützung der Zustandsübertragungsmuster direkte Übertragung (Z1), Zustandsänderung (Z2) und Zustandskombination (Z3).

Das konzeptionelle Modell liest sich wie folgt: Eine *Projektion* enthält einen Verweis auf das *Quellmodell* und das *Zielmodell*. Verschiedene Formen der Zustandsübertragung können als *Übertragungsvorlage* zur Wiederverwendung hinterlegt werden. Eine *Übertragungsregel* wendet eine *Übertragungsvorlage* auf eine Menge von *Quellaktivitäten* und eine *Zielaktivität* an. Dadurch können verschiedene Kombinationen der möglichen Zustände der *Quellaktivitäten* auf genau einen Zustand der *Zielaktivität* abgebildet werden. Eine *Quellaktivität* ist Bestandteil des *Quellmodells*, eine *Zielaktivität* Bestandteil des *Zielmodells* der Projektion.

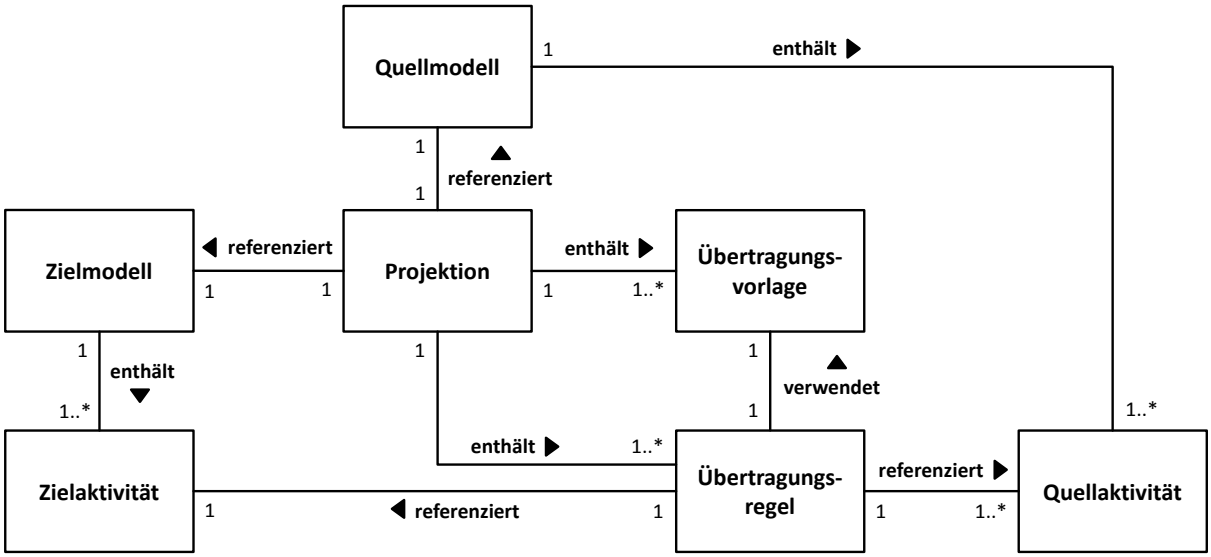


Abbildung 9.7: Konzeptionelles Modell zur Spezifikation einer Zustandsprojektion.

Eine Übertragungsvorlage wird durch Fallabfragen über die möglichen Kombinationen der Zustände der Quellaktivitäten spezifiziert. Dazu sind neben einer `sonst` Anweisung (engl. *else*) die einfachen Vergleichsoperatoren `enthält Zustand` (engl. *contains state*) und `alle Zustände gleichen` (engl. *all states equal*) vorgesehen. Die Reihenfolge der Auswertung

der Fallabfragen ist für das Ergebnis entscheidend. Eine beispielhafte Übertragungsvorlage einer Zustandskombination (Z3) ist in Listing 9.3 in Pseudocode formuliert.

```
Wenn alle Zustände gleichen Erledigt Dann Abgeschlossen
Wenn alle Zustände gleichen Inaktiv Dann Unbearbeitet
Wenn enthält Zustand Fehlerhaft Dann Unbearbeitet
Sonst In Bearbeitung
```

Listing 9.3: Fallabfragen einer beispielhaften Übertragungsregel, mit der Zustände kombiniert werden (Z3).

Abbildung 9.8 stellt eine beispielhafte Übertragungsvorlage zur Zustandsänderung (Z2) vom PI-Graph-Zustandsmodell auf das Zustandsmodell von Chevron grafisch dar.

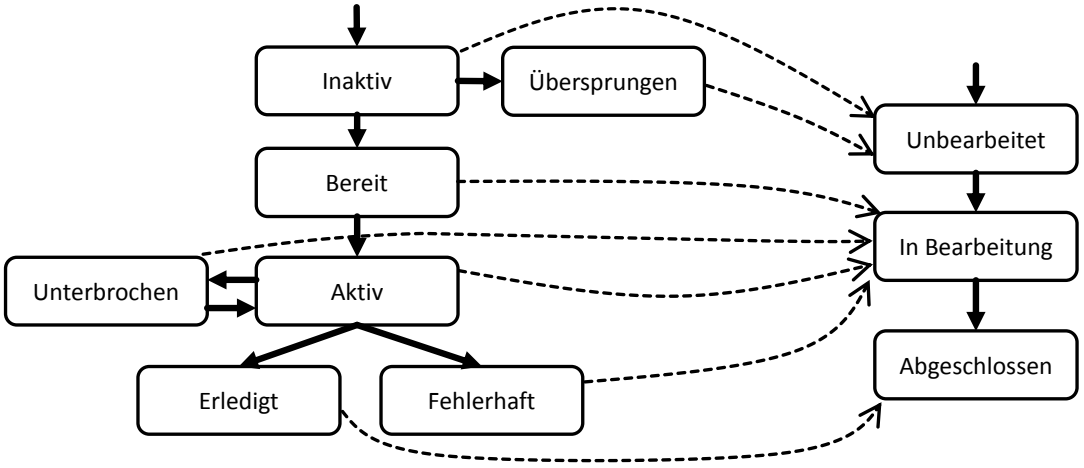


Abbildung 9.8: Grafische Darstellung einer beispielhaften Übertragungsvorlage, um den Aktivitätszustand von einem PI-Graphen (links) zu Chevron (rechts) zu ändern (Anwendung von Übertragungsmuster Z2).

9.2.2 Vorgehensweise zur Durchführung einer Zustandsprojektion

Schumm et al. [SKL+11] haben die Vorgehensweise zur Durchführung einer Zustandsprojektion von BPEL zu Chevron in fünf Schritte unterteilt. Diese Schritte sind in Abbildung 9.9 prozessual dargestellt und werden nachfolgend erläutert. Bei der prozessualen Darstellung wird zwischen dateibasierter Ablage und der Verarbeitung im Speicher unterschieden, dies entspricht der prototypischen Umsetzung nach [Lie11].

Schritt 1: Einlesen des BPEL-Quellmodells und der Aktivitätszustände. Im initialen Schritt werden die im Quellmodell enthaltenen Aktivitäten extrahiert und mit Zustandsinformationen angereichert. Daraus resultiert eine Menge zustandsbehafteter Quellaktivitäten.

Schritt 2: Einlesen der anzuwendenden Projektion. Zunächst werden die in der Projektion enthaltenen Übertragungsvorlagen eingelesen. Anschließend werden die Übertragungsregeln eingelesen und mit den Übertragungsvorlagen verknüpft.

Schritt 3: Auswertung der Übertragungsregeln. Die eingelesenen Übertragungsregeln werden auf Basis der zustandsbehafteten Quellaktivitäten ausgewertet. Bei der Auswertung einer Regel wird ein Aktivitätszustand bestimmt, der den Zustand der in der Übertragungsregel

referenzierten Zielaktivität repräsentiert. Das eigentliche Zielmodell ist in diesem Schritt noch unbekannt.

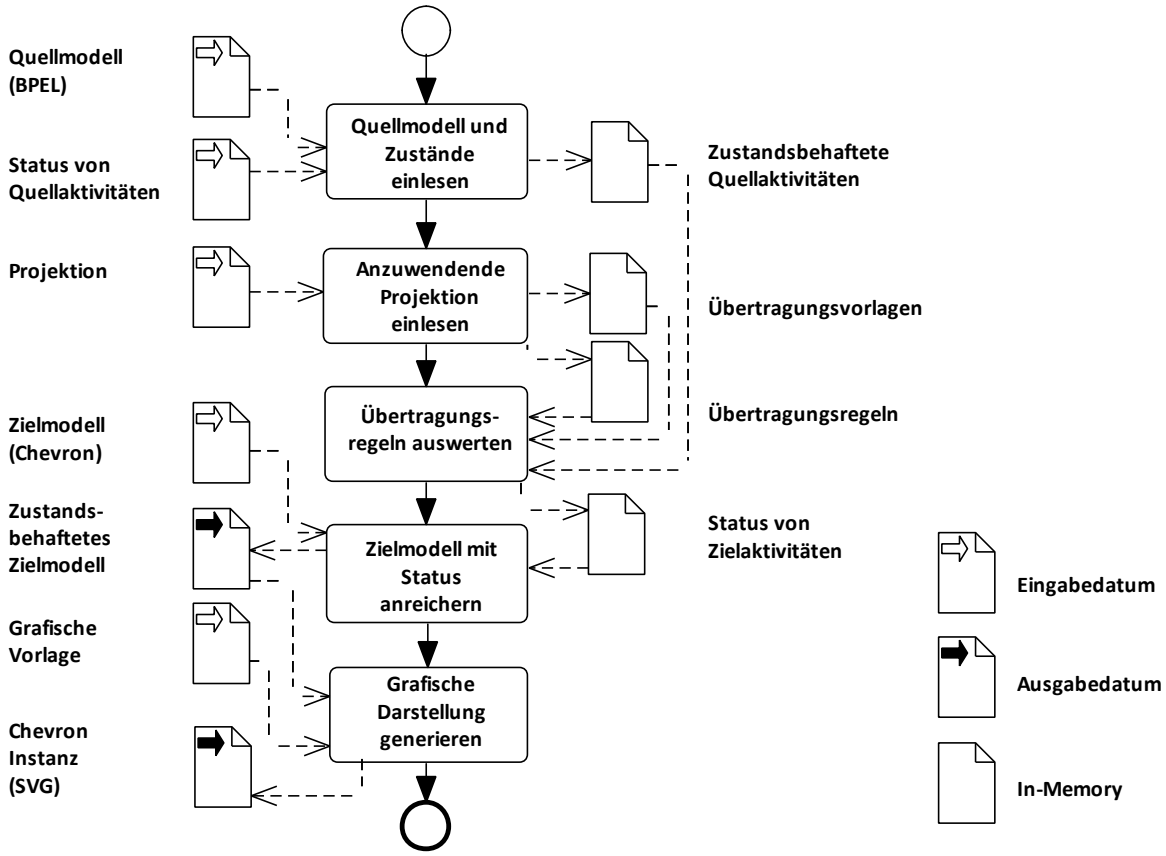


Abbildung 9.9: Vorgehensweise zur Durchführung einer Zustandsprojektion, angepasst nach [SKL+11].

Schritt 4: Anreicherung des Zielmodells mit Aktivitätszuständen. Die ermittelten Aktivitätszustände werden auf die Aktivitäten des Chevron-Zielmodells übertragen. Anschließend werden die Status der Phasen des Chevron-Modells bestimmt, sofern diese nicht explizit durch eine Übertragungsregel definiert wurden. Dieser Schritt erzeugt ein zustandsbehaftetes Chevron-Modell.

Schritt 5: Generierung der grafischen Darstellung. Die Phasen und Aktivitäten des zustandsbehafteten Chevron-Modells werden unter Verwendung vordefinierter grafischer Vorlagen nach Chevron ausgelegt. Dafür wird SVG [W3C05] eingesetzt. Mit SVG können Vektorgrafiken in XML beschrieben und in gängigen Webbrowsern gerendert werden. Dieser Schritt schließt die Vorgehensweise zur Durchführung einer Zustandsprojektion ab.

9.2.3 Spezifikation von Zustandsübertragungsregeln

Das vorgestellte Konzept der Zustandsübertragung ist weitgehend generisch. Für die Nutzung von Zustandsübertragungen mit der Quellsprache BPEL müssen lediglich die Menge von Aktivitätszuständen von BPEL-Prozessinstanzen und die Referenzierungsmöglichkeiten von Quell- und Zielaktivitäten betrachtet werden. Unter der Annahme, dass die Menge von Aktivitätszuständen in BPEL der eines PI-Graphen entspricht (vgl. Definition 2, Abschnitt

2.1.4), sind die Aktivitätszustände *Inaktiv*, *Bereit*, *Aktiv*, *Unterbrochen*, *Erledigt*, *Fehlerhaft* und *Übersprungen* möglich. Als mögliche Aktivitätszustände in Chevron wurden *Unbearbeitet*, *In Bearbeitung* und *Abgeschlossen* festgelegt, vgl. Abschnitt 5.3.1.

Listing 9.4 zeigt eine Übertragungsvorlage in XML. Diese Vorlage bildet eine Menge von Aktivitätszuständen der Quellsprache BPEL auf einen Aktivitätszustand der Zielsprache Chevron ab („Status“). Die Vorlage beinhaltet zwei Fallabfragen und einen Sonst-Zweig. Die erste Fallabfrage besagt, dass, wenn sich mindestens eine der Quellaktivitäten in einem der Zustände *Bereit*, *Aktiv*, *Unterbrochen* oder *Fehlerhaft* befindet, der Status der Zielaktivität als *In Bearbeitung* gesetzt werden soll. Die zweite Fallabfrage besagt, dass wenn alle Quellaktivitäten sich im Zustand *Erledigt* befinden, der Status der Zielaktivität als *Abgeschlossen* gesetzt werden soll. Andernfalls soll der Status der Zielaktivität als *Unbearbeitet* gesetzt werden. Eine Übertragungsregel kann diese Vorlage auf eine konkrete Menge von BPEL-Quellaktivitäten und eine konkrete Chevron-Zielaktivität anwenden.

```
<statePropagationSet name="Kombination">
   <condition>
      <containState>Bereit;Aktiv;Unterbrochen;Fehlerhaft</containState>
      <targetState>In Bearbeitung</targetState>
   </condition>
   <condition>
      <allStatesEqual>Erledigt</allStatesEqual>
      <targetState>Abgeschlossen</targetState>
   </condition>
   <else>Unbearbeitet</else>
</statePropagationSet>
```

Listing 9.4: Beschreibung einer Übertragungsvorlage in XML, angepasst nach [Lie11].

Listing 9.5 zeigt eine Übertragungsregel, die Quellaktivitäten `<fromActivities>`, eine Zielaktivität `<toActivity>` und eine Referenz auf die zu verwendende Übertragungsvorlage `<stateSet>` umfasst. Die Aktivitäten *InitiateDelivery* und *AwaitCompletion* sind über das identifizierende Attribut `name` referenziert. Die Zielaktivität in Chevron ist mit dem identifizierenden Attribut `activityID` und dem Wert `2.2` referenziert. Durch die Möglichkeit, das identifizierende Attribut individuell festzulegen, können auch andere Identifikationsmerkmale genutzt werden, wie bspw. eine eindeutige Kennzeichnung einer Aktivität eines rr-Fragments oder ein werkzeugspezifisches Identifikationsmerkmal.

```
<statePropagationRule name="Auslieferung">
      <fromActivities>
            <activity identifier="name"
                      value="InitiateDelivery"/>
            <activity identifier="name"
                      value="AwaitCompletion"/>
      </fromActivities>
      <toActivity>
            <activity identifier="activityID" value="2.2"/>
      </toActivity>
      <stateSet pattern="Kombination"/>
</statePropagationRule>
```

Listing 9.5: Beschreibung einer Übertragungsregel in XML, angepasst nach [Lie11].

9.2.4 Realisierung von Chevron mit XML und SVG

Listing 9.6 zeigt einen Ausschnitt eines zustandsbehafteten Chevron-Modells in XML. Der Ausschnitt umfasst die Phase *Versand*, abgebildet als `<complexActivity>` sowie zwei darin enthaltene Aktivitäten, abgebildet als `<activity>`. Der Phasenstatus ist dem untergeordneten Element `<complexActivityStatus>` zu entnehmen. Analog enthält auch jede Aktivität ein untergeordnetes Element für den Aktivitätsstatus, `<activityStatus>`. Der jeweilige Status wurde durch Auswertung der betreffenden Übertragungsregel bestimmt. Der Status der Aktivität mit der `activityID 2.2` wurde in diesem Beispiel zu *Unbearbeitet* ausgewertet. Die gewünschte Positionierung der Aktivitäten in der Darstellung kann der Reihenfolge in XML oder einer aufsteigenden Sortierung über das Attribut `activityID` entnommen werden.

```
<complexActivity activityCaption="Versand" activityColor="blue"
                 activityDocumentation="Versand der Waren"
                 activityID="2" activityIconForCaption="delivery.png"
                 activityIconForComplexActivity="plus.png">
   <complexActivityStatus>In Bearbeitung</complexActivityStatus>
   <activity activityCaption="Vorbereiten" activityColor="blue"
             activityDocumentation="Vorbereiten des Versands"
             activityID="2.1" activityIconForCaption="">
      <activityStatus>In Bearbeitung</activityStatus>
   </activity>
   <activity activityCaption="Ausliefern" activityColor="blue"
             activityDocumentation="Auslieferung der bestellten Waren"
             activityID="2.2" activityIconForCaption="">
      <activityStatus>Unbearbeitet</activityStatus>
   </activity>
</complexActivity>
```

Listing 9.6: Ausschnitt eines zustandsbehafteten Chevron-Modells in XML, angepasst nach [Lie11].

Unter Zuhilfenahme von grafischen Vorlagen kann das zustandsbehaftete Chevron-Modell in SVG überführt werden. Eine bereits befüllte SVG-Vorlage für eine Chevron-Aktivität ist in Listing 9.7 aufgeführt. Es beschreibt die Darstellung der Sparren-Form der Aktivität durch das Element `<polygon>` anhand von Koordinaten, angegeben als `points`. Ferner sind die Aktivitätsbezeichnung durch das Element `<text>` und ein Symbol `<image>` zur Darstellung des Status der Aktivität enthalten. Abbildung 9.10 zeigt das Rendering der in SVG beschriebenen zustandsbehafteten Aktivität in einem Standardbrowser.

```
<polygon points="70,355 220,355 250,385 220,415 70,415 100,385"
         style="fill:#d6d6d6;stroke:#000000;stroke-width:2"/>
<text font-family="Franklin Gothic Heavy" font-size="14"
      style="fill:black" x="110" y="390">Vorbereiten</text>
<image x="190" y="335" width="40" height="40" xlink:href="running.png"/>
```

Listing 9.7: Ausschnitt eines zustandsbehafteten Chevron-Modells in SVG, angepasst nach [Lie11].

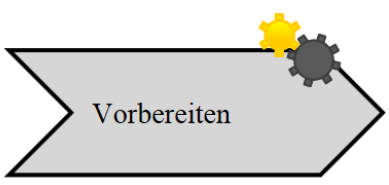


Abbildung 9.10: Rendering einer zustandsbehafteten Chevron-Aktivität in einem Webbrowser.

Weitere Listings zur Implementierung der Zustandsprojektion von BPEL zu Chevron, eine vollständige Beschreibung von Chevron in XML Schema sowie ein umfassendes Beispiel einer Zustandsprojektion anhand eines konkreten Szenarios können dem technischen Bericht von Schumm et al. [SKL+11] entnommen werden. Eine Implementierung der Zustandsprojektion von BPEL zu Chevron wurde in einer studentischen Arbeit vorgenommen [Lie11].

9.3 Erweiterungen von BPEL für rr-Fragmente

Nachfolgend werden Erweiterungen von BPEL für rr-Fragmente vorgestellt, um das in Abschnitt 6.3 beschriebene Konzept auf die Prozesssprache BPEL abzubilden. Die nachfolgende Beschreibung basiert auf der Arbeit Schumm et al. [SAL+10a], in der diese Erweiterungen vorgestellt und im Themenkomplex Compliance positioniert wurden. Die vorgeschlagenen Erweiterungen werden nur zur Entwurfszeit benötigt und erfordern keine Anpassung der Laufzeitumgebung, sondern lediglich eine Anpassung der Modellierungsumgebung. Außer der Erweiterung `ext:id` für eindeutige Identifikationsmerkmale werden alle Erweiterungselemente bei der Integration in den Prozess entfernt oder durch Standard-Elemente ersetzt, vgl. zur Integration Abschnitt 8.1. Durch Nutzung des in BPEL definierten Attributs `mustUnderstand="yes"` für den Namensraum der Erweiterungen, die hier durch das Präfix `frg` gekennzeichnet werden, wird die Ausführung eines Prozesses ausgeschlossen der diese Erweiterungen noch enthält.

Abbildung 9.11 zeigt die Charakteristika eines beispielhaften rr-Fragments zur Realisierung einer Genehmigungsanforderung. Erweiterungen von BPEL sind durch Nummern gekennzeichnet. Als Erweiterungen werden Eingänge (1), Ausgänge (2), Platzhalter (3), Annotationsmöglichkeiten (4) und der Kontext (5) unterschieden. Die Erweiterungen sind an dem graphbasierten Teil von BPEL ausgerichtet, der Aktivität `<flow>`.

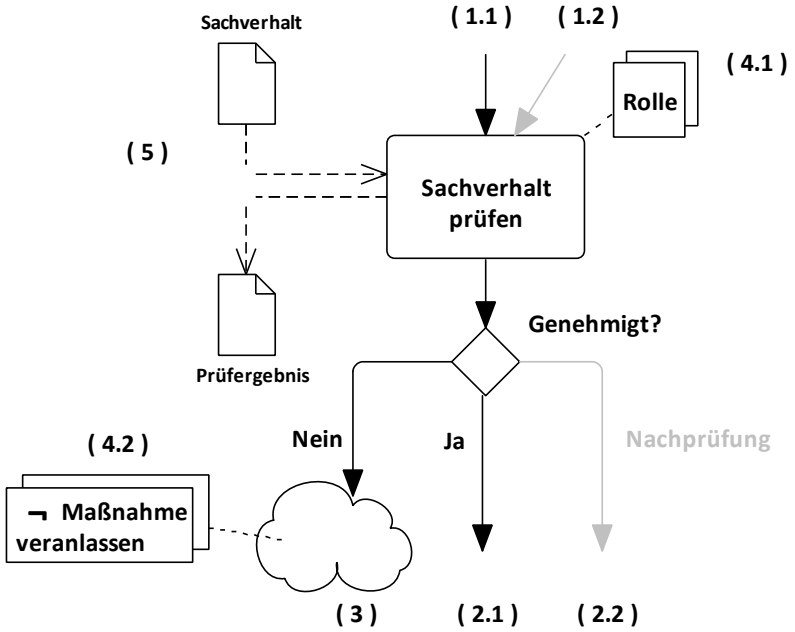


Abbildung 9.11: Kennzeichnung von Spracherweiterungen von BPEL für rr-Fragmente am Beispiel eines rr-Fragments zur Realisierung einer Genehmigungsanforderung.

Eingänge. Die Eingänge eines rr-Fragments kennzeichnen den eingehenden Kontrollfluss. Listing 9.8 zeigt das Schema der BPEL-Erweiterung `<fragmentEntry>` für den Eingang eines rr-Fragments. Ein fragmentEntry-Konstrukt kann mehrere Kontrollverbindungen haben, die von ihm ausgehen, jedoch kann es keine eingehenden Kontrollverbindungen besitzen. In BPEL entspricht eine Kontrollverbindung einem `<link>` in einem `<flow>`. Zur Kennzeichnung, dass der Eingang der Ursprung von Kontrollverbindungen ist, werden diese Kontrollverbindungen in dem `<sources>` Element als `<source>` mit deren Name aufgeführt. Das Zeichen `+` beim Element `<source>` in Listing 9.8 gibt an, dass das Element ein oder mehrere Male auftreten kann. Das Attribut `type` gibt für einen Eingang an, ob er verpflichtend in den Prozess integriert werden muss oder ob er bei Bedarf entfernt werden kann, ohne die regelrealisierende Semantik des rr-Fragments zu beeinträchtigen.

```
<frg:fragmentEntry name="entryName" type="mandatory|optional">
   <bpws:sources>
      <bpws:source linkName="linkName" />+
   </bpws:sources>
</frg:fragmentEntry>
```

Listing 9.8: Erweiterung von BPEL für den Eingang eines rr-Fragments [SAL+10a].

Ausgänge. Der Ausgang eines rr-Fragments kennzeichnet den ausgehenden Kontrollfluss zur Integration in einen Prozess. Ein Ausgang muss mindestens eine eingehende Kontrollverbindung haben. Dies wird durch das Element `<target>` dargestellt, in dem der `linkName` der eingehenden Kontrollverbindung genannt ist. Analog zu Eingängen können auch Ausgänge durch das Attribut `type` in verpflichtende und optionale Ausgänge unterschieden werden. Listing 9.9 zeigt das Schema der Erweiterung `<fragmentExit>` zur Modellierung eines Ausgangs eines rr-Fragments.

```
<frg:fragmentExit name="exitName" type="mandatory|optional">
   <bpws:targets>
      <bpws:target linkName="linkName"/>+
   </bpws:targets>
</frg:fragmentExit>
```

Listing 9.9: Erweiterung von BPEL für den Ausgang eines rr-Fragments [SAL+10a].

Platzhalter. Ein Platzhalter `<fragmentRegion>` zeichnet in einem rr-Fragment eine Stelle aus, an der andere Prozessstrukturen eingefügt werden können, ohne die regelrealisierende Semantik des rr-Fragments zu beeinträchtigen. Listing 9.10 zeigt das Schema dieser Erweiterung. Durch eingehende Kontrollverbindungen `<targets>` und ausgehende Kontrollverbindungen `<sources>` kann der Platzhalter in den Kontrollfluss innerhalb des rr-Fragments eingebunden werden. In BPEL kann zwar der Aktivitätstyp `<opaqueActivity>` in abstrahierten Prozessen oder in Prozessvorlagen als Platzhalter verwendet werden. Dieses Element hat in abstrahierten Prozessen allerdings die Semantik, Details einer Aktivität zu verbergen, aber die Existenz der Aktivität zu offenbaren. Um die Semantik dieses Elements

nicht zu verändern, wurde eine Erweiterung von BPEL für die Platzhalter in einem rr-Fragment vorgenommen.

```
<frg:fragmentRegion name="regionName">
   <bpws:targets>
      <bpws:target linkName="linkName"/>+
   </bpws:targets>
   <bpws:sources>
      <bpws:source linkName="linkName"/>+
   </bpws:sources>
</frg:fragmentRegion>
```

Listing 9.10: Erweiterung von BPEL für Platzhalter [SAL+10a].

Constraints. Das vorgestellte Konzept für rr-Fragmente erlaubt die Annotation von Platzhaltern und Variablen mit Constraints. Damit kann der Modellierer des rr-Fragments angeben, welche Anforderungen beim Ersetzen der Platzhalter oder beim Setzen von Werten zu beachten sind, um die regelrealisierende Semantik des rr-Fragments zu gewährleisten. Das Konzept sieht vor, verschiedene Sprachen zur Annotation zuzulassen. Bspw. kann eine DSL eingesetzt werden, wie etwa die von Schleicher et al. [SGL+11] vorgeschlagene Regelsprache mit der Anforderungen an den Datenfluss mit Anforderungen an den Kontrollfluss kombiniert werden können. Durch die Nutzung eines generischen Annotationsmechanismus können Constraints in einem beliebigen Format in vom Prozess separierten Dateien abgelegt werden. Dies setzt allerdings voraus, dass die einzuschränkenden Elemente des rr-Fragments eindeutig identifiziert werden können. Dies ist über einen XPath-Ausdruck zwar möglich, allerdings müssten derartige Pfadausdrücke bei der Änderung der Prozessstruktur angepasst werden. Daher besteht die Notwendigkeit einer Erweiterung zur eindeutigen Identifikation. Diese ist nicht Standardumfang der Prozesssprache BPEL.

Eindeutige Identifikation. Ein eindeutiges Identifikationsmerkmal an BPEL-Elementen erlaubt es, Constraints und Parameter außerhalb des Prozessmodells zu spezifizieren und durch einen Verweis auf ein Identifikationsmerkmal zu annotieren. Des Weiteren können die Bestandteile von rr-Fragmenten mit einem Identifikationsmerkmal versehen werden, um damit ein Ausblenden oder Hervorheben zu steuern. Wie die Arbeiten im Projekt COMPAS gezeigt haben, kann ein solches Identifikationsmerkmal zur verbesserten Nachvollziehbarkeit zu Events hinzugefügt werden. Diese Erweiterung nutzt mit dem Präfix `ext` und der Eigenschaft `mustUnderstand="no"` einen anderen Namensraum als die übrigen Erweiterungen, da sie vor einer Ausführung nicht entfernt oder ersetzt werden muss. Listing 9.11 zeigt die Erweiterung an einem Eingang, einem Platzhalter und einer Variable.

```
<!-- for constrained entries -->
<frg:fragmentEntry name="entryName" ext:id="unique" />

<!-- for constrained regions -->
<frg:fragmentRegion name="regionName" ext:id="unique" />

<!—for constrained variables -->
<bpws:variable name="variableName" ext:id="unique" />
```

Listing 9.11: Erweiterung von BPEL für eindeutige Identifikationsmerkmale [SAL+10a].

Kontext. Der Kontext eines rr-Fragments in BPEL kann `<variables>` und weitere in den Prozess zu übernehmende technische Aspekte enthalten, wie bspw. `<partnerLinks>` und `<correlationSets>`. In BPEL wird der Kontext in einer `<scope>` Aktivität definiert. Um den Kontext von einem rr-Fragment von einer `<scope>` Aktivität unterscheiden zu können, aber dessen Ausdrucksmöglichkeiten zu nutzen, wurde ein `<fragmentScope>` von einem herkömmlichen BPEL `<scope>` abgeleitet und ihm so dieselbe Charakteristik verliehen aber nicht dessen Ausführungssemantik übernommen. Der Kontext eines rr-Fragments geht bei der Integration in einen BPEL-Prozess in den Kontext des Prozesses über. Wie in Listing 9.12 gezeigt, dient der `<fragmentScope>` außerdem als Container für den eigentlichen Graphen des rr-Fragments, der in dem Konstrukt `<fragmentFlow>` untergebracht ist.

```
<frg:fragmentScope name="fragmentScopeName">
   <bpws:variables>
      <bpws:variable ... />+
   </bpws:variables>
   <!— Other context -->
   ...
   <frg:fragmentFlow name="fragmentFlowName">
```

Listing 9.12: Erweiterung von BPEL für den Fragmentkontext [SAL+10a].

Graph. Zur Repräsentation des Graphen eines rr-Fragments kommen in BPEL zwei Aktivitätstypen in Betracht, von denen die Eigenschaften abgeleitet werden können: die blockstrukturierte `<sequence>` Aktivität und die graphbasierte `<flow>` Aktivität. Für rr-Fragmente mit einer SESE-Einschränkung würde eine `<sequence>` Aktivität genügen. Um allerdings auch rr-Fragmente ohne diese Einschränkung abbilden zu können, ist die Semantik eines `<flow>` als Basis für die Repräsentation des Graphen eines rr-Fragments erforderlich. Ein `<fragmentFlow>` (vgl. Listing 9.13) ist ein Container für Aktivitäten und für den Kontrollfluss, der sie verbindet. Analog zum `<fragmentScope>` geht ein `<fragmentFlow>` bei der Integration in einen Prozess in den Prozess über, d.h. die Aktivitäten und `<links>` des `<fragmentFlow>` werden in einen bereits bestehenden `<flow>` im Prozesses überführt.

```
<frg:fragmentFlow name="fragmentFlowName">
   <bpws:links>
      <bpws:link name="linkName" />*
   </bpws:links>
   <frg:fragmentEntry ...
```

Listing 9.13: Erweiterung von BPEL für graphbasierte rr-Fragmente mit mehreren Eingängen oder Ausgängen [SAL+10a].

Das vollständige XML Schema der gezeigten Erweiterungen und ein rr-Fragment in XML kann in einem Ergebnisdokument [COM10a] aus dem Forschungsprojekt COMPAS nachgelesen werden.

KAPITEL 10

ARCHITEKTUR UND IMPLEMENTIERUNG EINER UNTERSTÜTZENDEN INFRASTRUKTUR

In diesem Kapitel werden Architekturen vorgestellt, mit denen wesentliche Aspekte der vorliegenden Dissertation realisiert werden können. Die Architekturen beschreiben den Aufbau von Komponenten und deren Verbindungen untereinander. Die diskutierten Architekturen sind weitgehend generisch und können als konzeptionelle Ausgangsbasis für unterschiedliche Implementierungen und Einsatzszenarien betrachtet werden. Zu jeder Architektur ist beschrieben, wie sie durch neu entwickelte Komponenten als auch durch die Anpassung und Nutzung bestehender Komponenten im Kontext der Prozesssprache BPEL prototypisch implementiert wurde. Die Implementierung ist im Rahmen von studentischen Arbeiten, in den Forschungsprojekten COMPAS[8] und SimTech[9] und in Eigenarbeit erfolgt.

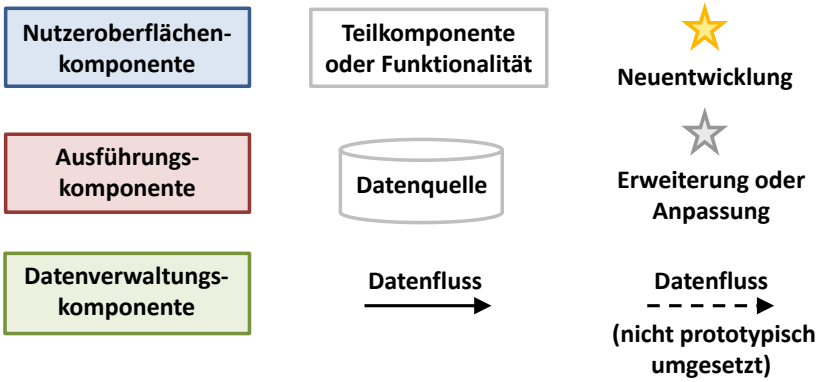


Abbildung 10.1: Legende zur Architekturbeschreibung.

Die Architekturdiagramme setzen sich aus verschiedenen grafischen Elementen zusammen, die in Abbildung 10.1 zusammengefasst sind. Die Farbcodierung entspricht der in dieser Arbeit eingesetzten Unterscheidung von Schichten und Komponenten mit wesentlichem Bezug zum Nutzer (blau), zur Ausführung (rot) und zur Datenhaltung (grün), vgl. Abschnitt 2.1.2. In den Architekturdiagrammen sind Marker enthalten, welche darüber Auskunft geben, ob eine Komponente oder Funktion in der Implementierung neu entwickelt wurde (goldener Stern), ob eine Anpassung einer bestehenden Komponente erforderlich war (silberner Stern) oder ob eine bestehende Komponente unverändert genutzt wurde (ohne Marker). Ein Marker

[8] COMPAS – Compliance-driven Models, Languages, and Architectures for Services
[9] SimTech – Stuttgart Research Centre for Simulation Technology und Exzellenzcluster Simulation Technology

wird von übergeordneten an enthaltene Elemente vererbt. Konzipierte, aber nicht implementierte Verbindungen zwischen Komponenten sind mit einem gestrichelten Pfeil dargestellt.

10.1 Architekturüberblick

Jede der in diesem Kapitel vorstellten Architekturen hat einen bestimmten Schwerpunkt und betrifft ein in sich abgeschlossenes Thema. Bspw. kann die dienstbasierte Realisierung von Sichten (vgl. Abschnitt 10.2) als eigenständige Lösung betrachtet werden, die den Prozessserver und Transformationsdienste integriert. Auch stellt die Monitoring-Plattform (vgl. Abschnitt 10.5) eine eigenständige Lösung dar, die ebenfalls den Prozessserver integriert. Durch die gemeinsam genutzten bzw. integrierten Komponenten der einzelnen Architekturen kann ein Zusammenhang hergestellt werden, der in Abbildung 10.2 dargestellt ist. Die Monitoring-Plattform kann in diesem Zusammenhang etwa dazu verwendet werden, generierte Dienstkompositionen zu überwachen. Die Fragment-Bibliothek (vgl. Abschnitt 10.3) trägt zur Unterstützung der Modellierung bei (vgl. Abschnitt 10.4), könnte aber auch im Zusammenhang mit der Monitoring-Plattform zur Hervorhebung von rr-Fragmenten sinnvoll eingesetzt werden. Zur Reduzierung der Komplexität abstrahiert der Architekturüberblick in Abbildung 10.2 von Details, etwa von der Übergabe von Parametern beim Monitoring. Die einzelnen Architekturbeschreibungen umfassen diese Details und gehen auf die Teilkomponenten, den Datenfluss und die verwendeten Technologien ein.

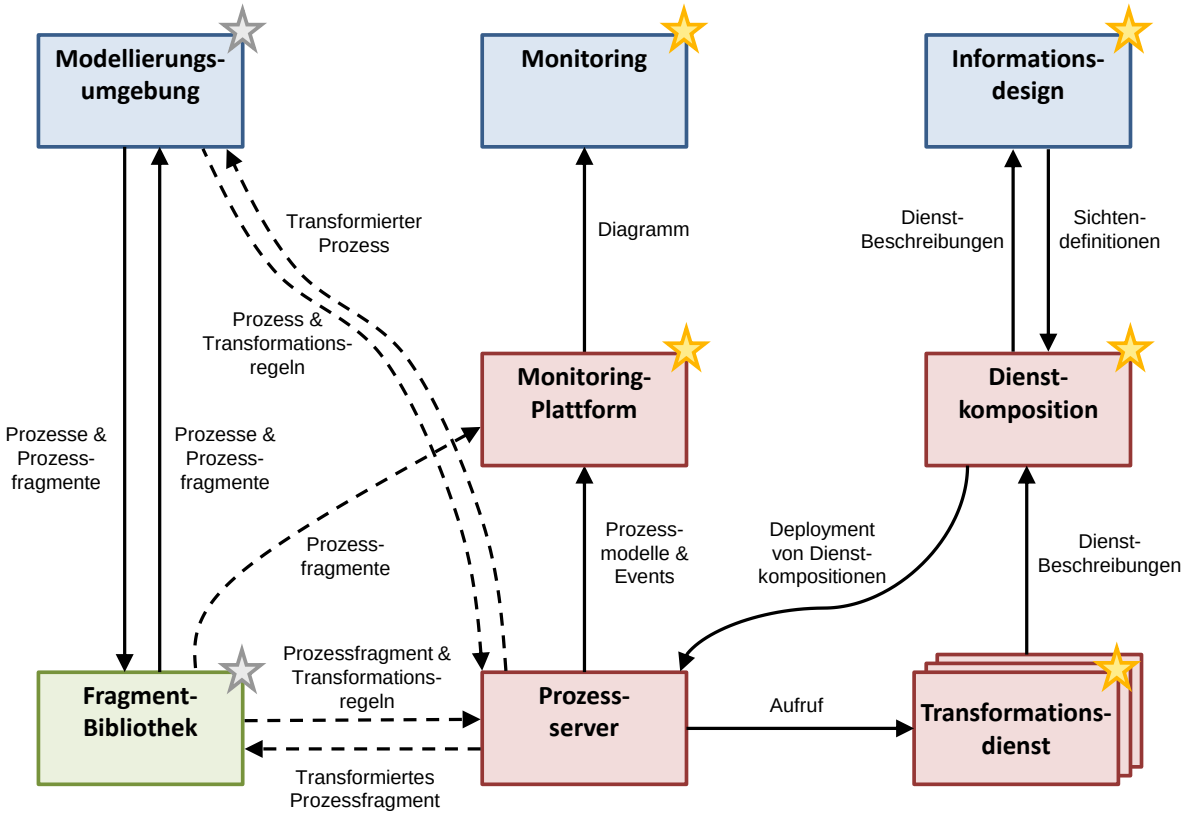


Abbildung 10.2: Architekturüberblick und Zusammenhang der Komponenten.

Während die Modellierungsumgebung und die Fragment-Bibliothek Erweiterungen und Anpassungen von bestehenden Systemen darstellen, wurden die Monitoring-Plattform, die Dienstkomposition und die Transformationsdienste im Rahmen der vorliegenden Dissertation neu entwickelt. Die Monitoring-Plattform und die Dienstkomposition wurden als webbasierte Anwendungen entwickelt, die mit einem Webbrowser genutzt werden können. Die als Neuentwicklung gekennzeichneten Nutzeroberflächenkomponenten sind in diesem Sinne nicht als Webbrowser, sondern als Frontend-Teilkomponenten der Web-Applikationen zu verstehen. In den Implementierungen wurden grundsätzlich bestehende Frameworks, Application Server, Datenbanksysteme, Middleware und Webbrowser verwendet.

10.2 Dienstkomposition zur Realisierung von Sichten

Das Konzept der dienstbasierten Realisierung von Sichtenentwürfen und beispielhafte Transformationsdienste wurden in Abschnitt 9.1 vorgestellt. Nachfolgend wird eine Architektur und Implementierung vorgestellt, die diese Teilaspekte in einem System integriert.

10.2.1 Architektur

Die nachfolgende Architekturbeschreibung basiert auf dem Artikel zur Komposition von Transformationsdiensten von Schumm et al. [SCF+11]. Das System erlaubt es, eine Sequenz aus Dienstaufrufen zu modellieren und daraus ausführbaren BPEL-Code zur Orchestrierung der gewählten Dienste automatisch generieren zu lassen. Die Architektur ist nach einem 3-Schichten-Modell aufgebaut (engl. *three-tier architecture*), vgl. Abbildung 10.3.

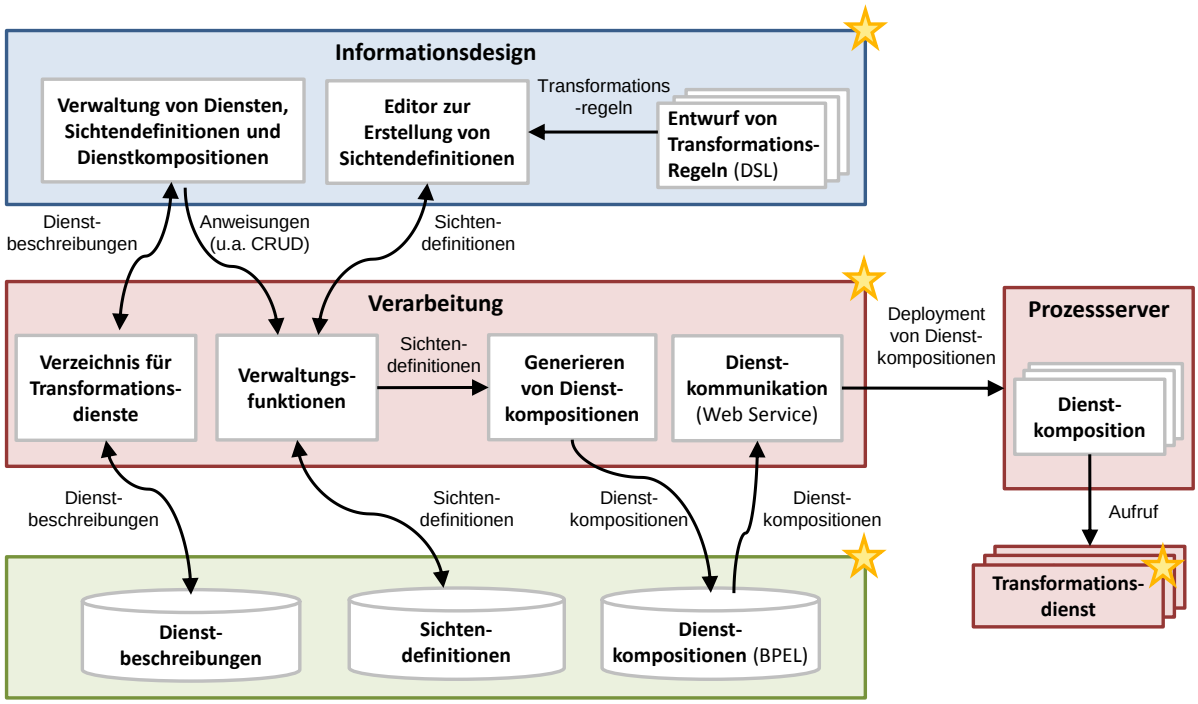


Abbildung 10.3: Architektur der Dienstkomposition zur Realisierung von Sichten, angepasst nach [SCF+11].

Die Nutzerschicht bietet verschiedene administrative Funktionen an: Es können verfügbare Dienste durch den Import von deren Schnittstellenbeschreibung (WSDL) im System registriert und bei Bedarf wieder entfernt werden; Sichtendefinitionen können erstellt, geändert oder gelöscht werden; Dienstkompositionen können aus Sichtendefinitionen generiert und vom System heruntergeladen oder direkt auf einen Prozessserver deployed werden. Der Editor zur Erstellung von Sichtendefinitionen ist die Komponente zur inhaltlichen Arbeit im System. Dieser Editor erlaubt die Auswahl von Operationen, die von registrierten Diensten angeboten werden. Ausgewählte Operationen können parametriert und in die gewünschte Reihenfolge gebracht werden. Dienstspezifische Editoren für Transformationsregeln unterstützen an dieser Stelle die Dienst-Parametrierung.

Dem Nutzer angebotene Verwaltungsfunktionen werden von der Geschäftsschicht verarbeitet. Das Verzeichnis für Transformationsdienste übernimmt die Funktion, die WSDL-Schnittstellendatei in seine Informationsbestandteile zu zerlegen und diese für Abfragen zugänglich zu machen. Für ein späteres Deployment von Dienstkompositionen wird die Schnittstellenbeschreibung zusätzlich in unveränderter Form abgelegt. Die Funktion zum Generieren von Dienstkompositionen erzeugt einen BPEL-Prozess mit einer Sequenz aus Dienstaufrufen, entsprechend der Sichtendefinition. Eine generierte Dienstkomposition kann über den Aufruf einer bestimmten Operation des Prozessservers auf diesen deployed werden. Dieser Aufruf wird von einem Kommunikationsmodul durchgeführt. Die einfach gehaltene Datenschicht kann bspw. durch Tabellen in einer Datenbank abgebildet werden.

Der linke Teil der Geschäftsschicht, die Verarbeitung, repräsentiert die Entwurfszeit. Der rechte Teil stellt die Laufzeit dar, bestehend aus Prozessserver, generierten Dienstkompositionen und Transformationsdiensten. Nach dem Deployment können die komponierten Transformationsdienste aus anderen Anwendungen heraus durch einen Aufruf des Prozessservers genutzt werden und zudem können sie als neuer, kompositer Transformationsdienst registriert werden. Ein Aufruf kann bspw. aus einer Modellierungsumgebung oder aus einem Werkzeug zur Prozessanalyse heraus erfolgen. Ein Werkzeug zur Prozessanalyse unter Verwendung generierter Dienstkompositionen wurde im Rahmen der vorliegenden Dissertation nicht entwickelt, aber im Rahmen einer Zusammenarbeit konzipiert [NKL+12].

10.2.2 Implementierung

Die Dienstkomposition zur Realisierung von Sichten wurde als webbasierte Anwendung implementiert [Cai11]. Die Benutzerschicht der Anwendung wird über das Framework Apache MyFaces [Apa11b] (JavaServer Faces, JSF) gesteuert und ist über einen Webbrowser zugänglich. Die Geschäftsschicht wurde in Java unter Einsatz des Spring Frameworks [Spr11] entwickelt und als Web Archive (WAR) auf einem Apache Tomcat [Apa11c] Application Server eingerichtet. Die Datenschicht wurde mit MySQL [Ora11] und dem Framework Hibernate [JBo11] realisiert. Zur Kommunikation mit dem Prozessserver wurde das Web Service Framework Apache Axis [Apa11a] verwendet. Als Prozessserver kam in dieser, wie auch in den anderen Implementierungen, die BPEL-Engine Apache ODE [Apa11d] zum Einsatz. Der Screenshot in Abbildung 10.4 zeigt eine in der Anwendung erstellte Sichtendefinition.

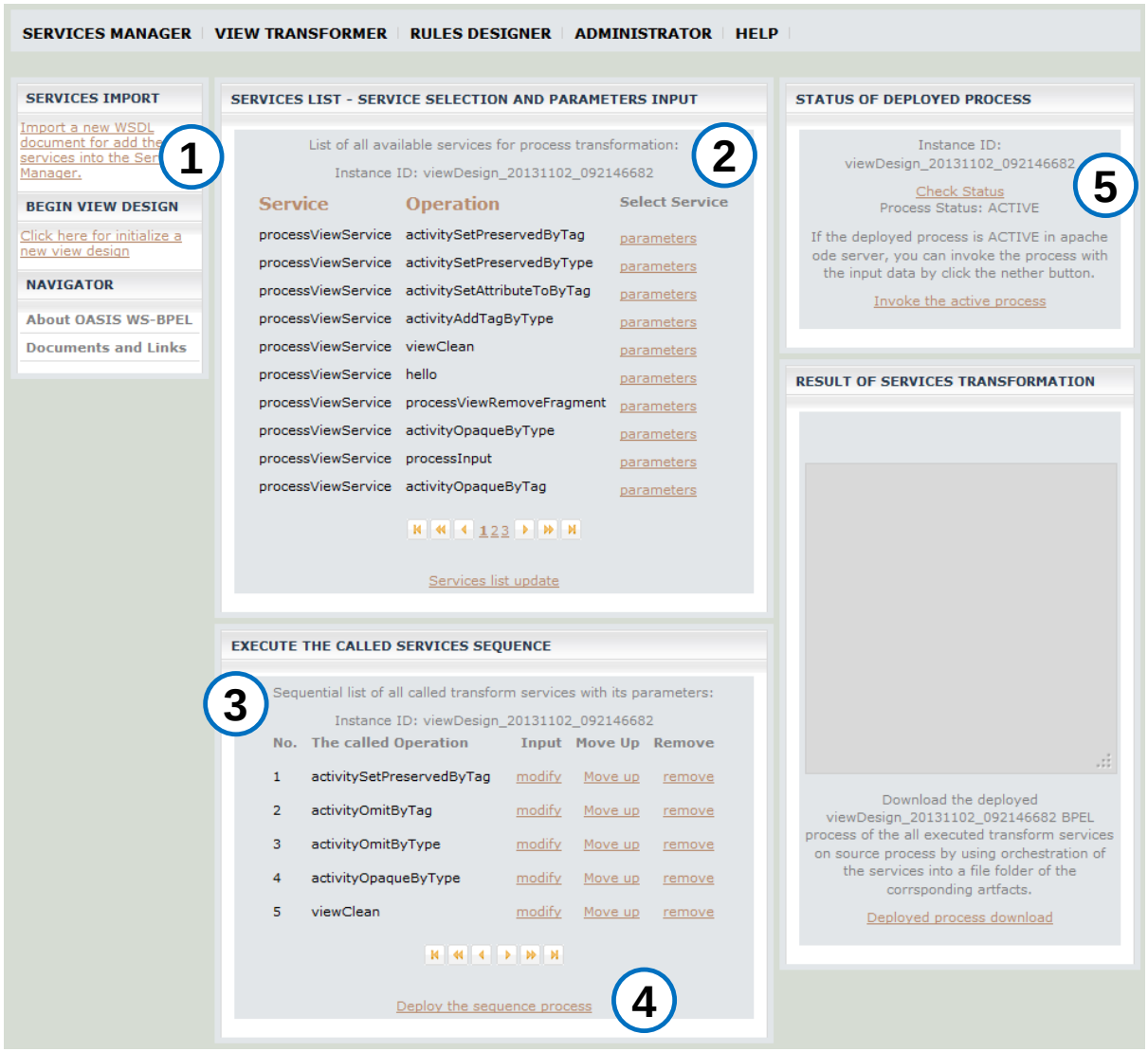


Abbildung 10.4: Screenshot zur Sichtendefinition in der Implementierung nach [Cai11].

Die Oberfläche enthält verschiedene Funktionen: (1) WSDL-Schnittstellendefinitionen von Transformationsdiensten können importiert werden. Beim Import werden die von dem Dienst angebotenen Operationen zur Liste verfügbarer Operationen hinzugefügt. (2) Aus dieser Liste können Operationen ausgewählt und für den konkreten Einsatz in einer Sichtendefinition mit statischen Eingabewerten versehen werden. (3) Ein konfigurierter Dienstaufruf wird anschließend an das Ende der Sichtendefinition angehängt. Die Sichtendefinition kann durch weiteres Hinzufügen von Dienstaufrufen, durch Änderung der Reihenfolge und durch Änderung der Parameter angepasst werden. (4) Aus dieser Sichtendefinition kann anschließend ein BPEL-Prozess generiert werden und dieser mitsamt der erforderlichen Laufzeit-Artefakte, d.h. WSDL und Deployment Descriptor, über eine Web Service Schnittstelle an den Prozessserver zum Deployment übermittelt werden. (5) Der Deployment-Status kann über einen Web Service Aufruf beim Prozessserver abgefragt werden. Nach erfolgreichem Deployment kann die Dienstkomposition zur Erzeugung von Sichten genutzt werden.

Parameter des Dienstaufrufs `activityOmitByType` (Aktivitäten entfernen, M1) sind in Abbildung 10.5 dargestellt. In diesem Beispiel sollen alle Aktivitäten vom Typ `<assign>`

entfernt werden. Jede Dienstoperation hat einen Parameter zur Eingabe des zu transformierenden Eingabeprozesses (1). Durch Auswahl einer Option (2) kann der Nutzer angeben, dass der Ausgabeprozess des vorangehenden Dienstaufrufs als Eingabe verwendet werden soll. Typischerweise ist diese Option nur beim allerersten Dienstaufruf einer Sichtendefinition abgewählt. Auch können statische Parameter hinterlegt werden (3). Statische Parameter sind im generierten BPEL-Prozess hart codiert und können beim Aufruf der Dienstkomposition nicht übergeben werden.

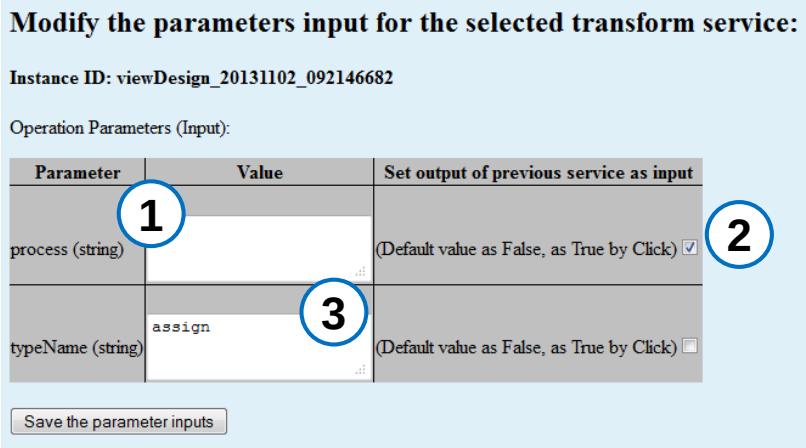


Abbildung 10.5: Screenshot zu Parametern eines Dienstaufrufs in der Implementierung nach [Cai11].

10.3 Bibliothek für Prozessfragmente

Im Rahmen des Projekts COMPAS wurde der Prototyp „Fragmento“ [SAD+09] realisiert, ein Repository zur versionierten Verwaltung von Prozessen, Prozessfragmenten im Allgemeinen und rr-Fragmenten im Besonderen, als auch von prozessbezogenen Artefakten für die Prozesssprache BPEL. Die Systemarchitektur von Fragmento wurde von Schumm et al. [SKL+10] beschrieben und die Implementierung in [Sch11a] dokumentiert. In Schumm et al. [SKK+11] wurden weiterführende Nutzungsmöglichkeiten mehrerer Bibliotheken in der Entwicklung prozessbasierter und regelkonformer Anwendungen diskutiert. In ähnlicher Ausgestaltung wurde das Konzept von Prozessbibliotheken für ausführungsnahe Prozesssprachen auch von anderen Autoren realisiert, bspw. das BPEL Reuse Repository von Ma [Ma12] oder Apromore von La Rosa et al. [RRA+11]. Erste Ansätze zur Nutzung von Bibliotheken von Prozessbausteinen reichen jedoch weit länger zurück, vgl. bspw. [RFK+00]. Nachfolgend wird die Architektur und Implementierung der Bibliothek Fragmento nach Schumm et al. [SKL+10] vorgestellt.

10.3.1 Architektur

Die Architektur der Prozess- und Prozessfragmentbibliothek Fragmento ist in Abbildung 10.6 dargestellt. Die Nutzerschicht bietet zwei Zugangsarten. Zum einen werden alle Funktionen als Web Service zur Verfügung gestellt. Mit dieser Schnittstelle können Anwendungen wie bspw. Modellierungswerkzeuge eine nahtlose und automatisierte Integration mit der Biblio-

thek herstellen, wie in Abschnitt 10.4 diskutiert wird. Zum anderen bietet die Bibliothek ein Web Frontend, über das sie interaktiv mit einem Standardbrowser bedient werden kann.

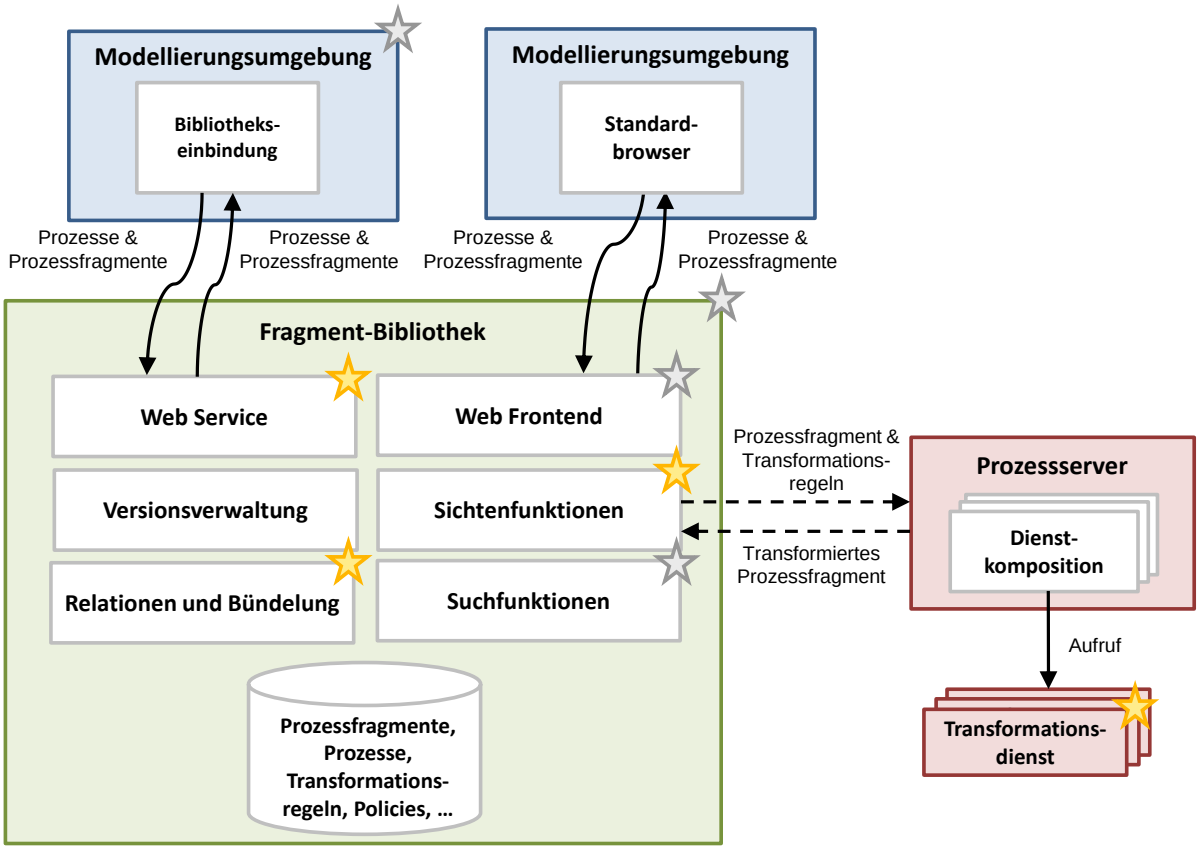


Abbildung 10.6: Architektur der Bibliothek für Prozessfragmente, angepasst nach [SKL+10].

Fragmento bietet verschiedene Verwaltungsfunktionen für die gespeicherten Artefakte an: Erstellen, Einchecken, Auschecken und die Verwaltung von Locks sind für eine derartige Bibliothek obligatorisch; Relationen und Bündelung erlauben es, typisierte Verknüpfungen zwischen Artefakten herzustellen. Eine Relation vom Typ *Annotation* drückt etwa die Annotation eines Prozesses mit einem Prozessfragment aus. Dadurch lässt sich in der Bibliothek bspw. abbilden, welche rr-Fragmente in einem Prozess enthalten sind. Ein weiterer wichtiger Relationstyp ist *Container*. Mit dieser Relation werden Artefakte gebündelt und als zusammengehörig behandelt. Dies dient dem Zweck, alle zusammengehörigen Artefakte in einem Dienstaufruf gebündelt abfragen zu können, bspw. einen BPEL-Prozess mit zugehörigen WSDL-Dateien. Verschiedene Suchfunktionen unterstützen die Navigation über den Datenbestand der Bibliothek, wie bspw. über ein Schlüsselwort mit Volltextsuche, über die Auswahl des Artefakttyps oder über einen Erstellungszeitraum.

In der Bibliothek sind Sichtenfunktionen für reduzierende Strukturveränderung integriert, vgl. dazu die Sichtenfunktionen in Abschnitt 9.1.2. Die für den Aufruf der Sichtenfunktionen erforderlichen Transformationsregeln werden als separate Artefakte in der Bibliothek abgelegt. Beim Anzeigen und beim Auschecken von Prozessen und Prozessfragmenten können diese Transformationsregeln ausgewählt und angewendet werden. Anwendungsfälle für diese

Funktionalität sind das Entfernen von Erweiterungsattributen und Informationen für bestimmte Nutzergruppen, wodurch ein eingeschränktes Leserecht realisiert werden kann.

10.3.2 Implementierung

Die Fragment-Bibliothek wurde im Rahmen des Projekts COMPAS auf Basis des XMLO Repositorys [Ans09] entwickelt. Dieses Repository wurde passend konfiguriert, Web Service Schnittstellen neu definiert, das Web Frontend angepasst und eine Erweiterung für Relationen und Bündelung vorgenommen. Analog zur Dienstkomposition zur Realisierung von Sichten wurde die Bibliothek für Prozessfragmente in Java unter Einsatz von Spring, Hibernate und Apache Axis entwickelt und als Web Archive auf einem Apache Tomcat eingerichtet. Das Web Frontend wurde mit JavaServer Pages (JSP) und der Displaytag Library [Dis11] implementiert. Als Datenbanksystem wurde PostgreSQL [Pos10a] eingesetzt. Abbildung 10.7 zeigt die Erstellung eines Prozessfragments im Web Frontend.

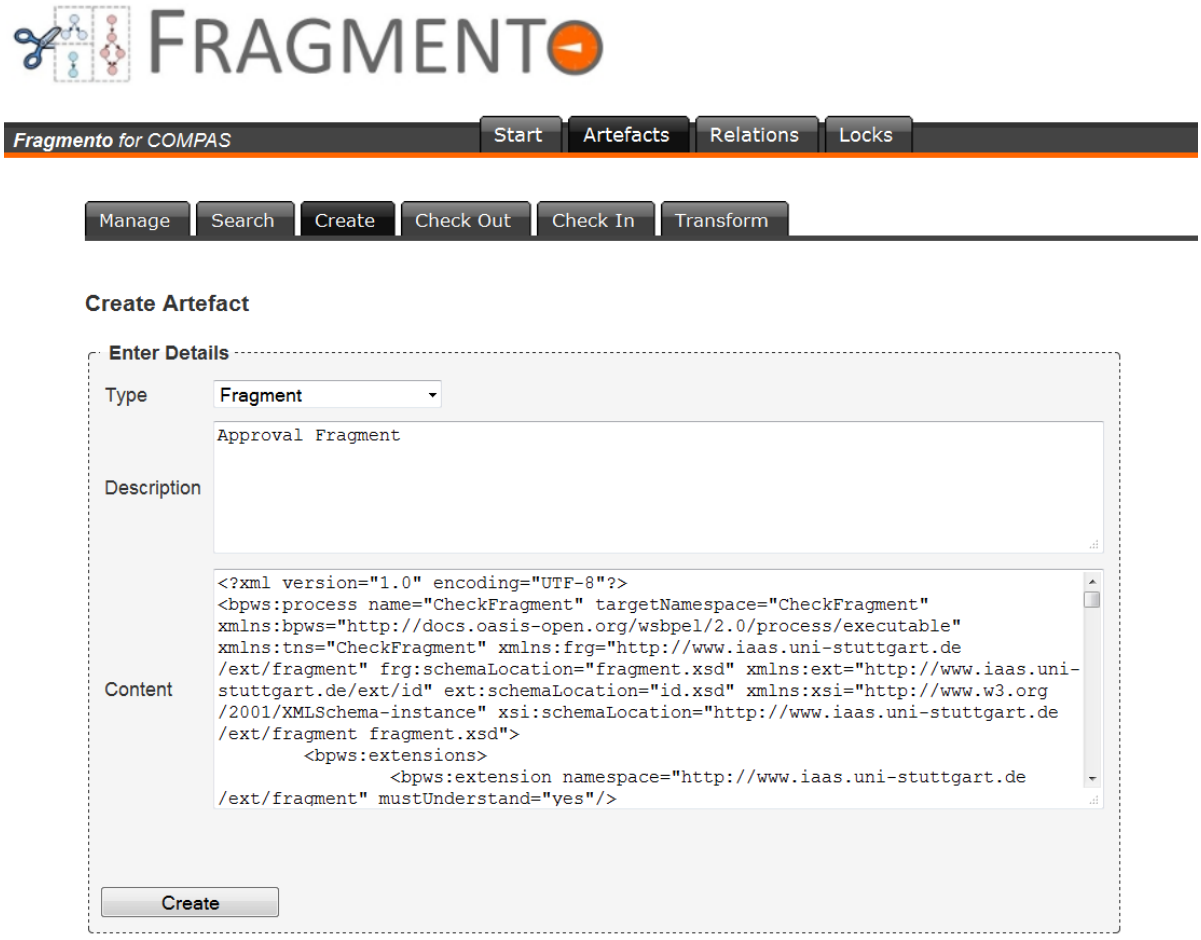


Abbildung 10.7: Erstellen eines Prozessfragments in der Bibliothek für Prozesse und Prozessfragmente.

Zur initialen Befüllung der Bibliothek wurde mit SoapUI [Evi10] eine Testsuite erstellt. Die Ausführung dieser Testsuite bewirkt das Einchecken verschiedener beispielhafter Artefakte und die Herstellung von Relationen zwischen diesen Artefakten. Aus Performancegründen und für eine einfachere Installation wurden die Sichtenfunktionen als Java-Bibliothek integriert. Im Zuge der Integration mit einer Modellierungsumgebung wurden nach der ini-

tialen Implementierung Anpassungen an der Bibliothek durch Hahn und Sonntag [SDH+12] vorgenommen. Weitere Details können in der Dokumentation nachgelesen werden [Sch11a].

10.4 Modellierungsumgebung für Prozesse

Die vorgestellten Konzepte zu Sichten auf Prozesse, Compliance und dessen Unterstützung durch rr-Fragmente erfordern einen unterschiedlich tiefen Eingriff in eine Modellierungsumgebung. Die Funktionalität, rr-Fragmente aus einer Bibliothek in einen bestehenden Prozess zu integrieren erfordert einen hohen Aufwand und zahlreiche Anpassungen. Auch erfordert die Anpassung vom Erscheinungsbild von Aktivitäten einen tieferen Eingriff. Der Aufruf einer Transformationsfunktion, bei der keine Verbindung zum originalen Prozessmodell erhalten werden muss, kann hingegen mittels eines Dienstaufrufs in bestehende Umgebungen integriert werden. Die Möglichkeit, Prozesselemente mit Annotationen zu versehen erfordert ebenfalls nur geringfügige Anpassungen.

10.4.1 Architektur

Die Architektur der Modellierungsumgebung berücksichtigt vier Aspekte: (1) Prozessfragmente zur Unterstützung der Modellierung regelkonformer Prozesse, (2) Anpassung der Darstellung von Aktivitäten, (3) Annotationen zur Unterstützung der Behandlung von rr-Fragmenten und zur Unterstützung der Sprachemulation durch Chevron und (4) den Aufruf von Diensten und Dienstkompositionen. Die Manipulation eines Prozesses in einer abstrahierenden Sicht wurde im Rahmen der Anforderungsanalyse (vgl. Abschnitt 2.4.2) eingeschränkt, da eine angemessene Betrachtung dieses Problems den Rahmen der vorliegenden Dissertation gesprengt hätte. Stattdessen wurde der Schwerpunkt auf den Entwurf von Sichten sowie auf die Erzeugung von Sichten zur Darstellung gelegt. Funktionen zum Ausblenden und Aggregieren von Aktivitäten wurden in die Monitoring-Plattform zur Darstellung von Prozessinstanzen integriert, vgl. dazu Abschnitt 10.5. Die resultierende Architektur der Modellierungsumgebung ist in Abbildung 10.8 dargestellt.

Der Prozesseditor stellt die Funktionalität zur Erstellung und Änderung von Prozessmodellen bereit. Diese Komponente setzt ein bestimmtes Prozessmetamodell voraus. Wird dieses Metamodell angepasst oder erweitert, wie etwa zur Nutzung der abstrakten Aktivität `<opaqueActivity>` oder für Erweiterungskonstrukte zur Modellierung von rr-Fragmenten, so ist eine Anpassung dieser Komponente erforderlich. Zusätzlich muss die grafische Oberfläche angepasst werden, bspw. muss die Palette von Prozesselementtypen um neue Typen erweitert werden, damit diese vom Nutzer verwendet werden können.

Die Integration eines Prozessfragments in bestehende Prozesse erfolgt unter Einsatz eines Assistenten (engl. *wizard*), welcher die zur Integration erforderlichen Parameter vom Modellierer abfragt, den Code in die interne Repräsentation des Prozesses einfügt und den Kontext des Prozessfragments in den Prozesskontext überführt.

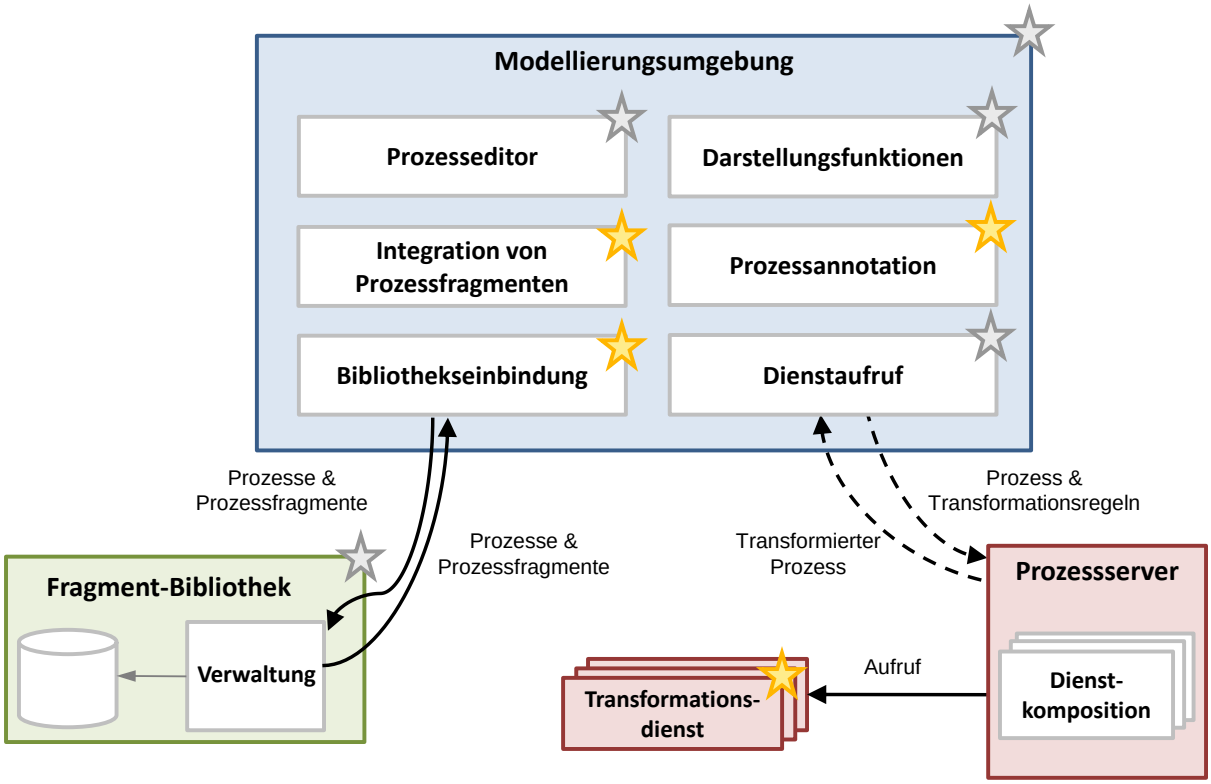


Abbildung 10.8: Architektur der Modellierungsumgebung für Prozesse.

Die Bibliothekseinbindung zur Verwaltung und zum Austausch von Prozessfragmenten stellt eine Komponente dar, die funktional davon gesondert betrachtet werden kann, aber gewisse Verbindungspunkte zur Integration mit der Palette von Prozesselementtypen besitzt. Wie von Schumm et al. [SDH+12] architekturseitig diskutiert, ermöglicht die Bibliothekseinbindung das Suchen, Abfragen, Einchecken, Auschecken und Austauschen von rr-Fragmenten und erleichtert auf diese Weise die Wiederverwendung.

Die Darstellungsfunktionen sind ein integraler Bestandteil einer grafischen Modellierungsumgebung. Schumm et al. [SKL+09] haben etwa gezeigt, wie eine Anpassung der Darstellungsfunktionen von Aktivitäten (M16) für eine Sprachemulation von BPEL durch BPMN genutzt werden kann. Die Anpassung von Darstellungsfunktionen kann zudem zur Hervorhebung von rr-Fragmenten oder beliebig anderen Prozessstrukturen eingesetzt werden.

Die Prozessannotation (M13) erfordert eine Anpassung des Metamodells der Modellierungsumgebung, sofern Annotationen nicht in bestehenden Strukturen wie bspw. in einem `<documentation>` Element platziert werden. Eine Platzierung von Annotationen in dem Prozessmodell stellt einerseits eine Prozessbelastung dar, erfordert andererseits keinen tiefen Eingriff in den Prozesseditor und dessen Metamodell. Wenn die Prozesselemente mit einem eindeutigen Identifikationsmerkmal versehen werden, kann eine vom Prozessmodel separierte Ablage von Annotationen unter Einsatz entsprechender Referenzierung erfolgen.

Dienstaufrufe binden einzelne Transformationsdienste oder deren Kompositionen ein. Ein Dienstaufruf erzeugt eine vom Originalprozess entkoppelte Sicht, setzt aber dennoch eine gewisse Integration in die Modellierungsumgebung voraus: Eine Nutzeranforderung zum

Dienstaufruf muss registriert werden, der Dienstaufruf muss vorbereitet und ausgeführt werden, der Ausgabeprozess muss wieder entgegengenommen und persistiert werden und abschließend muss die erzeugte Sicht im Prozesseditor geöffnet werden. Für die Durchführung all dieser Schritte muss der Dienstendpunkt bekannt sein und die erforderlichen Parameter vom Nutzer eingegeben oder statisch hinterlegt worden sein.

Bei der Einbindung von Diensten zur Sichtenerzeugung in ein BPEL-Modellierungswerkzeug zeigt sich eine Problematik der technischen Realisierung: Ein ausführbarer BPEL-Prozess besteht nicht nur aus dem Prozessmodell in Form einer BPEL-Datei, sondern auch aus Schnittstellen- und Datentypbeschreibungen in WSDL und XML Schema. Zusätzlich können weitere, für das Modellierungswerkzeug spezifische Dateien bestehen, etwa zur Ablage von Informationen zum Layout von Aktivitäten in einem `<flow>` in X/Y-Koordinaten. Sofern auch werkzeugspezifische Artefakte von einem Dienst zur Sichtenerzeugung verarbeitet werden sollen, ist dieser an das Modellierungswerkzeug gebunden. Das Ausschneiden eines rr-Fragments kann zwar als Dienstaufruf bewerkstelligt werden, jedoch liegt die Realisierung als integrierte Funktion näher. Dies legt den Schluss nahe, dass das Anwendungsgebiet der dienstbasierten Sichtenrealisierung weniger in der Modellierung, sondern vielmehr in der grafischen Prozessanalyse zu sehen ist.

10.4.2 Implementierung

Die Erläuterung der Implementierung beginnt mit einer musterbasierten Analyse des Modellierungswerkzeugs Eclipse BPEL Designer [Ecl11]. Anschließend werden Anpassungen an diesem Werkzeug vorgestellt, die von verschiedenen Beteiligten im Rahmen studentischer Arbeiten und in Zusammenarbeit in SimTech vorgenommen wurden.

Musterbasierte Analyse. Der Eclipse BPEL Designer ist ein modular aufgebautes Werkzeug, welches aus mehreren Plug-Ins besteht, welche das Eclipse Framework um Funktionalität zur Modellierung von BPEL-Prozessen erweitern. Eclipse bietet mit Perspektiven (engl. *perspectives*) die Möglichkeit, verschiedene Fenster (engl. *views*) zu öffnen und auf der Werkzeugoberfläche zu arrangieren. Die einzelnen Fenster können untereinander in Abhängigkeit stehen: Wird bspw. im Hauptfenster der BPEL Perspektive, in dem der BPEL-Prozessgraph dargestellt wird, eine Aktivität ausgewählt, so werden automatisch dessen Eigenschaften in das Eigenschaftsfenster (engl. *properties view*) geladen. Der Eclipse BPEL Designer implementiert bereits eine Vielzahl der Sichtenentwurfsmuster, vgl. dazu die Fensterkennzeichnungen in Abbildung 10.9.

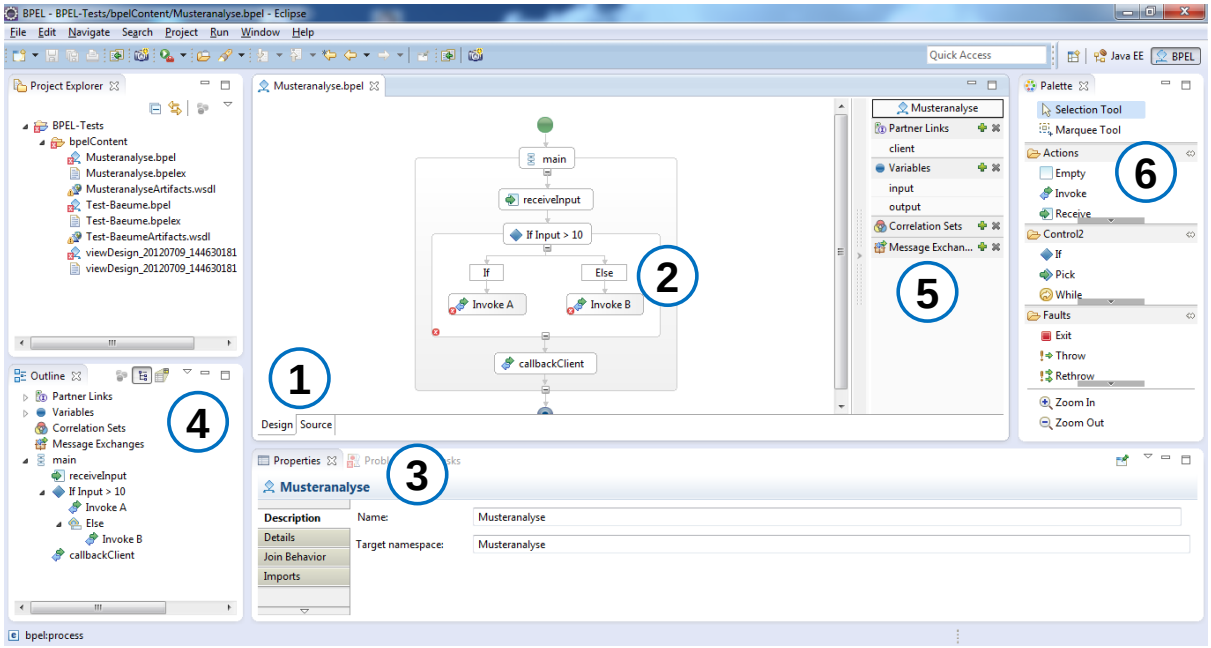


Abbildung 10.9: Screenshot des Eclipse BPEL Designer [Ecl11] für die musterbasierte Werkzeuganalyse.

Im Hauptfenster (vgl. 1) erfolgt die Prozessdarstellung entweder grafisch in einer Mischung aus Graphschema (M19) und Blockschema (M21) oder in Code (M22). Der Prozessgraph kann in der grafischen Darstellung entweder mit einer oben-unten oder einer links-rechts Orientierung ausgelegt werden (M24). Das Layout von Aktivitäten in einem `<flow>` kann durch den Nutzer auch manuell vorgegeben werden (M13, M28). Der Prozessgraph fokussiert mit der schlichten Aktivitätsdarstellung (M16) mit einem Symbol und der Aktivitätsbezeichnung auf die Prozessstruktur. Erst über eine MouseOver-Funktion werden die Eigenschaften in einem Kontextfenster dargestellt. Im Graph- wie auch im Codeschema können strukturierte Aktivitäten aggregiert werden (M2), wie z.B. ein `<if>` Block (vgl. 2). Ein Ausblenden von Aktivitäten ist in dem Werkzeug derzeit nicht gegeben. Die Durchführung einer Syntaxprüfung liefert eine statische Anreicherung des Prozesses (M14). Diese angereicherte Information kann im Erscheinungsbild von Aktivitäten (M16) und in der Darstellung von Verbindungen (M17) durch Symbole codiert werden, welche über das bestehende Erscheinungsbild von Aktivitäten gezeichnet werden (Fehlersymbole bei 2).

Das Eigenschaftenfenster (vgl. 3) zeigt das ausgewählte Element in Textform an. Der Einsatz von Sichtenentwurfsmustern wirkt unhandlich und letztlich unpraktisch, um dieses Fenster zu beschreiben: Eine entsprechende Sicht würde das ausgewählte Element bewahren/selektieren (M31), alle Aktivitäten entfernen (M1), alle Datenelemente entfernen (M5) und das verbleibende Element mit einem textuellen Erscheinungsbild darstellen (M16, M17, M18).

Der Prozess wird neben dem Hauptfenster auch in einem Grundriss (engl. *outline*) in einem Baumschema (M20) darstellt (vgl. 4). In dem Baumschema werden Aktivitäten (M16), Datenelemente (M18) und weitere Artefakte gezeigt. Kontrollverbindungen, die als `<link>` in einem `<flow>` enthalten sind, können im Baumschema nicht darstellt werden und somit hat dieses Fenster einen anderen Informationsgehalt als das Hauptfenster. Datenelemente werden im Hauptfenster in einem gesonderten Bereich zugänglich gemacht (vgl. 5). Die für die

Modellierung erforderlichen Typen des Prozessmetamodells werden in einer Palette angeboten (vgl. 6). Diese Palette enthält zudem eine Funktion für ein optisches Zoomen.

Vorgenommene Anpassungen. Die Anpassungen umfassen (1) die Erweiterung des Metamodells zur Unterstützung abstrakter Aktivitäten und Prozesse, (2) Anpassungen für die Annotation von Prozessen, (3) die Möglichkeit des Dienstaufrufs zum Ausschneiden von entsprechend annotierten Prozessfragmenten, (4) die Einbindung der Bibliothek für Prozessfragmente und (5) Darstellungsanpassungen. Abbildung 10.10 nach Schumm et al. [SDH+12] zeigt an, wo diese Anpassungen dem Nutzer zugänglich gemacht werden.

1. Abstrakte Aktivitäten und Prozesse. Die Erweiterung des Prozessmetamodells um die abstrakte Aktivität `<opaqueActivity>` (M4) und explizit als abstrahiert gekennzeichnete Prozesse wurde in der Arbeit von Streule [Str09] durch einen Eingriff in den Quellcode des Werkzeugs realisiert. Diese Erweiterung wurde an das Projekt Eclipse BPEL Designer [Ecl11] übergeben und in den öffentlichen Quellcode des Werkzeugs übernommen.

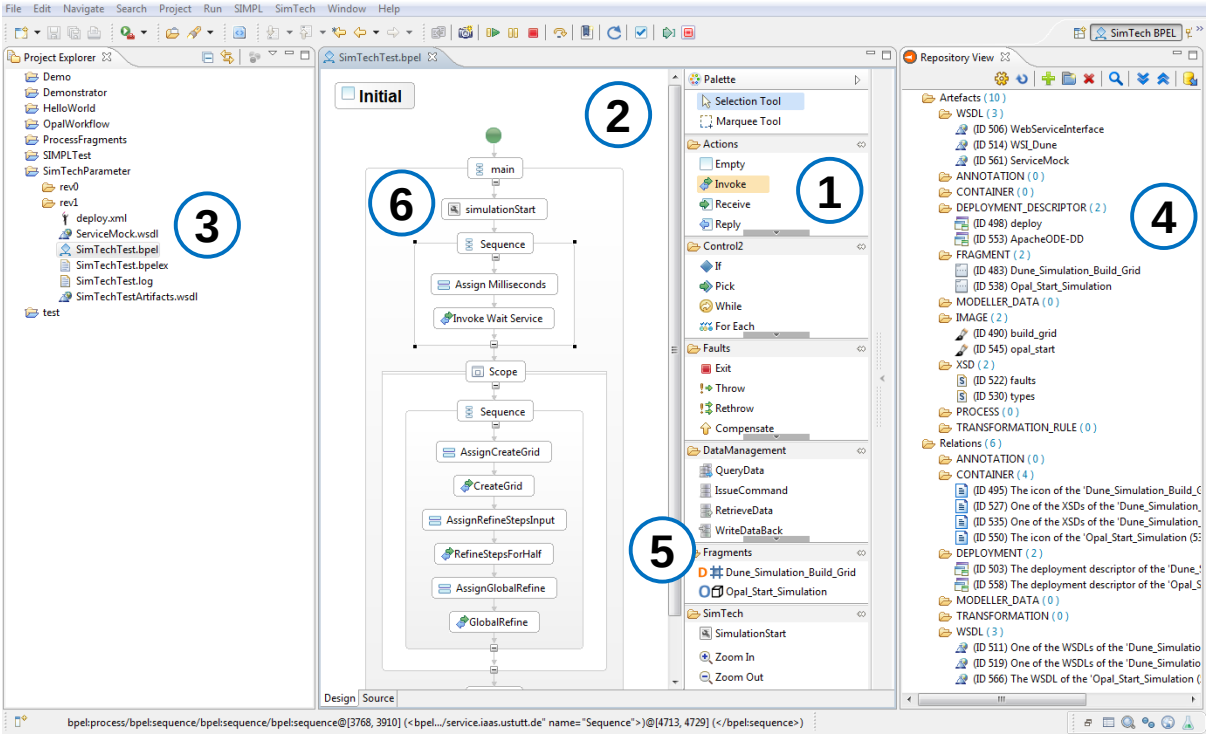


Abbildung 10.10: Screenshot der erweiterten Modellierungsumgebung für Prozesse [SDH+12].

2. Annotation von Prozessen. Die Annotation von Aktivitäten durch den Nutzer (M13) wurde in der Arbeit von Streule [Str09] durch eine Erweiterung des Aktivitätskontextmenüs mit einem Eclipse Plug-In gezeigt. Diese Implementierung ermöglicht es, vordefinierte Annotationen `Public`, `Private` und `Confidential` zur Kennzeichnung von Sichtbarkeitsstufen in der Aktivitätsdokumentation zu hinterlegen sowie zu bewahrende Aktivitäten mit der Annotation `Preserve` zu kennzeichnen, vgl. M31. All diese Kennzeichnungen können zur Steuerung von reduzierenden Transformationen verwendet werden, u.a. für das Markieren und Ausschneiden von Prozessfragmenten. Für die Unterstützung der Sprachemulation durch Chevron wurde außerdem die Annotation mit den Phasen `Preprocessing`, `Simu-`

`lationrun`, `Postprocessing` und `Visualization` beispielhaft umgesetzt. Benutzerdefinierte Phasen können ad hoc annotiert werden.

3. Dienstaufruf zum Ausschneiden. Der Dienstaufruf zum Ausschneiden von Prozessfragmenten ist aus dem Arbeitsbereich des Eclipse Workspace heraus ermöglicht worden. Der Dienst basiert auf der Implementierung der Sichtentransformationen von Streule [Str09]. Transformationsregeln werden als separate XML-Datei im Workspace hinterlegt. Auf Basis dieses Dienstes können die Prozessteile, die nicht explizit als zu bewahrend gekennzeichnet wurden (M13) und nicht länger verwendeter Kontext entfernt werden (M1, M8, M29). In der Implementierung ist der Dienst als Code in Eclipse integriert und lediglich die BPEL-Datei wird verarbeitet – das entsprechende WSDL, XML Schema und die Zusatzdatei der Modellierungsumgebung werden nicht transformiert. Im Rahmen von SimTech wurde von Hahn und Sonntag [SDH+12] später eine Ausschneidefunktion von SESE-Fragmenten fest in das Werkzeug integriert.

4. Bibliothekseinbindung. Die Einbindung der Bibliothek für Prozessfragmente wurde in Form eines Eclipse Plug-Ins implementiert [Den11]. Dadurch können die in der Bibliothek verfügbaren Inhalte lokal im Workspace bereitgestellt und dem Modellierer in einer Eclipse View über einen Baum zugänglich gemacht werden. Das Plug-In ist durch Aufruf der von Fragmento angebotenen Web Service Schnittstellen mit der Bibliothek lose gekoppelt. Es können mehrere Instanzen der Bibliothek genutzt werden. Dadurch kann ein Nutzer Prozessfragmente auf einfache Art und Weise mit anderen Nutzern austauschen. Die Verbindung dieser Bibliotheksintegration mit der Modellierungsoberfläche basiert auf einer Zusammenarbeit im Rahmen von SimTech [SDH+12]. In der Anpassung nach Hahn und Sonntag können Prozessfragmente aus der Bibliothekseinbindung als Bausteine in die Modellierungspalette geladen werden. Von dort aus können sie per Drag & Drop in den Prozess eingefügt werden. Der Drop eines Fragments im Prozessgraphen bewirkt den Aufruf eines Assistenten, der die Verbindung des Kontexts des Fragments mit dem Kontext des Prozesses steuert, Parameter vom Modellierer abfragt und schließlich das Fragment in den Prozess einfügt. Diese Lösung macht von den BPEL-Erweiterungen für rr-Fragmente (vgl. Abschnitt 9.3) keinen Gebrauch, sondern ist aufgrund der einfacheren Implementierbarkeit auf SESE-Fragmente beschränkt.

5. Darstellungsanpassungen. In der Erforschung von Sichten im Umfeld von Simulationsworkflows ist eine Darstellungsanpassung entstanden, die von Sonntag in der Zusammenarbeit [SGK+11] implementiert wurde: Ein Symbol zur Repräsentation der Semantik eines Dienstaufrufes wird als URL an eine WSDL-Operation angehängt und dieses Symbol dynamisch vom Modellierungswerkzeug geladen, sobald die Operation bei einem Dienstaufruf verwendet wird. Das geladene Symbol wird anstatt der sonst verwendeten Symbole angezeigt, mit denen die Aktivitätstypen `<invoke>`, `<receive>`, `<reply>` usw. unterschieden werden. Eine weitere Anpassung des Erscheinungsbildes von Aktivitäten wurde in der studentischen Arbeit [Zen10] prototypisch implementiert. Durch die Bindung der Größe, Farbe und Form von Aktivitäten an annotierte Informationen kann damit bspw. eine Hervorhebung von gekennzeichneten Aktivitäten bewirkt werden.

10.5 Monitoring-Plattform

Zunächst wird die Plattform für das Monitoring von Prozessinstanzen auf Musterbasis konzipiert. Eine rudimentäre Sicht zum Monitoring von Prozessinstanzen (vgl. Abschnitt 4.3.1) sieht zunächst eine Anreicherung mit Laufzeitinformationen (M15) vor. Auf Basis der daraus resultierenden Prozessinstanz können wenig relevante Strukturen wie bspw. tote Pfade ausgeblendet werden (M1), wobei der Prozess nicht in Fragmente zerfallen soll (M29). Die Darstellung kann nach einem Graphschema (M19) erfolgen. Das Erscheinungsbild von Aktivitäten gibt Auskunft über deren Ausführungszustand (M16), wobei verschiedene Vorlagen unterschiedliche Detailgrade ermöglichen. Diese Grundlage kann um verschiedene Aspekte erweitert werden, um ein größeres Spektrum an Informationsbedarfen abzudecken. So können neben Aktivitäten auch Pfade von Kontrollverbindungen in die Darstellungsanpassung einbezogen werden (M17) und die Aggregation verbundener Strukturen (M2) als weitere Abstraktionsmöglichkeit genutzt werden, was besonders bei Blockstrukturen wie bspw. `<sequence>` zur Geltung kommt. Aus der Betrachtung der Anforderungen aus dem Thema Compliance kann ein Prozess zusätzlich mit Informationen über die darin enthaltenen rr-Fragmente angereichert werden (M14). Diese angereicherte Information kann wiederum zur Ansteuerung der Hervorhebung von rr-Fragmenten (vgl. Abschnitt 7.1.3) oder für deren Ausblenden genutzt werden (vgl. Abschnitt 7.1.2). Durch diese Verfeinerungen werden Laufzeitaspekte von Prozessen mit Compliance verknüpft (A17). Auch kann der sprachübergreifende Ansatz einer Zustandsprojektion (S2) in einer Monitoring-Plattform ansetzen, d.h. an der Quelle der dazu erforderlichen Informationen. Die ausgewählten Sichtenentwurfsmuster können als Anforderungskatalog an die Plattform zum Monitoring von Prozessinstanzen verstanden werden, welche nachfolgend vorgestellt wird.

10.5.1 Architektur

Die Architektur der Plattform zum Monitoring von Prozessen ist in Abbildung 10.11 dargestellt. In dieser Architektur sind die Erkenntnisse zusammengeführt, die im Rahmen von studentischen Arbeiten [Lat10][Lie11], darauf aufbauenden technischen Berichten [SLL11] [SKL+11] und im wissenschaftlichen Diskurs zum sprachübergreifenden Monitoring von Prozessen gewonnen wurden [SLL+11]. Im Folgenden werden die Komponenten dieser Architektur kurz erläutert.

Die Architektur besteht aus fünf Komponenten: (1) der Prozessserver ist an sich eine Ausführungskomponente, dient in dieser Architektur jedoch als Datenquelle für Prozessmodelle und Instanzinformationen und ist daher grün eingefärbt. (2) Der Applikationsserver beinhaltet die Logik, um Prozessmodelle mit Instanzinformationen anzureichern und, sofern vom Nutzer angefordert, durch Sichtentransformationen strukturell anzupassen und anschließend für die Darstellung aufzubereiten. (3) Die Anbindung der Fragment-Bibliothek ermöglicht die statische Anreicherung von Prozessmodellen und deren Instanzen mit Informationen, wie bestimmte (Compliance-)Anforderungen in einem Prozess umgesetzt sind. (4) Im Rahmen des Informationsdesigns können abstrakte Prozessmodelle und Zustandsübertragungsregeln vorab modelliert und auf dem Applikationsserver hinterlegt werden, bspw. in Chevron

anhand der in Abschnitt 9.2.4 vorgestellten XML Artefakte. (5) Der Nutzer der Plattform kann bestehende Prozesse und Instanzen sowie für die Zustandsprojektion hinterlegte Prozesse und respektive Instanzen auswählen und sich diese grafisch in einem Webbrowser anzeigen lassen. Die Wahl einer webbasierten Realisierung erlaubt dies, ohne dafür besondere Komponenten beim Nutzer installieren zu müssen. Der Nutzer steuert auf Anfrage außerdem die Transformationen, die im Applikationsserver durchgeführt werden. Dadurch können verschiedene Interessengruppen unterstützt werden und diese ihren Informationsbedarf individuell konfigurieren.

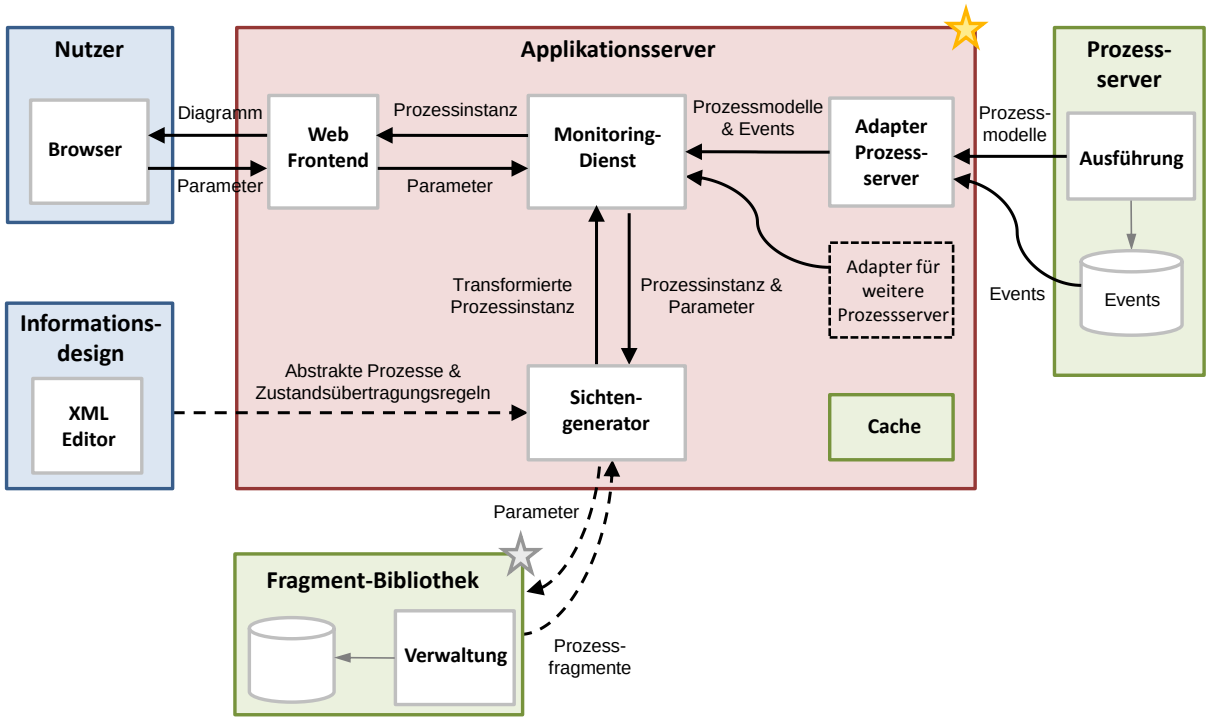


Abbildung 10.11: Architektur der Monitoring-Plattform für Prozesse, angepasst und erweitert nach [SLL11] und [SKL+11].

Der Applikationsserver beinhaltet verschiedene Unterkomponenten die miteinander interagieren. Der Prozessserveradapter stellt die Verbindung zu einem Prozessserver her. Ein Adapter nutzt herstellerspezifische Schnittstellen und ist auf einen konkreten Prozessserver zugeschnitten. Das Adapterkonzept erlaubt es, Verbindungen zu verschiedenen Prozessservern von unterschiedlichen Herstellern einzurichten. Entscheidend ist, dass die Prozessmodelle und die Events der Ausführung abgefragt werden können; diese Abfrage kann bspw. über den Aufruf einer Web Service Schnittstelle oder über SQL Abfragen auf einer aufgebauten Datenbankverbindung erfolgen. Prozessmodelle werden im Monitoring-Dienst in ein internes Datenformat überführt und können anschließend gecacht werden, da davon ausgegangen werden kann, dass sich die Prozessmodelle nur in größeren Abständen ändern. Der Monitoring-Dienst hat zudem die Funktion, Events zu Aktivitätszuständen zu aggregieren und das Prozessmodell mit diesen Zuständen anzureichern.

Geht vom Nutzer eine Anfrage ein, fragt der Monitoring-Dienst das angeforderte Prozessmodell vom Prozessserver ab, sofern es noch nicht im Cache zwischengespeichert ist oder der

Cache nicht aktuell ist. Das Prozessmodell wird anschließend mit aktuellen Zustandsinformationen angereichert. Wurde vom Nutzer eine bestimmte Sichtenfunktion ausgewählt, wie bspw. ein Ausblenden der Aktivitäten vom Typ `<assign>`, so werden entsprechende Routinen des Sichtengenerators mit den vorgegebenen Parametern aufgerufen. Der Sichtengenerator führt strukturelle Änderungen am internen Modell der Prozessinstanz durch und reichert diese mit weiteren Informationen an, z.B. mit einem Flag zur Steuerung, ob eine Aktivität hervorgehoben werden soll und welche Farbe zur Hervorhebung eingesetzt werden soll. Bevor die transformierte Prozessinstanz dargestellt werden kann, muss diese vom internen Modell in ein Darstellungsformat wie SVG gebracht werden, das in die Nutzeroberfläche eingebettet werden kann.

Einschränkungen. Die Architektur sieht die Integration mit der Fragment-Bibliothek konzeptionell vor, eine entsprechende Implementierung wurde im Rahmen der vorliegenden Dissertation allerdings nicht vorgenommen. Des Weiteren wurden für die Zustandsübertragung von BPEL zu Chevron lediglich die Voraussetzungen geschaffen (vgl. Abschnitt 9.2). Die Integration der dienstbasierten Realisierung von Sichtenentwürfen (vgl. Abschnitt 9.1) ist in dieser Architektur aus mehreren Gründen nicht vorgesehen. Bspw. kann durch die Nutzung eines internen Datenmodells für Prozessinstanzen und speziell darauf zugeschnittenen Sichtenfunktionen eine hohe Performance und sehr geringe Antwortzeiten in der Aktualisierung erzielt werden, da weder Dienstaufrufe noch Marshalling erforderlich sind. Des Weiteren besteht eine geringe Notwendigkeit für weitere Transformationsdienste, wenn die integrierten Funktionen die geforderten Muster bereits abdecken.

10.5.2 Implementierung

Die Monitoring-Plattform wurde in der Diplomarbeit von Latuske implementiert [Lat10]. Eine Portierung dieser Plattform auf einen anderen Applikations- und Prozessserver wurde in der Studienarbeit von Krein [Kre11] durchgeführt. Die nachfolgende Beschreibung der Implementierung basiert auf diesen Arbeiten und dem technischen Bericht von Schumm et al. [SLL11], der diese Plattform in einen größeren Kontext stellt.

Technologien. Die initial entwickelte Plattform verwendet einen Tomcat Application Server als Basis und beinhaltet einen Adapter zur Integration mit der BPEL Engine Apache ODE. Die Plattform ist als webbasierte Anwendung ausgelegt und kann mit einem SVG-fähigen Webbrowser genutzt werden. Asynchrone Aufrufe des Anwendungsservers unter Einsatz von XML (AJAX) erlauben die reibungslose Interaktion zwischen Nutzer und Server. Die Entscheidung zur Nutzung von SVG für die Darstellung und Java für die Implementierung der Unterkomponenten ist von deren erfolgreichem Einsatz in dem Werkzeug Oracle Java CAPS [Ora10] beeinflusst. Dieses Werkzeug stellt zudem eine Demonstration der Machbarkeit einer solchen Plattform dar. Aufgrund von Einschränkungen in der Unterstützung von BPEL betreffend `<flow>` wurde dieses quelloffene Werkzeug allerdings nicht als Codebasis verwendet. Die Portierung der Monitoring-Plattform setzt den WSO2 Business Process Server [WSO12] ein. Das Layout des BPEL-Prozesses wurde mit einer Orientierung von oben nach unten (M24) umgesetzt, wobei sowohl das graphbasierte Element `<flow>` (M19) als auch die Blockstrukturen von BPEL (M21) im Layout berücksichtigt wurden. Um diese

Form des Layouts zu realisieren, wurde ein einfach zu implementierender Algorithmus zum Layout von BPMN-Prozessen [KKL+09] auf BPEL angepasst und erweitert, vgl. [Lat10].

Funktionalität. Nachdem der Nutzer ein Prozessmodell und eine Instanz dieses Modells ausgewählt hat, wird diese grafisch dargestellt, wie in Abbildung 10.12 beispielhaft gezeigt.

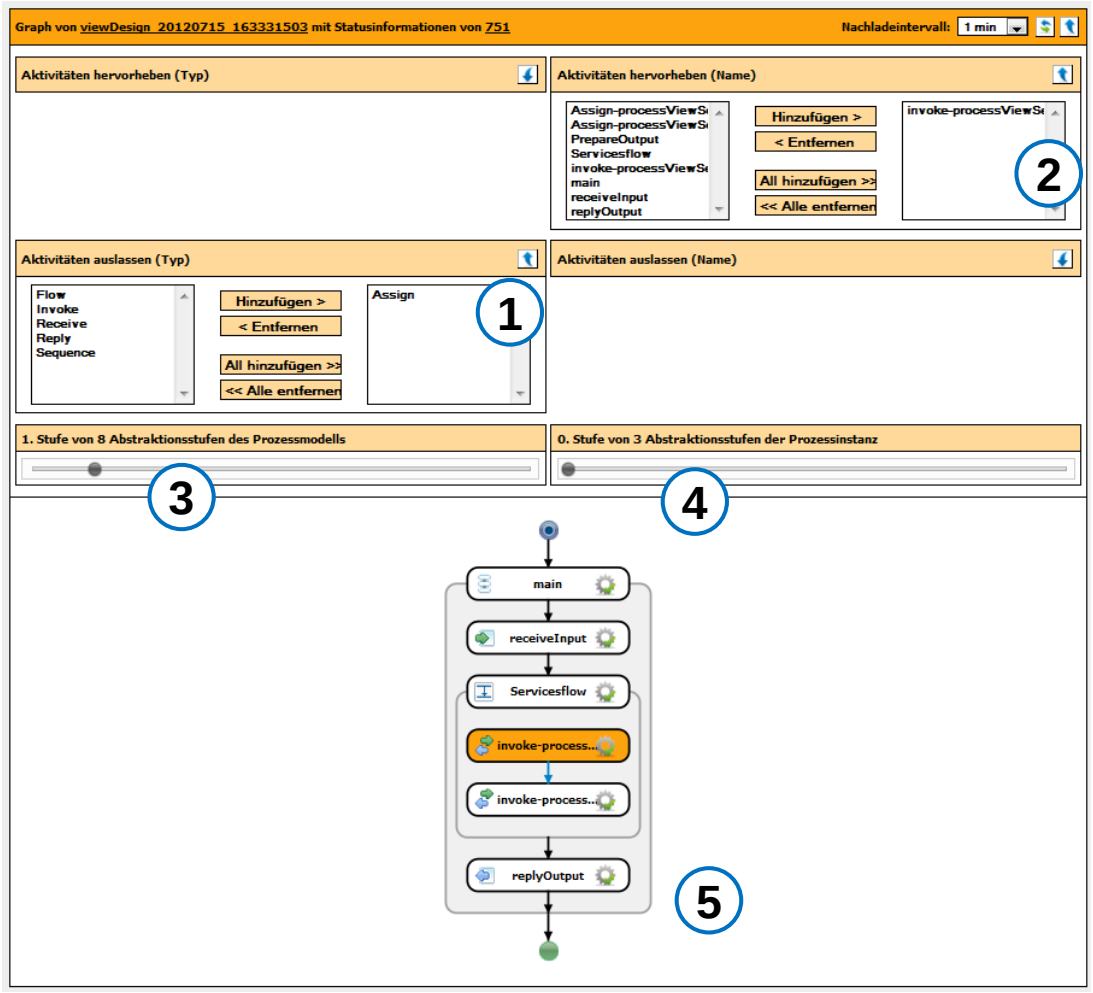


Abbildung 10.12: Screenshot einer Sicht auf eine Prozessinstanz in der Monitoring-Plattform in der Implementierung nach [Lat10].

Die in diesem Beispiel dargestellte Prozessinstanz ist eine Dienstkomposition, die mit dem in Abschnitt 10.1 vorgestellten Framework generiert wurde. Durch die Nutzung skalierbarer Vektorgrafik ist optisches Zoomen, d.h. eine Bildvergrößerung, im Webbrowser ohne Qualitätsverlust möglich. Was in der Literatur zum Teil als semantischer Zoom bezeichnet wird [Bob08], ist in der Implementierung durch mehrere Möglichkeiten gegeben. Dem Nutzer stehen vier Funktionen zur Verfügung, mit der die Sicht auf die Prozessinstanz konfiguriert werden kann. Diese Funktionen umfassen (1) das Ausblenden von Aktivitäten, (2) das Hervorheben von Aktivitäten, (3) die Wahl von Abstraktionsstufen auf Basis von Modellinformationen und (4) die Wahl von Abstraktionsstufen auf Basis von Instanzinformationen. Das Ausblenden und das Hervorheben können in der Implementierung entweder durch Auswahl eines oder mehrerer Aktivitätstypen wie z.B. `<assign>` bzw. `<invoke>` oder durch Auswahl konkreter Aktivitäten nach deren Name gesteuert werden (5).

Durch diese Funktionen werden die Sichten zum Ausblenden (vgl. Abschnitt 7.1.2) und zum Hervorheben (vgl. Abschnitt 7.1.3) von rr-Fragmenten prototypisch realisiert.

Der Einsatz eines Schiebereglers (engl. *slider*) zur stufenweisen Regelung der Abstraktion eines Prozessmodells wurde bereits von Polyvyanyy et al. [PSW08] diskutiert. In der entwickelten Monitoring-Plattform wird ebenfalls ein Modellschieberegler eingesetzt (3), der allerdings in seiner Wirkung auf die Eigenschaften der Prozesssprache BPEL zugeschnitten ist und damit eine andere Semantik besitzt. Zusätzlich zur Nutzung von Modellinformationen ist durch einen zweiten Schieberegler die Möglichkeit gegeben auch Instanzinformationen zur Abstraktion einzusetzen (4).

Eine Neuerung gegenüber anderen Werkzeugen besteht darin, dass die beiden Regler miteinander kombiniert werden können. Nachfolgenden wird die Semantik der beiden Regler erläutert. Der Modellabstraktionsregeler umfasst fünf fest definierte Stufen und weitere Stufen entsprechend der Tiefe an Verschachtelungen des Prozessmodells. In der ersten Stufe wird das Erscheinungsbild von Aktivitäten von einem Detailmodus in einen Kompaktmodus geändert, in dem lediglich der Aktivitätstyp als Symbol, der Aktivitätsname und der Aktivitätszustand als Symbol dargestellt werden. Der Detailmodus umfasst darüber hinaus auch die Start- und Stoppzeit der Ausführung. In der zweiten Stufe werden alle Aktivitäten vom Typ `<assign>` und `<empty>` ausgeblendet. In der dritten Stufe werden außerdem Aktivitäten vom Typ `<throw>`, `<rethrow>` und `<validate>` ausgeblendet. In der vierten Stufe werden Handler zur Kompensation und zur Terminierung aggregiert; Handler zur Ereignis- und Ausnahmebehandlung werden in der fünften Stufe aggregiert. In den weiteren Stufen werden am tiefsten verschachtelte Aktivitäten aggregiert.

Der Regler zur Abstraktion unter Einsatz von Instanzinformationen umfasst drei fest definierte Stufen. Die erste Stufe hebt den Ausführungspfad und alle darin enthaltenen Aktivitäten hervor. In der zweiten Stufe wird diese Hervorhebung durch eine Heatmap entsprechend der Aktivitätslaufzeit ersetzt. In der dritten Stufe werden außerdem tote Pfade ausgeblendet, vgl. dazu auch die Sicht zur vereinfachten Prozessinstanz in Abschnitt 4.3.1.

Der kombinierbare Einsatz der Schieberegler und ein explizites Ausblenden und Hervorheben von ausgewählten Aktivitäten implementiert die in Abbildung 10.13 dargestellte Mustersequenz. Dieser Sequenz ist zu entnehmen, dass ein Ausblenden Vorrang vor der Aggregation und vor dem Hervorheben hat. Hervorzuhebende Aktivitäten werden zudem vor dem Ausblenden nicht bewahrt. Ein markanter Unterschied zu der Patentanmeldung von Guo et al. [GHN+09] besteht neben der Nutzung grafischer Vorlagen und den Möglichkeiten zur Hervorhebung darin, Laufzeitinformationen in die Abstraktionsfunktionen einzubeziehen.

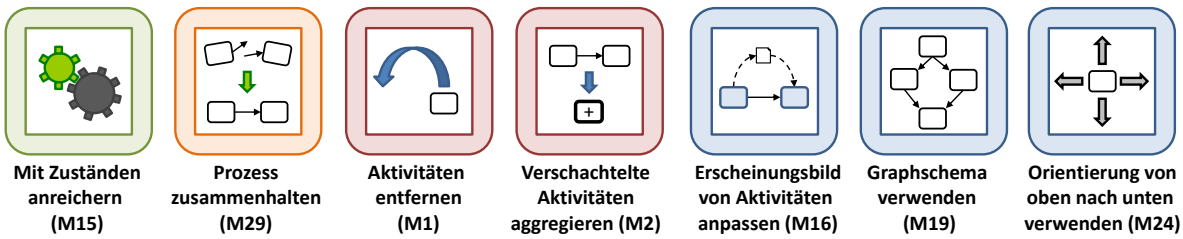


Abbildung 10.13: Effektive Mustersequenz für Sichten in der Monitoring-Plattform.

Teil V

Abschluss

Überblick zu Teil V

In Teil V werden die praktische und theoretische Relevanz von zentralen Aspekten kritisch reflektiert und die Ergebnisse der vorliegenden Dissertation zusammengefasst.

In Kapitel 11 wird die vorgenommene Validierung beschrieben. Den Schwerpunkt der Validierung bildet der musterbasierte Sichtenentwurf, der im Rahmen von Workshops mit Vertretern aus Industrie und Forschung untersucht wurde. Anschließend wird die praktische Anwendbarkeit des Ansatzes der rr-Fragmente diskutiert.

In Kapitel 12 werden die erzielten Ergebnisse zusammengefasst und die Erkenntnisse aus den entwickelten Ansätzen herausgestellt. Deren Stärken und Schwächen werden identifiziert und deren Möglichkeiten und Grenzen aufgezeigt.

In Kapitel 13 wird ein Ausblick auf eine mögliche weiterführende Forschung gegeben.

KAPITEL 11

VALIDIERUNG

Im Rahmen der Validierung werden die praktische Anwendbarkeit und die Eignung der entwickelten Ansätze untersucht. Dazu werden zwei Fallstudien aus dem Forschungsprojekt COMPAS herangezogen. Eine Fallstudie betrifft das Bankwesen (Kreditvergabe), die andere den Telekommunikationsbereich (Media Streaming). Die Validierung des musterbasierten Sichtenentwurfs fokussiert auf die Fallstudie im Bankwesen und basiert auf Workshops, die mit Probanden aus Industrie und Forschung durchgeführt wurden. Der Ansatz der rr-Fragmente wurde im Rahmen von COMPAS auf beide Fallstudien angewendet.

11.1 Validierung des musterbasierten Sichtenentwurfs

In der Zusammenfassung der Zielstellung der vorliegenden Dissertation (vgl. Abschnitt 2.4.5) wurden verschiedene Anforderungen an den musterbasierten Sichtenentwurf formuliert und in den entwickelten Ansätzen berücksichtigt. Diese Anforderungen beinhalten den Einsatz einer einheitlichen Beschreibungsform (A6, A7), eine Berücksichtigung der organisatorischen Dimension des Sichtenentwurfs (A8), Möglichkeiten zur Wiederverwendung von Sichten (A9), die Berücksichtigung bestehender Ansätze zum Einsatz grafischer Darstellung (A11) und das Monitoring von Prozessen mit Sichten (A12). In diesem Abschnitt wird eine eingeschränkte und initiale Validierung des musterbasierten Sichtenentwurfs vorgestellt, um die praktische Einsetzbarkeit des musterbasierten Sichtenentwurfs zu überprüfen (A10).

Eine Voraussetzung für die Überprüfung der praktischen Einsetzbarkeit war, dass der Ansatz für die Probanden in einer verständlichen Form vermittelt wurde. Die Vorgehensweise, die zur Erläuterung des musterbasierten Sichtenentwurfs eingesetzt wurde, ist im Anhang in Abschnitt 14.3 dargestellt. Im Anschluss an die Erläuterung wurde den Probanden ein Fragebogen zur Erfassung der Wahrnehmung vorgelegt, der sich an der Validierungstechnik der Benutzbarkeit in verwandten Arbeiten orientiert: die Validierung der von La Rosa et al. [RHW+11] vorgeschlagenen Muster stützt sich auf den Ansatz von Davis [Dav89], welcher zwei Konstrukte vorschlägt: wahrgenommene Nützlichkeit (engl. *perceived usefulness*) und wahrgenommene Einfachheit der Nutzung (engl. *perceived ease of use*). Die Erfassung dieser Konstrukte erlauben nach Davis einen Rückschluss auf die Nutzerakzeptanz (engl. *user acceptance*). Davis schlägt eine Skala von 1 bis 7 vor, wobei 1 für volle Zustimmung und 7 für volle Ablehnung stehen. Zur Validierung des Ansatzes werden die Konstrukte der wahrgenommenen Nützlichkeit, die Einfachheit der Nutzung und die Gestaltung der Antwortskala

übernommen. Diese Konstrukte werden nicht isoliert für einzelne Muster erfasst, sondern der musterbasierte Sichtenentwurf wird als Ganzes betrachtet. Zudem wird explizit nach der antizipierten Einsetzbarkeit gefragt. Ferner werden offene Fragen gestellt, um wahrgenommene Probleme, Risiken beim Einsatz und konstruktive Vorschläge qualitativ zu erheben.

Der verwendete Fragenkatalog umfasst fünf Fragen zur Wahrnehmung des Probanden:

1. Zur Nützlichkeit: *Können Sie sich vorstellen, dass durch den Einsatz der Mustersprache die Komplexität von Geschäftsprozessen reduziert werden kann?* Antwortskala: 1 = „sehr gut vorstellbar“, 7 = „sehr schwer vorstellbar“.
2. Zur Einfachheit der Nutzung: *Wie schwierig empfinden Sie die Nutzung der Mustersprache zum Entwurf von Sichten auf Geschäftsprozesse?* Antwortskala: 1 = „sehr einfach“, 7 = „sehr schwer“.
3. Zur antizipierten Einsetzbarkeit: *Können Sie sich vorstellen, die Mustersprache zu verwenden, um Sichten auf Geschäftsprozesse zu entwerfen?* Antwortskala: 1 = „sehr gut vorstellbar“, 7 = „sehr schwer vorstellbar“.
4. Zur qualitativen Erhebung: *Welche Risiken und Probleme sehen Sie bei der Nutzung des musterbasierten Sichtenentwurfs?*
5. Zur qualitativen Erhebung: *Weitere Anmerkungen oder Verbesserungsvorschläge?*

An der Validierung haben sich insgesamt 25 Probanden beteiligt; zwölf Probanden kommen aus der Industrie und sind in den Bereichen Unternehmensberatung, IT-Consulting, Business Process Management, Softwareentwicklung, IT-Management oder Produktentwicklung tätig. Diese zwölf Probanden verteilen sich wiederum auf vier verschiedene Unternehmen. Weitere dreizehn Probanden sind in der Forschung am Institut für Architektur von Anwendungssystemen (IAAS) der Universität Stuttgart in den Themengebieten Cloud, Geschäftsprozesse und Mustersprachen tätig.

Insgesamt wurden sieben Workshops mit einer durchschnittlichen Dauer von zwei Stunden durchgeführt. Der Ansatz wurde anhand einer Auswahl an häufig in Sichtenentwürfen verwendeten Mustern und deren Komposition vorgestellt und diskutiert. Eine detaillierte Vorstellung, beispielhafte Verwendung und umfassende Diskussion jedes einzelnen Musters des Musterkatalogs und aller erstellten Sichtenentwürfe hätten den möglichen Rahmen der Workshops gesprengt.

Diskussionsgrundlage für die Validierung war die Fallstudie aus dem Bankwesen, welche sich auf einen Kreditvergabeprozess mit etwa 30 Aktivitäten bezieht, der von einem Industriepartner aus COMPAS zur Verfügung gestellt wurde, vgl. Abbildung 11.1. Dieser Kreditvergabeprozess war im Graphschema (M19) dargestellt, in einer Orientierung von links nach rechts (M24) unter Berücksichtigung der Organisation (M26). Die Positionierung der Aktivitäten war manuell erfolgt (M28). Es bestanden keine Anpassungen am Erscheinungsbild, diese wurden erst im Verlauf der Musterdiskussion thematisiert. Dadurch wurde den Probanden als Ausgangsbasis bereits eine grafische Sicht auf den Prozess bereitgestellt. Dieser Vorgehensweise liegt die Annahme zugrunde, dass das Prozessmodell ohne grafische Darstellung zu schwer verständlich wäre.

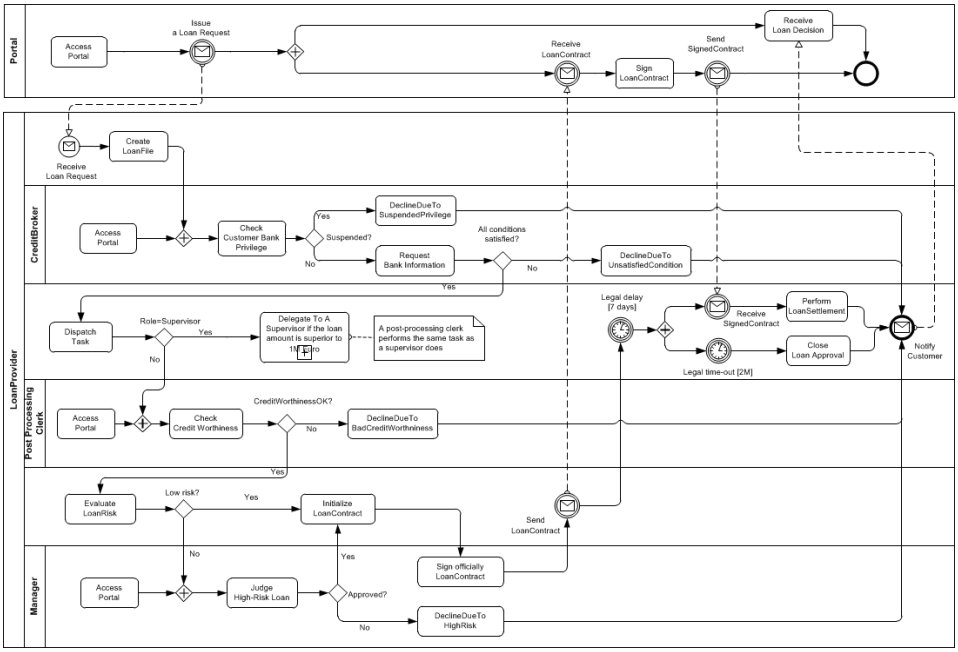


Abbildung 11.1: Kreditvergabeprozessmodell aus COMPAS als Beispielszenario zur Validierung.

Abbildung 11.2 zeigt eine Fotodokumentation von dem diskutierten Beispielszenario. Die Mustersymbole wurden ausgeschnitten in Papierform bereitgestellt und konnten somit auf den Prozess gelegt und deren Wirkung mit Filzstift skizziert und veranschaulicht werden.

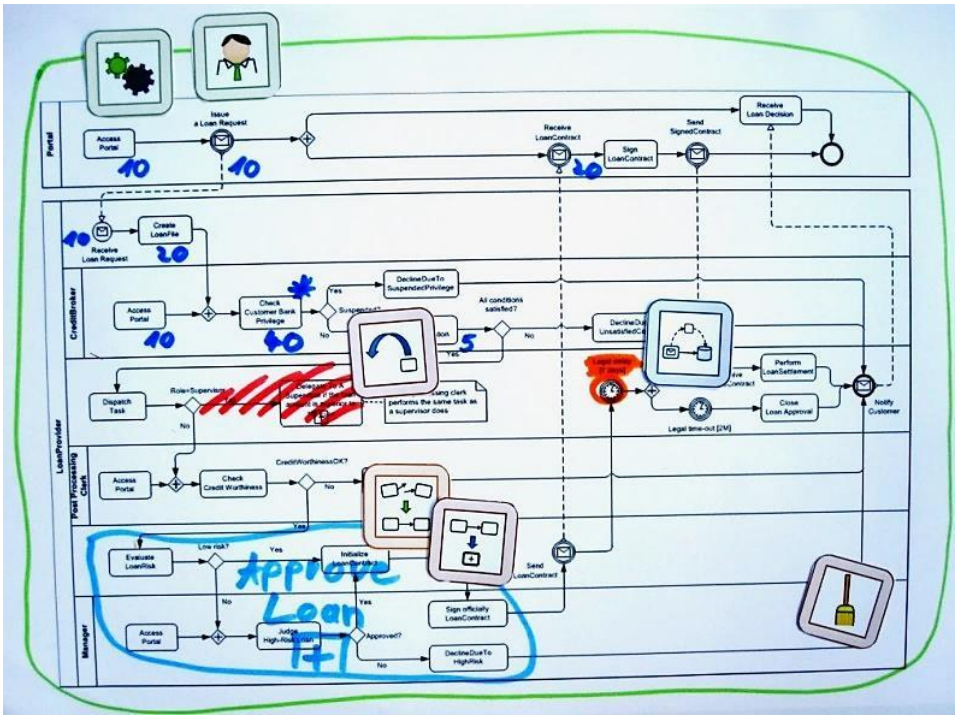


Abbildung 11.2: Diskussion der Effekte und Einsatzmöglichkeiten der Sichtenentwurfsmuster am Beispielszenario im Rahmen der Validierung.

Quantitative Ergebnisse. Eine anonymisierte Aufstellung der quantitativen Rückmeldungen ist in Tabelle 11.1 dargestellt. Diese Aufstellung umfasst die Fragen 1-3. Zur Erinnerung die Werte: 1 = „sehr gut vorstellbar / sehr einfach“, 7 = „sehr schwer vorstellbar / sehr schwer“.

Tabelle 11.1: Aufstellung der Rückmeldungen der befragten Teilnehmer.

Quelle	Forschung													Firma 1			Firma 2				Firma 3		Firma 4		
Proband	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Frage 1	1	1	1	1	1	1	2	1	1	1	1	1	1	1	1	2	2	2	1	2	2	1	6	3	2
Frage 2	2	2	3	2	1	-	3	2	2	2	2	1	2	1	2	3	5	2	3	3	3	4	5	5	5
Frage 3	1	2	2	3	2	1	2	1	1	2	1	1	1	1	1	5	3	1	1	2	1	1	3	7	3

Auf Basis der erhobenen Rückmeldungen wurden verschiedene Werte bestimmt, die in Tabelle 11.2 aufgelistet sind. Die Auswertung umfasst den Mittelwert, den Median, den Maximalwert und den Minimalwert. Die Rückmeldungen von Forschung und Industrie wurden sowohl separat als auch zusammen ausgewertet. Die Differenz stellt die unterschiedliche Sichtweise von Forschung und Industrie heraus.

Tabelle 11.2: Auswertung der Rückmeldungen mit Differenzierung von Forschung und Industrie.

Quelle	Forschung				Industrie				Gesamt				Differenz			
Wert	Mittelwert	Median	Minimum	Maximum	Mittelwert	Median	Minimum	Maximum	Mittelwert	Median	Minimum	Maximum	Mittelwert	Median	Minimum	Maximum
Frage 1 (Nützlichkeit)	1,1	1	1	2	2,1	2	1	6	1,6	1	1	6	1	1	0	4
Frage 2 (Einfachheit der Nutzung)	2,0	2	1	3	3,4	3	1	5	2,7	2	1	5	1,4	1	0	2
Frage 3 (Antizipierte Einsetzbarkeit)	1,5	1	1	3	2,4	2	1	7	2,0	1	1	7	0,9	1	0	4

Die Auswertung der Rückmeldungen lässt folgende Interpretation zu:

1. Nützlichkeit: Nach Einschätzung der befragten Probanden kann durch den Einsatz der Mustersprache die Komplexität von Geschäftsprozessen reduziert werden.
2. Einfachheit der Nutzung: Bevor die Mustersprache zum Entwurf von Sichten auf Geschäftsprozesse eingesetzt werden kann, müssen umfassendere Kenntnisse über deren Nutzung und die Auswirkungen des Einsatzes einzelner Muster vermittelt werden, d.h. dem Einsatz der Sprache muss ein Training vorangehen.
3. Antizipierte Einsetzbarkeit: Etwa die Hälfte der Industrievertreter kann sich sehr gut vorstellen, Sichten auf Geschäftsprozesse auf Basis der Mustersprache zu entwerfen.
4. Industrievertreter betrachten den Ansatz kritischer als Forschungsvertreter.

Qualitative Rückmeldungen. Die qualitativen Rückmeldungen wurden thematisch sortiert und konsolidiert. Zusammenfassend wurden in den Workshops folgende *Risiken und Probleme* von den Probanden festgestellt:

1. *Einstiegshürde:* Der Ansatz wurde teilweise als sehr abstrakt und nicht einfach verständlich empfunden. Eine gewisse Einlernzeit wurde als erforderlich angesehen. Gründe dafür waren die Anforderungen an das Vorstellungsvermögen sowie die hohe Anzahl an Mustern. Es wurde angemerkt, dass das Ergebnis für den Endnutzer eventuell nur schwer vorstellbar ist, wenn viele Muster angewendet werden.
2. *Werkzeugunterstützung:* Es wurde mehrfach als problematisch betrachtet, dass bei den Mustern ein nicht unerheblicher Interpretationsspielraum besteht und das Resultat der Anwendung eines Musters entsprechend vielfältig ausfallen kann, bspw. in der Abgrenzung von dem temporären Ausblenden und dem dauerhaften Entfernen von Strukturen. Eine einheitliche Funktionsweise der Muster wurde mehrfach gefordert; außerdem sollte sichergestellt werden können, dass ein Muster korrekt implementiert wurde. Ferner wurde eine durchgängige Toolunterstützung als Voraussetzung genannt, damit der Ansatz in der Praxis eingesetzt werden kann. Neben dem Grobentwurf mit Mustersymbolen sollte die Werkzeugkette auch einen tabellarischen Feinentwurf mit entsprechender Musterparametrierung unterstützen.
3. *Nutzungsschwierigkeiten:* Mehrere Probanden hatten Schwierigkeiten beim Festlegen einer Reihenfolge der Muster beim Entwurf einer Sicht. In der Diskussion hat sich gezeigt, dass mögliche Parameter der Muster und eine explizite Beschreibung von Vorbedingungen und Nachbedingungen der Musteranwendung einem Nutzer helfen, eine Sequenz zu bilden. Es wurde festgehalten, dass diese Schwierigkeit ggf. vom eingesetzten Werkzeug abhängt und wie dieses Werkzeug den Nutzer bei der Musterzusammensetzung unterstützt, etwa durch Best Practice. Es wurde erwähnt, dass der Sichtenentwurf von einem Experten durchgeführt werden sollte, der die Details des betreffenden Geschäftsprozesses kennt, für den eine Sicht bestimmt ist.
4. *Abstraktion:* Es wurde mehrfach die Befürchtung geäußert, dass eine vorgenommene Abstraktion zu stark ist und dadurch zu viel aus einem Prozess entfernt wird; ein Reinzoomen solle auch nach einer Abstraktion weiterhin möglich sein. Es wurde zudem die Einschätzung geäußert, dass die Bezeichnung aggregierter Strukturen nur unzureichend durch einen Automatismus unterstützt werden kann und daher auch in Zukunft eine vom Nutzer durchzuführende Aufgabe bleiben wird, da Missverständnisse und daraus resultierende Fehlentscheidungen zu große Risiken darstellen.
5. *Zentrales Prozessmodell:* Nach Angaben der Industrievertreter gibt es selten zentrale Modelle. Stattdessen modelliert heute typischerweise jeder seine Realität, d.h. es gibt mitunter mehrere Modelle von demselben Prozess aus unterschiedlicher Perspektive. Möglicherweise kann diesem Umstand durch eine Sichten unterstützende Werkzeugkette begegnet werden. Ggf. könnten die verschiedenen Modelle manuell zu einem konsistenten Modell zusammengeführt und die ursprünglichen Perspektiven als Sichten bereitgestellt werden. Die Beschreibung einer Perspektive könnte an dieser Stelle durch einen musterbasierten Sichtenentwurf erfolgen. Weidmann et al. [WAK+11] sowie Küster et al. [KVF+13] verfolgen einen alternativen Ansatz, indem

sie vorschlagen, verschiedene Modelle (d.h. Perspektiven) aufeinander abzubilden und Änderungen an einem der Modelle mit dem jeweils anderen Modell zu synchronisieren. Dieser Ansatz kann auf Modelle unterschiedlicher Abstraktionsebenen wie auch auf fachliche und technische Modelle angewendet werden.

6. *Semantik einer Sicht:* Eine Sicht ändert die Semantik des Prozesses. Automatisch erstellte Sichten können daher unter bestimmten Voraussetzungen etwas Falsches darstellen und demzufolge Fehlentscheidungen bewirken.

Die Probanden haben folgende *Anmerkungen und Vorschläge* gemacht:

1. *Einstieg erleichtern:* Es könnte eine Art Cheat Sheet mit anschaulichen Beispielen bereitgestellt werden, um dem Nutzer als Kurzreferenz zu dienen. Des Weiteren wären Guidelines für die Anwendung der Muster hilfreich, um den Nutzer beim Erlernen der Mustersprache zu unterstützen. Auch wäre ein Empfehlungssystem vorstellbar, das dem Nutzer zu einem Entwurf passende Muster empfiehlt. Es wurde außerdem vorgeschlagen, die Muster an einem Prozess zu erklären den der Proband gut kennt und mit dem er auch konkrete Fragestellungen verbindet. Um den Ansatz zu veranschaulichen, wurde der Einstieg anhand eines konkreten Tools oder einer durchgängigen Werkzeugkette empfohlen.

2. *Nutzerkreis einschränken:* Die Mustersprache erfordert eine gewisse Expertise und den Aufbau von Fachkenntnissen. Eine Lösungsmöglichkeit wurde darin gesehen, einen kleinen Baukasten für den Einstieg vorzusehen. Es wurde auch erwähnt, dass die Mustersprache nicht in Fachbereichen Anwendung finden würde, sondern ein Expertenwerkzeug von Beratern, Architekten oder Entwicklern sei und diese die zur Nutzung erforderlichen Kenntnisse aufbauen müssten; Fachbereiche würden lediglich vordefinierte Sichten nutzen.

3. *Standardisierung anstreben:* Von Industrievertretern wurde häufig angemerkt, dass der bestehende Mangel an Standardisierung der Wirkungsweise der Transformationen als auch der Mangel an standardisierten Sichten und deren Unterstützung durch Hersteller ein Problem für die Akzeptanz des Ansatzes darstellt. Es wurde vorgeschlagen, eine Standardisierung einer geringen Anzahl relevanter Sichten anzustreben. Als Vergleich dazu wurde neben Muster von Sichten der Standard ISO/IEC/IEEE 42010 [ISO11] genannt, in dem Viewpoints im EAM-Umfeld definiert sind. Dieser ISO-Standard schlägt vor, zu einem Viewpoint relevante Interessen, die Interessengruppen, die Arten der Modelle, die verwendete Notation, mögliche Analysemethoden und Modellierungsoperationen anzugeben.

   Weitere Anmerkungen:
   - Die Möglichkeiten zustandsbasierter Transformation für den Einsatz im Monitoring wurden von Industrievertretern als vielversprechend bewertet.
   - Für den Nutzer stellt sich die Frage wie man von einem Ziel, das man bei einer Sicht verfolgt, zu den richtigen Mustern und deren Kombination kommt.
   - Die Mustersymbole wurden zum Teil als nicht selbsterklärend empfunden.

- Für Nutzer aus dem IT-Umfeld sollte eine Bezugnahme zum Ansatz der Design Patterns in der OO-Modellierung/-Programmierung erfolgen.
- Zu den gestellten Fragen wurden verschiedene Anmerkungen gemacht:
  - Zu Frage 1: Nicht durch den Einsatz der Mustersprache direkt kann die Komplexität von Geschäftsprozessen reduziert werden, sondern durch die Nutzung/Anwendung der Prinzipien.
  - Zu Frage 2: Die Nutzungsschwierigkeit hängt auch vom Tooling ab.
  - Zu allen Fragen: Die Antworten beziehen sich auf die Annahme, dass ein Tool diese Methode unterstützt.

Fazit. Der musterbasierte Sichtenentwurf kann zusammenfassend als Methode betrachtet werden, die sich zur Reduzierung der Komplexität von Geschäftsprozessen eignet, sofern eine entsprechende Werkzeugkette bereitgestellt wird. Allerdings sollte die Komplexität des Musterkatalogs nach Angaben der Probanden reduziert werden, ggf. könnte zwischen relevanten und weniger relevanten Mustern unterschieden werden. Die Industrievertreter waren sich darin einig, dass die Mustersprache als Expertensprache für Entwickler von Modellierungs- und Analysewerkzeugen oder auch für Berater und IT-Consultants zu betrachten ist und für Fachbereiche kaum geeignet ist. Diese Rückmeldung gibt Aufschluss über die Verortung der eingeführten Aufgabe des Informationsdesigns. Die Diskussionen haben aber auch offenbart, dass für die Praxis eine überschaubare Menge an Sichten für die gängigsten Fragestellungen wichtiger ist als die Verfügbarkeit einer generischen Sprache für den Sichtenentwurf – diese Sichten sollten in ihrer Funktionsweise standardisiert und in den gängigen (Enterprise-/Office-)Werkzeugen nutzbar sein. Probanden der Industrie zufolge macht eine volle Kombinierbarkeit der verschiedenen Muster den Ansatz zu komplex.

11.2 Validierung des Konzepts der rr-Fragmente

Das Konzept der rr-Fragmente wurde im Rahmen des Forschungsprojekts COMPAS in zwei Fallstudien validiert, die von Industriepartnern aus COMPAS bereitgestellt wurden. Die entsprechenden Prozesse wurden zunächst nach BPMN modelliert und in einer Phase der technischen Implementierung von Projektpartnern in ausführbare BPEL-Modelle umgesetzt. Sowohl in den BPMN-Repräsentationen als auch in den entsprechenden BPEL-Modellen waren rr-Fragmente zur Regelrealisierung enthalten. In dem Kreditvergabeprozess waren bspw. ein rr-Fragment zur Genehmigung von Krediten zur Berücksichtigung einer unternehmensinternen Richtlinie sowie ein rr-Fragment zur Einhaltung einer gesetzlich vorgeschriebenen Wartezeit vor einem Kreditabschluss enthalten. In dem Prozess aus dem Telekommunikationsbereich war u.a. ein rr-Fragment zur Implementierung einer lizenzrechtlichen Vorgabe enthalten, vgl. dazu „time-based plan“ im technischen Bericht von Görlach et al. [GKL+11].

Neben der Bibliothek für Prozessfragmente (vgl. Abschnitt 10.3) wurde auch die Monitoring-Plattform (vgl. Abschnitt 10.5) in die Architektur im Projekt COMPAS integriert und konnte so in beiden Fallstudien eingesetzt werden [COM10c]. Unabhängig von der Nutzung im

Kontext Compliance konnten SESE-Prozessfragmente im Projekt SimTech zur Beschleunigung der Modellierung von Workflows zur Simulationssteuerung beitragen [SDH+12].

Abbildung 11.3 zeigt einen Ausschnitt aus einer Sicht auf eine Prozessinstanz von der Fallstudie aus dem Telekommunikationsbereich. In der dargestellten Sicht wird der Zustand von Aktivitäten in einem Kompaktmodus angezeigt. Zusätzlich sind Aktivitäten von rr-Fragmenten, die in den Prozess integriert sind, hervorgehoben.

Der Ansatz der rr-Fragmente zur Unterstützung von Compliance wurde sowohl von den Industriepartnern in COMPAS als auch von den Projektgutachtern der Europäischen Kommission [EC11] als relevant und vielversprechend eingeschätzt. Deutlich wurde jedoch auch, dass der Ansatz der rr-Fragmente nur einer von vielen Technologiebausteinen ist, die man benötigt, um der Heterogenität der Anforderungen im Themenkomplex Compliance gerecht zu werden [COM11].

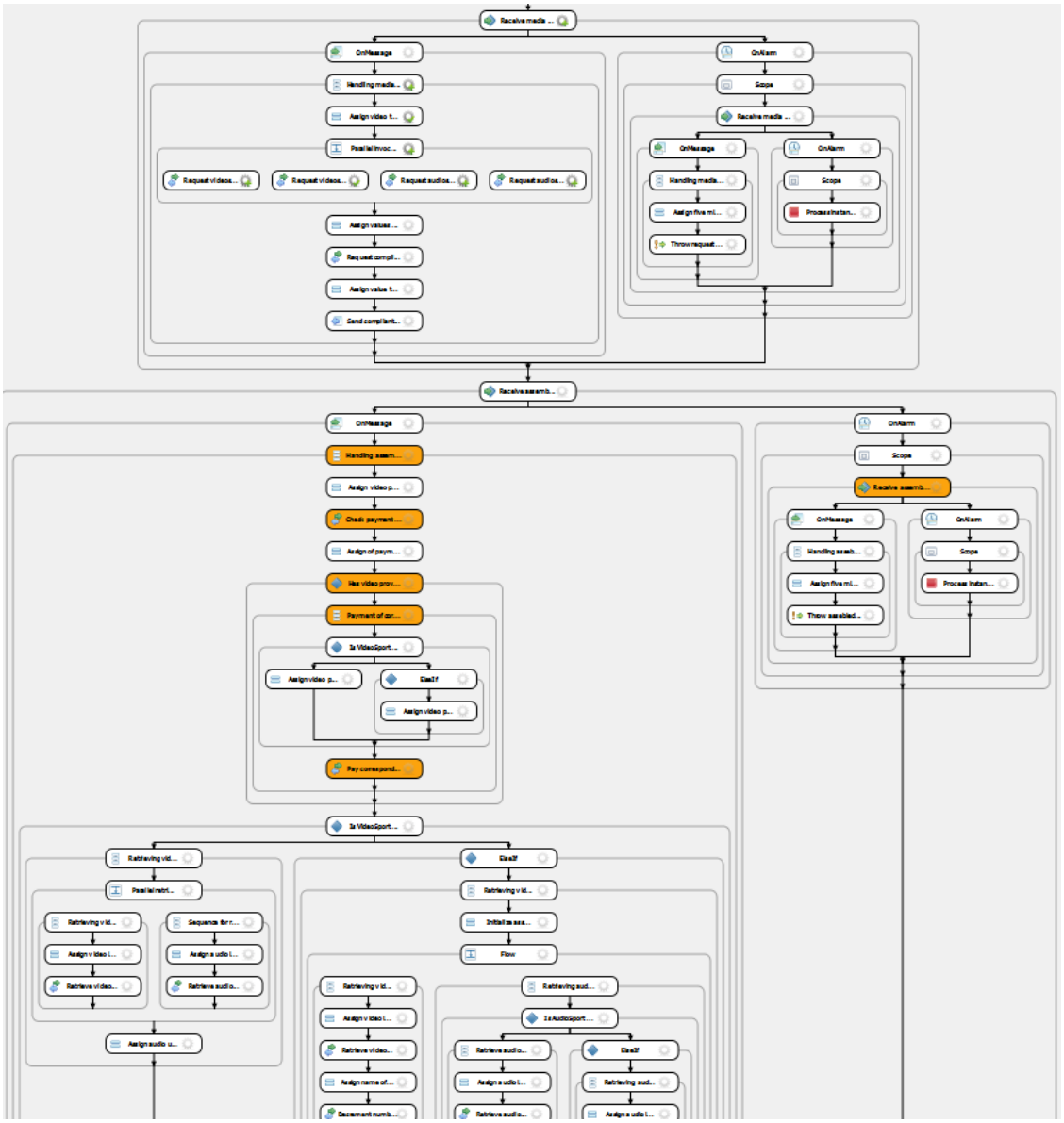


Abbildung 11.3: Darstellung einer abstrahierten Prozessinstanz im Kompaktmodus mit hervorgehobenen rr-Fragmenten [COM10b].

KAPITEL 12

ERGEBNISSE

12.1 Zusammenfassung

In der vorliegenden Dissertation wurde eine Mustersprache entwickelt, mit der Sichten auf Prozesse in abstrakter Form beschrieben werden können. Die Elemente dieser Mustersprache wurden von bestehenden, aber uneinheitlichen und zuvor nur schwer kombinierbaren Transformationstechniken aus verschiedenen Bereichen abgeleitet, aus deren Anwendungskontext herausgelöst und in einer einheitlichen Form als Muster beschrieben. Auf diese Weise werden diese Techniken einem größeren Nutzerkreis zugänglich. Die Innovation der auf diese Weise entstandenen musterbasierten Entwurfsmethode liegt in der neu geschaffenen Möglichkeit, Sichten auf Prozesse weitgehend unabhängig von konkreten Prozesssprachen, Werkzeugen und Technologien spezifizieren zu können. Ferner besteht mit der Mustersprache nun die Möglichkeit, Prozesswerkzeuge und Abstraktionsansätze systematisch auf deren Sichtenfunktionalität hin zu analysieren und vergleichen zu können.

Anhand von Chevron-Diagrammen wurde der weiterführende Ansatz sprachübergreifender Sichten diskutiert, welche ebenfalls mit der musterbasierten Entwurfsmethode beschrieben werden können. Der Stand der Technik wurde durch instanzprojizierende Sichten erweitert, die eine neuartige Möglichkeit bieten, Prozessinstanzen vereinfacht darzustellen.

Unter Einsatz der Mustersprache wurden Sichten für gängige Fragestellungen im Lebenszyklus von Geschäftsprozessen entworfen. Diese Sichtenentwürfe können als Grundlage für die Realisierung von Sichtenfunktionalität in unterschiedlichen Werkzeugen verwendet werden. Beispielhaft wurden Realisierungsmöglichkeiten in einer SOA diskutiert, wobei Transformationsfunktionen als Dienst bereitgestellt und zu einer Dienstorchestrierung verbunden werden.

Vertiefend wurde der Themenkomplex Compliance betrachtet und der Stand der Technik auf diesem Gebiet um den Ansatz der rr-Fragmente erweitert. Dadurch kann ein Prozess auf nachvollziehbare Weise um regelkonformes Verhalten erweitert werden. Exemplarisch für diesen Themenkomplex wurden verschiedene Sichten auf Basis der Mustersprache entworfen und prototypisch in Prozesswerkzeugen realisiert.

12.2 Erkenntnisse

Komplexität. Die Informatik bietet verschiedene Prinzipien, um komplexe Fragestellungen in kleinere zu zerlegen, nicht relevante Informationen zu verbergen und bestehende Lösungen wiederzuverwenden. Der Ansatz, der mit der Mustersprache verfolgt wird, setzt diese Prinzipien ein, um die inhärente Komplexität bestehender Techniken zur Komplexitätsreduzierung durch deren Abstraktion zu Mustern zunächst zu verringern. Durch den Einsatz von Sichtentechnologie wird allerdings neue Komplexität eingeführt, die in der Aufgabe des Informationsdesigns zusammengefasst wurde. Die neue Komplexität besteht darin, Informationsbedarfe zu erfassen, diesen Bedarfen durch Sichten nachzukommen und diese Sichten in Werkzeugen zu realisieren, bereitzustellen und zu pflegen.

Korrektheit. Mit der Klassifikation von Sichten wurde eine Möglichkeit zur Einordnung von Sichtenentwürfen eingeführt. Nicht strukturverändernde Sichten, in denen lediglich eine Modifikation der Darstellung oder auch die Einbindung externer Datenquellen vorgenommen wird, wurden wohlwollend aufgenommen: Wie bspw. die Hervorhebung von rr-Fragmenten, eine Detail- und Kompaktansicht von Aktivitäten, oder die Darstellung von Aktivitätszuständen für das Monitoring. Strukturverändernde Sichten haben allerdings Bedenken hervorgerufen, wie bspw.

„Wenn ich eine solche Sicht habe, ist das dann nicht falsch?“

„Was ist, wenn ich Entscheidungen falsch treffe, da die Sicht doch eben nicht das Wesentliche zeigt, sondern etwas Wesentliches weglässt?“

Eine Möglichkeit, um mit diesen Zweifeln angemessen umzugehen, besteht darin, ein weitreichendes Verständnis der Auswirkungen von Sichtentransformationen bei den Nutzern aufzubauen, damit sie sich derartige Fragen in ihrem konkreten Kontext selbst beantworten können. Dieser Ansatz ist den Rückmeldungen aus den Workshops zufolge vermutlich in der Praxis nicht umsetzbar. Eine andere Möglichkeit besteht darin, einen Satz an Sichten zu standardisieren und für diese Sichten erforderliche Vorbedingungen und geltende Nachbedingungen zu fixieren. Man könnte die Standardisierung als vertrauensbildende Maßnahme in die Sichtentechnologie verstehen.

Musterbasierter Sichtenentwurf. In der Arbeit mit der Mustersprache und der darauf aufbauenden Entwurfsmethode konnten verschiedene Erkenntnisse gewonnen werden:

- Eine Vorgehensweise, die sich im Sichtenentwurf bewährt hat, besteht darin, einen Prozess zunächst mit Informationen anzureichern, bei Bedarf die Struktur des Prozesses zu verändern und anschließend das Resultat entsprechend der Darstellungsvorgaben der Nutzer der Sicht aufzubereiten.
- Prozessmodelle enthalten gegenwärtig zu wenige Informationen im Vergleich zu dem prozessbezogenen Wissen, das in einzelnen Personen wie einem Modellierer oder einem Prozesseigner verhaftet ist. Wird dieses Wissen in irgendeiner Form semantischer Annotation mit dem Prozess in Verbindung gebracht, steigen die Möglichkeiten und der Nutzen von Sichten signifikant.

- Analog zu menschlichem Know-how ist davon auszugehen, dass es i.d.R. zahlreiche Datenquellen gibt, die prozessbezogene Informationen enthalten. Diese Informationen können allerdings aus Mangel an Anbindung an Werkzeuge zur Prozessmodellierung oder -analyse nur unzureichend im Kontext des Prozessmodells genutzt werden.
- Es ist anzunehmen, dass sich Ansätze zur automatischen Abstraktion (M2, M3) nicht etablieren werden, da eine treffende und dem Kontext eines Prozesses angemessene Benennung aggregierter Strukturen eine höhere geistige Leistung darstellt, die profunde Kenntnisse über einen Prozess, das jeweilige Unternehmen und die Terminologie der zu unterstützenden Interessengruppen erfordert.
- Während eine ausdrucksstarke Sprache aus wissenschaftlicher Perspektive interessant ist, ist sie aus praktischer Perspektive für die breite Nutzerzahl schnell zu kompliziert, d.h. komplex und zugleich schwer verständlich. Die Mustersprache kann daher als Hilfsmittel für eine eingeschränkte Gruppe von Experten betrachtet werden.

rr-Fragmente. Ob und inwieweit der Ansatz der rr-Fragmente, darauf bezogener Sichten und damit verbundener Anpassungen an Werkzeugen einen Weg in die Praxis des Compliance Managements finden, ist durch verschiedene Faktoren nur schwer abzusehen. Für den Einsatz von rr-Fragmenten bestehen bspw. kaum Angriffspunkte, solange Prozesse nicht stringent in Form von Prozessmodellen dokumentiert sind und diese Modelle auch tatsächlich gelebt und gepflegt werden.

KAPITEL 13

AUSBLICK

Die Rückmeldungen im Rahmen der Validierung haben zum Ausdruck gebracht, dass eine durchgängige Unterstützung der musterbasierten Methode in Werkzeugen eine wichtige Voraussetzung für deren Einsatz in der Praxis ist. Da ein Unternehmen jedoch i.d.R. keine eigenen Prozesswerkzeuge entwickelt, sondern sich auf sein Kerngeschäft konzentriert, können Sichten und deren musterbasierter Entwurf nur durch Werkzeughersteller tatsächlich Erfolg haben. Diesen Vendoren obliegt zudem die Standardisierung, die von den Industrievertretern als zentraler Akzeptanzfaktor für die Sichtentechnologie genannt wurde.

Standardisierung von Sichten. Durch die Identifikation und Beschreibung von Mustern wurden Ideen und bewährte Techniken einfach zugänglich gemacht. Allerdings besteht durch diesen einfachen Zugang die Gefahr, dass man durch eine geschickte Kombination von Mustern eine solch gute Lösung entwirft, dass sie bereits angemeldeten Schutzrechten unterliegt, bspw. den Patentanmeldungen zu Sichten auf Prozesse [GHN+08][GHN+09]. Bestehende Schutzrechte anderer Firmen sind ein Hindernis für Werkzeughersteller, schließen die Standardisierung einer geringen Zahl an Sichten aber nicht aus. Sofern die Besitzer von relevanten Schutzrechten an einem Standardisierungsvorhaben teilnehmen, sind diese i.d.R. dazu aufgefordert, diese Patente freizugeben. Eine Chance besteht darin, einen Satz an häufig benötigten und möglichst vielseitig einsetzbaren Sichten zu entwerfen, für die keine Schutzrechte bestehen. Die Mustersprache bietet an dieser Stelle eine Möglichkeit, diesbezügliche Diskussionen auf einer abstrakten gedanklichen Ebene zu führen, welche weitgehend von konkreten Prozesssprachen, Werkzeugen und Technologien abstrahiert. Im weiteren Verlauf einer Standardisierungsbemühung müssten die entworfenen Sichten aber für eine oder auch für mehrere konkrete Prozesssprachen ausgestaltet und deren Funktionsweise eindeutig spezifiziert werden. Eine Referenzimplementierung wäre erforderlich, um eine einheitliche Funktionsweise der Transformationen zu gewährleisten und um Fehler in den Sichtenentwürfen auszuschließen.

Erweiterung des Sichtenansatzes. Sichten auf Prozesse sind ein generischer Ansatz, um bestehende Prozessmodelle zu abstrahieren, zu konkretisieren, mit Informationen anzureichern und interessengruppenspezifisch darzustellen. Sofern ein Prozess über die Elemente des PM-Graphen hinaus weitere Elementtypen oder Verbindungstypen beinhaltet, kann die Mustersprache um entsprechende Muster erweitert werden. Zor et al. [ZLS11] beschreiben bspw., dass in Herstellungsprozessen neben dem Kontroll- und Datenfluss auch der Materialfluss und die Nutzung von Maschinen und Werkzeugen modelliert werden muss.

Auch für diese weiteren Element- und Verbindungstypen besteht der Bedarf, diese ganz oder teilweise auszublenden oder bestimmte Umfänge hervorzuheben.

Übertragung des Sichtenansatzes. Ein Teil der zu Mustern abstrahierten Transformationstechniken wurde Bereichen ohne Prozessbezug entnommen, bspw. die Aggregation von Pfaden (M7) stammt aus der Abstraktion von Architekturdiagrammen. Umgekehrt können die so gewonnenen Techniken auch vom Bereich Geschäftsprozesse auf andere Bereiche übertragen und auf andere Typen von Modellen angewendet werden. Offensichtlich kann eine Farbcodierung zur Darstellung bestimmter Eigenschaften universell verwendet werden. Auch kann ein Ausgrauen oder Entfernen von Modellelementen gänzlich verschiedene Modelltypen auf wesentliche Aspekte reduzieren. Binz et al. [BLN+12] haben eine derartige Übertragung vorgenommen, indem sie einen Teil der identifizierten Sichtenentwurfsmuster dazu verwendet haben, Sichten zur Analyse der IT-Infrastruktur zu definieren, die in Form eines hochkomplexen Modells vorliegt (engl. *enterprise topology graph*). Die Übertragbarkeit ist nicht auf den IT-Bereich beschränkt, sondern Sichten können auf diverse Modelltypen und Graphen angewendet werden, wie etwa auf den Kompositionsgraphen eines Kostüms im Bereich der vestimentären Kommunikation, wie Schumm et al. [SBL+12] anhand einer Farbcodierung der Kleidungsstücke eines Kostüms demonstrieren.

Compliance. In der vorliegenden Dissertation wurden zunächst verschiedene Ansätze zur Gewährleistung von Compliance betrachtet, dann der Stand der Technik um rr-Fragmente erweitert und dieser Ansatz anschließend mit Techniken der Compliance-Prüfung verbunden. Darüber hinaus wurden Sichten entworfen, um die Handhabung von rr-Fragmenten zu unterstützen. rr-Fragmente repräsentieren letztlich eine mehr oder weniger bestimmte Art von Anforderungen an einen Prozess. Es liegt auf der Hand, dass diese Anforderungen nicht notwendigerweise aus einem Gesetz oder einer internen Richtlinie stammen müssen, damit sie durch ein rr-Fragment umgesetzt werden können. Insofern stellt sich die Frage, inwieweit der vorgestellte Ansatz eine Bereicherung für das Requirements Engineering darstellt. Während Prozessbausteine in diesem Bereich zwar lange bekannt sind [RP93] und die Gewährleistung der Nachvollziehbarkeit ein Best Practice darstellt, können Sichten hier möglicherweise neue Impulse geben.

KAPITEL 14

ANHANG

14.1 Literaturverzeichnis

Alle URLs wurden zuletzt am 15.11.2014 geprüft.

[Aal99] W. van der Aalst, *Formalization and Verification of Event-driven Process Chains*. In: Information and Software Technology, 41(10):639–650, Elsevier, 1999.

[ACF96] D. Avrilionis, P. Cunin, C. Fernström, *OPSIS: A View Mechanism for Software Processes which Supports their Evolution and Reuse*. In: Proceedings of the 18th International Conference on Software Engineering (ICSE), IEEE Computer Society, 1996.

[ADG+05] W. van der Aalst, A. Dreiling, F. Gottschalk, M. Rosemann, M. H. Jansen-Vullers, *Configurable Process Models as a Basis for Reference Modeling*. In: Proceedings of the Workshop on Business Process Reference Models (BPRM), Springer, 2005.

[ADW08] A. Awad, G. Decker, M. Weske, *Efficient Compliance Checking using BPMN-Q and Temporal Logic*. In: Proceedings of the 6th International Conference on Business Process Management (BPM'08), Springer, 2008.

[AHE+06] M. Adams, A. ter Hofstede, T. Edmond, W. van der Aalst, *Worklets: A Service-oriented Implementation of Dynamic Flexibility in Workflows*. In: Proceedings of the 14th International Conference on Cooperative Information Systems (CoopIS), Springer, 2006.

[AHK06] J. Abello, F. van Ham, N. Krishnan, *ASK-GraphView: A Large Scale Graph Visualization System*. In: IEEE Transactions on Visualization and Computer Graphics, 12(5):669–676, IEEE, 2006.

[AHK+03] W. van der Aalst, A. ter Hofstede, B. Kiepuszewski, A. Barros, *Workflow Patterns*. In: Distributed and Parallel Databases, 14(1):5–51, Springer, 2003.

[AHV95] S. Abiteboul, R. Hull, V. Vianu, *Foundations of Databases*. Addison-Wesley, 1995.

[AIS97] C. Alexander, S. Ishikawa, M. Silverstein, *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, USA, 1997.

[AKM+08] F. Arbab, C. Koehler, Z. Maraikar, Y. Moon, J. Proenca, *Modeling, Testing and Executing Reo Connectors with the Eclipse Coordination Tools*. In: Tool Demo Session at FACS'08, 2008.

[AM08] F. Abouzaid, J. Mullins, *A Calculus for Generation, Verification, and Refinement of BPEL Specifications*. In: Electronic Notes in Theoretical Computer Science, 200:43–65, Elsevier, 2008.

[Ans09] T. Anstett, *XMLO*. 2009. [Online] Verfügbar: http://www.iaas.uni-stuttgart.de/forschung/projects/master/xmlo.php

[Apa11a] Apache Software Foundation, *Apache Axis2*. 2011. [Online] Verfügbar: http://axis.apache.org/axis2/java/core/

[Apa11b] Apache Software Foundation, *Apache MyFaces*. 2011. [Online] Verfügbar: http://myfaces.apache.org/index.html

[Apa11c] Apache Software Foundation, *Apache Tomcat*. 2011. [Online] Verfügbar: http://tomcat.apache.org/

[Apa11d] Apache Software Foundation, *Apache ODE*. 2011. [Online] Verfügbar: http://ode.apache.org/

[Arb04] F. Arbab, *Reo: A Channel-based Coordination Model for Component Composition*. In: Mathematical Structures in Computer Science, 14(3):329–366, Cambridge University Press, 2004.

[AW10] A. Awad, M. Weske, *Visualization of Compliance Violation in Business Process Models*. In: Proceedings of the 5th International Workshop on Business Process Intelligence, Springer, 2010.

[Awa10] A. Awad, *A Compliance Management Framework for Business Process Models*. Dissertation, Universität Potsdam, 2010.

[Bas06] Basel Committee on Banking Supervision, *Basel II: International Convergence of Capital Measurement and Capital Standards: A Revised Framework - Comprehensive Version*. Bank for International Settlements, 2006.

[Bas10] Basel Committee on Banking Supervision, *Basel III: International Framework for Liquidity Risk Measurement, Standards and Monitoring*. Bank for International Settlements, 2010.

[BAS+10] I. Brandic, T. Anstett, D. Schumm, F. Leymann, S. Dustdar, R. Konrad, *Compliant Cloud Computing (C3): Architecture and Language Support for User-driven Compliance Management in Clouds*. In: Proceedings of the 3rd International Conference on Cloud Computing (IEEE Cloud), IEEE Computer Society, 2010.

[BDH05] A. Barros, M. Dumas, A. ter Hofstede, *Service Interaction Patterns*. In: Proceedings of the 3rd International Conference on Business Process Management, Springer, 2005.

[BDL+10] A. Birukou, V. D'Andrea, F. Leymann, J. Serafinski, P. Silveira, S. Strauch, M. Tluczek, *An Integrated Solution for Runtime Compliance Governance in SOA*. In: Proceedings of the 8th International Conference on Service-Oriented Computing (ICSOC'10), Springer, 2010.

[BDW07] M. Born, F. Dörr, I. Weber, *User-friendly Semantic Annotation in Business Process Modeling*. In: Proceedings of Web Information Systems Engineering Workshops (WISE), Springer-Verlag, 2007.

[BEL+07] S. Buckl, A. Ernst, J. Lankes, K. Schneider, C. Schweda, *A Pattern based Approach for Constructing Enterprise Architecture Management Information Models*. In: Wirtschaftsinformatik Proceedings 2007, AIS Electronic Library (AISeL), 2007.

[BEL+08] S. Buckl, A. Ernst, J. Lankes, F. Matthes, *Enterprise Architecture Management Pattern Catalog (Version 1.0)*. Technischer Bericht TB 0801, TU München, Software Engineering for Business Information Systems (sebis), 2008.

[BKL+09] M. Bischof, O. Kopp, T. van Lessen, F. Leymann, *BPELscript: A Simplified Script Syntax for WS-BPEL 2.0*. In: Proceedings of the 35th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2009), IEEE, 2009.

[BLN+12] T. Binz, F. Leymann, A. Nowak, D. Schumm, *Improving the Manageability of Enterprise Topologies Through Segmentation, Graph Transformation, and Analysis Strategies*. In: Proceedings of 16th IEEE Enterprise Distributed Object Computing Conference (EDOC), IEEE, 2012.

[BMR08] A. Benameur, F. Massacci, N. Rassadko, *Security Views for Outsourced Business Processes*. In: Proceedings of the 2008 ACM Workshop on Secure Web Services, ACM, 2008.

[Bob08] R. Bobrik, *Konfigurierbare Visualisierung komplexer Prozessmodelle*. Dissertation, Universität Ulm, 2008.

[BPR07] J. Becker, D. Pfeiffer, M. Räckers, *PICTURE – A new Approach for Domain-Specific Process Modelling*. In: Proceedings of the CAiSE'07 Forum at the

19th International Conference on Advanced Information Systems Engineering, CEUR Workshop Proceedings, 2007.

[BRW07] L. Baacke, P. Rohner, R. Winter, *Aggregation of Reference Process Building Blocks to Improve Modeling in Public Administrations*. In: Proceedings of Ongoing Research, Project Contributions and Workshops of the 6th International EGOV Conference, Trauner Druck Verlag, 2007.

[BSI09] Bundesamt für Sicherheit in der Informationstechnik, *Informationssicherheit: Ein Vergleich von Standards und Rahmenwerken*. 2009.

[Bun12] Bundesverwaltungsamt, *Verzahnung von Fachbereichen und IT*. In: Schriftenreihe des Kompetenzzentrums Prozessmanagement, Köln, Oktober 2012.

[Cai10] J. Cai, *Abstrakte Sichten auf BPEL Prozesse*. Studienarbeit Nr. 2250, Universität Stuttgart, 2010.

[Cai11] J. Cai, *Entwicklung eines Frameworks zur Verwaltung von abstrakten Sichten auf BPEL Prozesse*. Diplomarbeit Nr. 3079, Universität Stuttgart, 2011.

[CCF+00] F. Casati, S. Castano, M. Fugini, I. Mirbel, B. Pernici, *Using Patterns to Design Rules in Workflows*. In: IEEE Transactions on Software Engineering, 26(8):760–785, IEEE, 2000.

[CDT06] I. Chebbi, S. Dustdar, S. Tata, *The View-based Approach to Dynamic Inter-organizational Workflow Cooperation*. In: Data & Knowledge Engineering, 56(2):139–173, Elsevier, 2006.

[CH03] K. Czarnecki, S. Helsen, *Classification of Model Transformation Approaches*. In: Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture, Anaheim, 2003.

[Cha04] D. Chappell, *Enterprise Service Bus: Theory in Practice*. O'Reilly Media, 2004.

[Chi08] M. Chinosi, *Representing Business Processes: Conceptual Model and Design Methodology*. PhD Thesis, Università degli studi dell'Insubria, Italien, 2008.

[CKA10] B. Changizi, N. Kokash, F. Arbab, *A Unified Toolset for Business Process Model Formalization*. In: Proceedings of the International Workshop on Formal Engineering approaches to Software Components and Architectures (FESCA'10), 2010.

[CLR08] D. Caprasse, J. Laurent, W. Reed, *Three Lines of Defence: How to take the Burden out of Compliance*. In: European Insurance Digest, PricewaterhouseCoopers, April 2008.

[CMN+06] J. Cardoso, J. Mendling, G. Neumann, H. Reijers, *A Discourse on Complexity of Process Models*. In: Proceedings of Workshop on Business Process Design, Springer, 2006.

[COM09] COMPAS Project, *Initial Specification of Compliance Language Constructs and Operators*. Project Deliverable D2.1, 2009. [Online] Verfügbar: http://ec.europa.eu/information_society/apps/projects/logos/5/215175/080/deliverables/D2.2_Initial-specification-of-compliance-language-constructs-and-operators.pdf

[COM10a] COMPAS Project, *BPEL Extensions for Compliant Services*. Project Deliverable D4.2, 2010. [Online] Verfügbar: http://ec.europa.eu/information_society/apps/projects/logos/5/215175/080/deliverables/D4.2_BPEL-extensions-for-compliant-services.pdf

[COM10b] COMPAS Project, *Runtime Environment*. Project Deliverable D1.4, 2010. [Online] Verfügbar: http://ec.europa.eu/information_society/apps/projects/logos/5/215175/080/deliverables/001_D14RuntimeEnvironment.pdf

[COM10c] COMPAS Project, *Supporting Infrastructure – Process Engine, Process Artefact Repository, Process Generation Tool*. Project Deliverable D4.4 final version, 2010. [Online] Verfügbar: http://ec.europa.eu/information_society/apps/projects/logos/5/215175/080/deliverables/001_D44bSupportingInfrastructurefinal.pdf

[COM10d] COMPAS Project, *Conceptual Model and Terminology*. 2010.

[COM11] COMPAS Project, *Achievements & Lessons Learned*. 2011.

[Con02] Congress of the United States, *Public Company Accounting Reform and Investor Protection Act (Sarbanes-Oxley Act)*. 2002. Pub. L. No. 107-204, 116 Stat. 745.

[DAC98] M. Dwyer, G. Avrunin, J. Corbett, *Property Specification Patterns for Finite-State Verification*. In: Proceedings of the International Workshop on Formal Methods on Software Practice, ACM, 1998.

[Dav89] F. Davis, *Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology*. In: MIS Quarterly, 13(3):319–340, 1989.

[DCD+09] F. Daniel, F. Casati, V. D'Andrea, S. Strauch, D. Schumm, F. Leymann, E. Mulo, U. Zdun, S. Dustdar, S. Sebahi, F. de Marchi, M.-S. Hacid, *Business Compliance Governance in Service-Oriented Architectures*. In: Proceedings of the IEEE 23rd International Conference on Advanced Information Networking and Applications (AINA), IEEE Computer Society, 2009.

[Den11] D. Dentsas. *Integration von Fragmento in eine Rich Client Plattform*. Studienarbeit Nr. 2331, Universität Stuttgart, 2011.

[Den12] D. Dentsas, *Einbindung von Menschen in Scientific Workflows*. Diplomarbeit Nr. 3297, Universität Stuttgart, 2012.

[DGW94] W. Deiters, V. Gruhn, H. Weber, *Software Process Evolution in Melmac*. In: The Impact of CASE Technology on Software Processes, World Scientific, 1994.

[DHL+12] L. Dyer, F. Henry, I. Lehmann, G. Lipof, F. Osmani, D. Parrott, W. Peeters, J. Zahn, *Scaling BPM Adoption: From Project to Program with IBM Business Process Manager*. IBM Redbook, 2012.

[Dij82] E. W. Dijkstra, *EWD 447: On the Role of Scientific Thought*. In: Selected Writings on Computing: A Personal Perspective, Springer-Verlag, 1982.

[Dis11] The Displaytag Team, *Display Tag Library*, 2011. [Online] Verfügbar: http://www.displaytag.org/1.2/

[DK75] F. DeRemer, H. Kron, *Programming in the Large versus Programming in the Small*. In: Proceedings of the International Conference on Reliable Software, ACM, 1975.

[DR09] P. Dadam, M. Reichert, *The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support - Challenges and Achievements*. In: Computer Science - Research and Development, 23(2):81–97, Springer, 2009.

[DS96] P. Darke, G. Shanks, *Stakeholder Viewpoints in Requirements Definition: A Framework for Understanding Viewpoint Development Approaches*. In: Requirements Engineering, 1(2):88–105, Springer, 1996.

[Ebe14] H. Eberle, *Prozessbausteine*. Dissertation, Universität Stuttgart, 2014.

[EC11] European Commission, *FP7-215175 COMPAS Review Report 4*, 2011.

[Ecl11] The Eclipse Foundation, *Eclipse BPEL Designer*. 2011. [Online] Verfügbar: http://www.eclipse.org/bpel/

[EG08] R. Eshuis, P. Grefen, *Constructing Customized Process Views*. In: Data & Knowledge Engineering, 64(2):419–438, Elsevier, 2008.

[Elg12] A. Elgammal, *Towards a Comprehensive Framework for Business Process Compliance*. Dissertation, Tilburg University, 2012.

[ELS+10] H. Eberle, F. Leymann, D. Schleicher, D. Schumm, T. Unger, *Process Fragment Composition Operations*. In: Proceedings of the IEEE Asia-Pacific Services Computing Conference (APSCC), IEEE Computer Society, 2010.

[Esp12] EsperTech, *Esper*. 2012. [Online] Verfügbar: http://esper.codehaus.org/

[EUL09] H. Eberle, T. Unger, F. Leymann, *Process Fragments*. In: Proceedings of the 17th International Conference on Cooperative Information Systems (CoopIS), Springer, 2009.

[Evi10] Eviware, *SoapUI*, 2011. [Online] Verfügbar: http://www.soapui.org/

[Fah95] U. Fahrwinkel, *Methode zur Modellierung und Analyse von Geschäftsprozessen zur Unterstützung des Business Process Reengineering*. Dissertation, Universität-Gesamthochschule Paderborn, 1995.

[FBB+14] M. Falkenthal, J. Barzen, U. Breitenbücher, C. Fehling, F. Leymann, *From Pattern Languages to Solution Implementations*. In: Proceedings of the 6th International Conferences on Pervasive Patterns and Applications (PATTERNS 2014), Xpert Publishing Services, 2014.

[FES05] A. Förster, G. Engels, T. Schattkowsky, *Activity Diagram Patterns for Modeling Quality Constraints in Business Processes*. In: Proceedings of the 8th International Conference on Model Driven Engineering Languages and Systems, Springer, 2005.

[FGM+03] G. Ferrari, S. Gnesi, U. Montanari, M. Pistore, *A Model-checking Verification Environment for Mobile Processes*. In: ACM Transactions on Software Engineering and Methodology (TOSEM), 12(4):440–473, ACM, 2003.

[Fil11] H.-G. Fill, *Using Semantically Annotated Models for Supporting Business Process Benchmarking*. In: Proceedings of the 10th International Conference on Perspectives in Business Informatics Research, Springer, 2011.

[FT96] W. Frakes, C. Terry, *Software Reuse: Metrics and Models*. In: ACM Computing Surveys (CSUR), 28(2):415–435, ACM, 1996.

[Gar09] Gartner, *Gartner Identifies the Top 10 Strategic Technologies for 2010*. 2009. [Online] Verfügbar: http://www.gartner.com/it/page.jsp?id=1210613

[GCS+10] G. Gheorghe, B. Crispo, D. Schleicher, T. Anstett, F. Leymann, R. Mietzner, G. Monakova, *Combining Enforcement Strategies in Service Oriented Architectures*. In: Proceedings of the 8th International Conference on Service-Oriented Computing (ICSOC'10), Springer, 2010.

[GD11] G. R. Gangadharan, V. D'Andrea, *Service Licensing: Conceptualization, Formalization, and Expression*. In: Service Oriented Computing and Applications, 5(1):37–59, Springer, 2011.

[GHJ95] E. Gamma, R. Helm, R. Johnson. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman, 1995.

[GHN+08] P. Gilbert, D. Heredia, M. Nonemacher, M. Moeller, G. Sanderson, A. Cotner, P. Chobantonov, A. Moffat, M. Howitt, *Business Process Diagram Visualization Using Heat Maps*. United States Patent Application Publication US 2008/0120573 A1, Mai 2008.

[GHN+09] C. Guo, Y. Hui, J. Nan, T. Ou, W. Sun, Z. Wang, W. Xu, Y. Zhang, *Apparatus and Method for Generating a Monitoring View of an Executable Business Process*. United States Patent Application Publication, US 2009/0112666 A1, April 2009.

[GKL+11] K. Görlach, O. Kopp, F. Leymann, D. Schumm, S. Strauch, *WS-BPEL Extension for Compliance Fragments (BPEL4CFrags)*. Technischer Bericht Nr. 2011/01, Universität Stuttgart, 2011.

[GKW08] T. Gschwind, J. Köhler, J. Wong, *Applying Patterns During Business Process Modeling*. In: Proceedings of the 6th International Conference on Business Process Management (BPM), Springer, 2008.

[GL06] V. Gruhn, R. Laue, *Complexity Metrics for Business Process Models*. In: Proceedings of the 9th International Conference on Business Information Systems, Springer, 2006.

[GL09] V. Gruhn, R. Laue, *Reducing the Cognitive Complexity of Business Process Models*. In: Proceedings of the 8th IEEE International Conference on Cognitive Informatics (ICCI'09), IEEE, 2009.

[GMW97] D. Garlan, R. Monroe, D. Wile, *ACME: An Architecture Description Interchange Language*. In: Proceedings of CASCON 97, 1997.

[GRA05] F. Gottschalk, M. Rosemann, W. van der Aalst, *My Own Process: Providing Dedicated Views on EPCs*. In: EPK, Gesellschaft für Informatik, 2005.

[GSL+05] A. Goderis, U. Sattler, P. Lord, C. Goble, *Seven Bottlenecks to Workflow Reuse and Repurposing*. In: Proceedings of the 4th International Semantic Web Conference, Springer, 2005.

[HC93] M. Hammer, J. Champy, *Reengineering the Corporation*. HarperCollins Publishers, 1993.

[Hir91] J. Hirschberger, *Geschichte der Philosophie, Band I: Altertum und Mittelalter*. 14. Auflage, Herder, 1991.

[HL93] H.-J. Habermann, F. Leymann, *Repository: Eine Einführung*. Oldenbourg, 1993.

[HLD+05] M. Hepp, F. Leymann, J. Domingue, A. Wahler, D. Fensel, *Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management*. In: Proceedings of the IEEE International

Conference on e-Business Engineering (ICEBE), IEEE Computer Society, 2005.

[HPO08] W.-J. van den Heuvel, M. Papazoglou, B. Orriens, *On the Risk Management and Auditing of SOA based Business Processes*. In: Proceedings of the 3rd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA), Springer, 2008.

[HRL09] O. Holschke, J. Rake, O. Levina, *Granularity as a Cognitive Factor in the Effectiveness of Business Process Model Reuse*. In: Proceedings of the 7th International Conference on Business Process Management (BPM'09), Springer, 2009.

[HW03] G. Hohpe, B. Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Longman, 2003.

[HZD+10] T. Holmes, U. Zdun, F. Daniel, S. Dustdar, *Monitoring and Analyzing Service-based Internet Systems through a Model-Aware Service Environment*. In: Proceedings of the 22nd International Conference on Advanced Information Systems Engineering, Springer, 2010.

[IBM09] International Business Machines Corporation (IBM), *WebSphere Business Modeler*, 2009.

[IDW01] Institut der Wirtschaftsprüfer (IDW), *IDW Prüfstandard EPS 330: Abschlußprüfung bei Einsatz von Informationstechnologie*, 2001.

[ISO11] ISO / IEC / IEEE, *Systems and Software Engineering - Architecture Description*. Reference Number ISO/IEC/IEEE 42010:2011(E), 2011.

[ITG05] IT Governance Institute (ITGI), *Control Objectives for Information and related Technology (COBIT)*. Version 4.0, 2005.

[JBo11] JBoss, *Hibernate*. 2011. [Online] Verfügbar: http://www.hibernate.org/

[JBS97] S. Jablonski, M. Böhm, W. Schulze, *Workflow Management: Entwicklung und Anwendung von Systemen*. dpunkt Verlag, 1997.

[JFE+11] E.-M. Jakobs, R. Fiehler, D. Eraßme, A. Kursten, *Industrielle Prozessmodellierung als kommunikativer Prozess. Eine Typologie zentraler Probleme*. In: Gesprächsforschung 12:223–264, 2011.

[KDS+12] D. Karastoyanova, D. Dentsas, D. Schumm, M. Sonntag, L. Sun, K. Vukojevic, *Service-based Integration of Human Users in Workflow-driven Scientific Experiments*. In: Proceedings of the 8th IEEE International Conference on eScience (eScience 2012), IEEE, 2012.

[KEL+10] O. Kopp, H. Eberle, F. Leymann, T. Unger, *The Subprocess Spectrum*. In: Proceedings of the 3rd Business Process and Services Computing Conference, Gesellschaft für Informatik (GI), 2010.

[KGK+11] O. Kopp, K. Görlach, D. Karastoyanova, F. Leymann, M. Reiter, D. Schumm, M. Sonntag, S. Strauch, T. Unger, M. Wieland, R. Khalaf, *A Classification of BPEL Extensions.* In: Journal of Systems Integration, 2(4):3–28, Online, 2011.

[KKL08] O. Kopp, R. Khalaf, F. Leymann, *Deriving Explicit Data Links in WS-BPEL Processes*. In: Proceedings of the International Conference on Services Computing (SCC 2008), IEEE Computer Society Press, 2008.

[KKL+05] M. Kloppmann, D. König, F. Leymann, G. Pfau, A. Rickayzen, C. Riegen, P. Schmidt, I. Trickovic, *WS-BPEL Extension for Sub-processes – BPEL-SPE*. White Paper, 2005.

[KKL+09] I. Kitzmann, C. König, D. Lübke, L. Singer. *A Simple Algorithm for Automatic Layout of BPMN Processes*. In: Proceedings of the 2009 IEEE Conference on Commerce and Enterprise Computing, IEEE, 2009.

[KKR12] J.Kolb, K. Kammerer, M. Reichert, *Updatable Process Views for User-centered Adaption of Large Process Models*. In: Proceedings of the 10th International Conference on Service Oriented Computing (ICSOC'12), Springer, 2012.

[KKS+06] D. Karastoyanova, R. Khalaf, R. Schroth, M. Paluszek, F. Leymann, *BPEL Event Model*. Technischer Bericht Nr. 2006/10, Universität Stuttgart, 2006.

[KL09] D. Karastoyanova, F. Leymann, *BPEL'n'Aspects: Adapting Service Orchestration Logic*. In: Proceedings of the IEEE International Conference on Web Services (ICWS), IEEE, 2009.

[KLM+97] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, J. Irwin, *Aspect-oriented Programming*. In: Proceedings of the European Conference on Object-Oriented Programming, Springer, 1997.

[KLR+10] D. Knuplesch, L. Ly, S. Rinderle-Ma, H. Pfeifer, P. Dadam, *On Enabling Data-Aware Compliance Checking of Business Process Models*. In: Proceedings of the 29th International Conference on Conceptual Modeling, Springer, 2010.

[KMW+09] O. Kopp, D. Martin, D. Wutke, F. Leymann, *The Difference Between Graph-Based and Block-Structured Business Process Modelling Languages*. In: Enterprise Modelling and Information Systems, 4(1):3–13, 2009.

[KNS92] G. Keller, M. Nüttgens, A.-W. Scheer, *Semantische Prozeßmodellierung auf der Grundlage „Ereignisgesteuerter Prozeßketten (EPK)"*.

Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi), Universität des Saarlandes, Heft 89, 1992.

[KR13a] J. Kolb, M. Reichert, *A Flexible Approach for Abstracting and Personalizing Large Business Process Models*. In: Applied Computing Review, 13(1):6–17, ACM, 2013.

[KR13b] J. Kolb, M. Reichert, *Supporting Business and IT through Updatable Process Views: The proView Demonstrator*. In: Demo Track of the 10th International Conference on Service Oriented Computing (ICSOC'12), Springer, 2013.

[Kre11] J. Krein, *Web-based Application Integration: Advanced Business Process Monitoring in WSO2 Carbon*. Studienarbeit Nr. 2311, Universität Stuttgart, 2011.

[KSM+08] M. Kharbili, S. Stein, I. Markovic, E. Pulvermüller, *Towards a Framework for Semantic Business Process Compliance Management*. In: Proceedings of the Workshop on Governance, Risk and Compliance for Information Systems, CEUR Proceedings, 2008.

[KTD09] K. Klai, S. Tata, J. Desel, *Symbolic Abstraction and Deadlock-Freeness Verification of Inter-Enterprise Processes*. In: Proceedings of the 7th International Conference on Business Process Management (BPM'09), Springer, 2009.

[KVF+13] J. Küster, H. Völzer, C. Favre, M. C. Branco, K. Czarnecki, *Supporting Different Process Views through a Shared Process Model*. In: Proceedings of the 9th European Conference on Modelling Foundations and Applications, Springer, 2013.

[LA94] F. Leymann, W. Altenhuber, *Managing Business Processes as an Information Resource*. In: IBM Systems Journal, 33(2):326–348, 1994.

[Lat10] G. Latuske, *Sichten auf Geschäftsprozesse als Werkzeug zur Darstellung laufender Prozessinstanzen*. Diplomarbeit Nr. 3036, Universität Stuttgart, 2010.

[LD99] F. Lindert, W. Deiters, *Modelling Inter-organizational Processes with Process Model Fragments*. In: Proceedings of Workshop Informatik '99: Enterprise-wide and Crossenterprise Workflow Management: Concepts, Systems, Applications, CEUR Workshop Proceedings, 1999.

[LGC12] J. de Lara, E. Guerra, J. Cuadrado, *Abstracting Modelling Languages: A Reutilization Approach*. In: Proceedings of the 24th International Conference on Advanced Information Systems Engineering (CAiSE), Springer, 2012.

[LGR+08] L. Ly, K. Göser, S. Rinderle-Ma, P. Dadam, *Compliance of Semantic Constraints – A Requirements Analysis for Process Management Systems*. In:

Proceedings of the 1st International Workshop on Governance, Risk and Compliance – Applications in Information Systems (GRCIS'08), CEUR, 2008.

[Lie11] S. Lie, *Abstract Business Process Monitoring*. Studienarbeit Nr. 2316, Universität Stuttgart, 2011.

[LKR+10] L. Ly, D. Knuplesch, S. Rinderle-Ma, K. Göser, H. Pfeifer, M. Reichert, P. Dadam, *SeaFlows Toolset – Compliance Verification Made Easy for Process-aware Information Systems*. In: Proceedings of CAiSE'10 Forum, Springer, 2010.

[LMR11] H. Leopold, J. Mendling, H. Reijers, *On the Automatic Labeling of Process Models*. In: Proceedings of the 23rd International Conference on Advanced Information Systems Engineering (CAiSE'11), Springer, 2011.

[LMX07] Y. Liu, S. Müller, K. Xu, *A Static Compliance-Checking Framework for Business Process Models*. IBM Research Report RZ 3679, 2006.

[LR00] F. Leymann, D. Roller, *Production Workflow: Concepts and Techniques*. Prentice-Hall, 2000.

[LS03] D. Liu, M. Shen, *Workflow Modeling for virtual Processes: An Order-preserving Process-view Approach*. In: Information Systems, 28(6):505–532, Elsevier, 2003.

[LVD08] N. Lohmann, E. Verbeek, R. Dijkman, *Petri Net Transformations for Business Processes - A Survey*. Transactions on Petri Nets and Other Models of Concurrency, Springer, 2008.

[Ma12] Z. Ma, *Process Fragments: Enhancing Reuse of Process Logic in BPEL Process Models*. Dissertation, Universität Stuttgart, 2012.

[MB06] O. Maqbool, H. Babri, *Automated Software Clustering: An Insight using Cluster Labels*, In: The Journal of Systems and Software, 19(11):1632–1648, Elsevier, 2006.

[ME07] C. Miles, A. Ewan, *Model and Simulate Business Processes, Part 2: Displaying SVG diagrams on the WebSphere Business Monitor Dashboards*. 2007. [Online] Verfügbar: http://www.ibm.com/developerworks/library/ar-model2/

[Met13] Method Park Software AG, *Stages 6.2*. 2013. [Online] Verfügbar: http://stages.methodpark.de

[MG06] T. Mens, P. van Gorp, *A Taxonomy of Model Transformations*. In: Electronic Notes in Theoretical Computer Science, 152:125–142, Elsevier, 2006.

[Mic10] Microsoft Corporation, *Microsoft PowerPoint 2010*. [Online] Verfügbar: http://office.microsoft.com/de-de/powerpoint/

[ML08] Z. Ma, F. Leymann, *A Lifecycle Model for Using Process Fragment in Business Process Modeling*. In: Proceedings of the 9th Workshop on Business Process Modeling, Development, and Support (BPDMS 2008), 2008.

[MLL09] Z. Ma, W. Lu, F. Leymann, *Query Structural Information of BPEL Processes*. In: Proceedings of the 4th International Conference on Internet and Web Applications and Services (ICIW 2009), IEEE, 2009.

[MLZ06] J. Mendling, K. B. Lassen, U. Zdun, *Transformation Strategies between Block-Oriented and Graph-Oriented Process Modelling Languages*. In: Multikonferenz Wirtschaftsinformatik, 2006.

[Moo09] D. Moody, *The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering*. In: IEEE Transactions on Software Engineering, 35:756–779, IEEE, 2009.

[Moo10] K. Moosmayer, *Compliance: Praxisleitfaden für Unternehmen*. Verlag C. H. Beck, 2010.

[MR11] R. Müller, A. Rogge-Solti, *BPMN for Healthcare Processes*. In: Proceedings of the 3rd Central-European Workshop on Services and their Composition (ZEUS'11), 2011.

[MRA10] J. Mendling, H.A. Reijers, W. van der Aalst, *Seven Process Modeling Guidelines (7PMG)*. In: Information and Software Technology (IST), 52(2):127–136, Elsevier, 2010.

[MRR10] J. Mendling, J. Recker, H.A. Reijers, *On the Usage of Labels and Icons in Business Process Modeling*. In: International Journal of Information System Modeling and Design (IJISMD), 1(2):40–58, IGI-Global, 2010.

[Müh04] M. zur Mühlen, *Workflow-based Process Controlling*. Logos Verlag Berlin, 2004.

[MZ05] J. Mendling, J. Ziemann, *EPK-Visualisierung von BPEL4WS Prozess-definitionen*. In: Proceedings of the 7th Workshop Software-Reengineering, Gesellschaft für Informatik (GI), 2005.

[MZD10] E. Mulo, U. Zdun, S. Dustdar, *An Event View Model and DSL for Engineering an Event-based SOA Monitoring Infrastructure*. In: Proceedings of the 4th ACM International Conference on Distributed Event-Based Systems (DEBS), ACM, 2010.

[Net06] NetBeans Community. *NetBeans BPEL Designer*. 2006. [Online] Verfügbar: http://enterprise.netbeans.org/soa/

[Nie93] J. Nielsen, *Usability Engineering*. Morgan Kaufmann, 1993.

[NKL+12] A. Nowak, D. Karastoyanova, F. Leymann, A. Rapoport, D. Schumm, *Flexible Information Design for Business Process Visualizations*. In: Proceedings of Service Oriented Computing and Applications (SOCA), IEEE Computer Society, 2012.

[NS07] K. Namiri, N. Stojanovic, *Pattern-Based Design and Validation of Business Process Compliance*. In: Proceedings of the 2007 OTM Confederated International Conference, Springer, 2007.

[NSS01] A. Naef, C. Schuler, H. Schuldt, *Monitoring komplexer Dienste in unternehmensübergreifenden Prozessen am Beispiel von SAP R/3 Business Workflows*. Datenbanksysteme in Büro, Technik und Wissenschaft (BTW‘01), Springer, 2001.

[NWG01] Network Working Group, *Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)*. RFC 3161, 2001. [Online] Verfügbar: http://tools.ietf.org/html/rfc3161

[OAS05] Organization for the Advancement of Structured Information Standards (OASIS), *eXtensible Access Control Markup Language (XACML) Version 2.0*. OASIS Standard, Februar 2005. [Online] Verfügbar: https://www.oasis-open.org/committees/xacml/

[OAS07] Organization for the Advancement of Structured Information Standards (OASIS), *Web Services Business Process Execution Language Version 2.0. OASIS Standard*, 2007. [Online] Verfügbar: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[OAS09a] Organization for the Advancement of Structured Information Standards (OASIS), *WS-BPEL Extension for People (BPEL4People)*. Version 1.1, 2009.

[OAS09b] Organization for the Advancement of Structured Information Standards (OASIS), *Web Services Human Task (WS-HumanTask)*. Version 1.1, 2009.

[ODH+08] C. Ouyang, M. Dumas, A. ter Hofstede, W. van der Aalst. *Pattern-Based Translation of BPMN Process Models to BPEL Web Services*. International Journal of Web Services Research, 5(1):42–62, 2008.

[OMG11a] Object Management Group (OMG), *Unified Modeling Language (UML)*. OMG Document Number formal/2011-08-06, 2011. [Online] Verfügbar: http://www.omg.org/spec/UML/2.4.1

[OMG11b] Object Management Group (OMG), *Business Process Model and Notation (BPMN) Version 2.0*. OMG Document Number formal/2011-01-03, 2011. [Online] Verfügbar: http://www.omg.org/spec/BPMN/2.0/PDF

[Ora10] Oracle, *Java CAPS Business Process Monitor: Custom Monitoring Tools with Java CAPS*. 2010.

[Ora11] Oracle, *MySQL*, 2011. [Online] Verfügbar: http://www.mysql.de/

[OZD09] E. Oberortner, U. Zdun, S. Dustdar, *Tailoring a Model-Driven Quality-of-Service DSL for Various Stakeholders*. In: Workshop on Modeling in Software Engineering (MiSE) at ICSE, 2009.

[PA06] M. Pesic, W. van der Aalst, *A Declarative Approach for Flexible Business Processes Management*. In: Proceedings of Business Process Management Workshops, Springer, 2006.

[Par72] D. Parnas, *On the Criteria to be used in Decomposing Systems into Modules*. Communications of the ACM, 15(12):1053–1058, ACM, 1972.

[PCI10] PCI Security Standards Council, *Data Security Standard (DSS)*. 2010.

[Pet62] C. A. Petri, *Kommunikation mit Automaten*. Dissertation, Institut für Angewandte Mathematik, Bonn, 1962.

[Pet95] M. Petre, *Why Looking Isn't Always Seeing: Readership Skills and Graphical Programming*. In: Communications of the ACM, 38(6):33–44, ACM, 1995.

[PIC06] PICTURE - *Process Identification and Clustering for Transparency in Reorganising Public Administrations*. 2006 - 2009. [Online] Verfügbar: http://www.picture-eu.org/

[Pnu77] A. Pnueli, *The Temporal Logic of Programs*. In: Proceedings of the 18th Annual Symposium on Foundations of Computer Science, IEEE, 1977.

[Pol12] A. Polyvyanyy, *Structuring Process Models*. Dissertation, Universität Potsdam, 2012.

[Por85] M. E. Porter, *Competitive Advantage*. Free Press, New York, 1985.

[Pos10a] PostgreSQL Global Development Group, *PostgreSQL*, 2010. [Online] Verfügbar: http://www.postgresql.org/

[Pos10b] A. Poszlovszki, *Process Fragment Recognition and Emphasis*. Diplomarbeit Nr. 3001, Universität Stuttgart, 2010.

[PSW08] A. Polyvyanyy, S. Smirnov, M. Weske, *Process Model Abstraction: A Slider Approach*. In: Proceedings of the 12th IEEE Enterprise Distributed Object Computing Conference (EDOC), IEEE Computer Society, 2008.

[PSW10] A. Polyvyanyy, S. Smirnov, M. Weske. *Business Process Model Abstraction.* In: International Handbook on Business Process Management, Springer, 2010.

[Räc09] M. Räckers, *Domänenspezifisches Prozessmanagement - Konstruktion und Evaluation einer Prozessmodellierungs- und -analysemethode für die öffentliche Verwaltung*. Dissertation, Wirtschaftswissenschaftliche Fakultät der Westfälischen Wilhelms-Universität Münster, 2009.

[Rap11] A. Rapoports, *Configurable Visual Representations for Business Process Monitoring.* Diplomarbeit Nr. 3181, Universität Stuttgart, 2011.

[RBR+06] S. Rinderle, R. Bobrik, M. Reichert, T. Bauer, *Business Process Visualization – Use Cases, Challenges, Solutions*. In: Proceedings of the International Conference on Enterprise Information Systems (ICEIS), INSTICC Press, 2006.

[RBR+10] M. Reichert, S. Bassil, R. Bobrik, T. Bauer, *The Proviado Access Control Model for Business Process Monitoring Components*. In: Enterprise Modelling and Information Systems Architectures - An International Journal. Gesellschaft für Informatik (GI), 2010.

[RFA12] E. Ramezani, D. Fahland, W. van der Aalst, *Where did I misbehave? Diagnostic Information in Compliance Checking*. In: Proceedings of the 10th International Conference on Business Process Management (BPM'12), Springer, 2012.

[RFK+00] C. Rupprecht, M. Fünffinger, H. Knublauch, T. Rose, *Capture and Dissemination of Experience about the Construction of Engineering Processes*. In: Proceedings of the 12th International Conference on Advanced Information Systems Engineering (CAiSE'00), Springer, 2000.

[RHW+11] M. La Rosa, A. ter Hofstede, P. Wohed, H. Reijers, J. Mendling, W. van der Aalst, *Managing Process Model Complexity via Concrete Syntax Modifications*. In: IEEE Transactions on Industrial Informatics, 7(2):255–265, IEEE, 2011.

[RP93] C. Rolland, N. Prakash, *Reusable Process Chunks*. In: Proceedings of the 4th International Conference Database and Expert Systems Applications (DEXA'93), Springer, 1993.

[RRA+11] M. La Rosa, H. Reijers, W. van der Aalst, R. Dijkman, J. Mendling, M. Dumas, L. Garcia-Banuelos, *Apromore: An Advanced Process Model Repository*. In: Expert Systems with Applications (ESWA), 38(6):7029–7040, 2011.

[RRF+01] C. Rupprecht, T. Rose, M. Fünffinger, H. Schott, A. Sieper, C. Schlick, M. Mühlfelder, *Management von Prozesswissen in Fahrzeugentwicklungsprojekten*. In: Professionelles Wissensmanagement (WM'01), Baden-Baden, 2001.

[RWM+11] M. La Rosa, P. Wohed, J. Mendling, A. ter Hofstede, H. Reijers, W. van der Aalst, *Managing Process Model Complexity Via Abstract Syntax Modifications*. In: IEEE Transactions on Industrial Informatics, 7(4):614–629, IEEE, 2011.

[RZ12] A. Rensink, E. Zambon, *Pattern-Based Graph Abstraction*. In: Proceedings of the 6th International Conference on Graph Transformation (ICGT 2012), Springer, 2012.

[SAD+09] D. Schumm, T. Anstett, D. Dentsas, M. Hahn, M. Sonntag, *Fragmento*, 2009-2012. [Online] Verfügbar: http://www.iaas.uni-stuttgart.de/forschung/projects/fragmento/start.htm

[SAL+09] D. Schleicher, T. Anstett, F. Leymann, R. Mietzner, *Maintaining Compliance in Customizable Process Models*. In: Proceedings of the 17th International Conference on Cooperative Information Systems (CoopIS 2009), Springer, 2009.

[SAL+10a] D. Schumm, T. Anstett, F. Leymann, D. Schleicher, S. Strauch, *Essential Aspects of Compliance Management with Focus on Business Process Automation*. In: Proceedings of the 3rd International Conference on Business Process and Services Computing (BPSC), Gesellschaft für Informatik (GI), 2010.

[SAL+10b] D. Schumm, T. Anstett, F. Leymann, D. Schleicher, *Applicability of Process Viewing Patterns in Business Process Management*. In: Proceedings of the International Workshop on Models and Model-driven Methods for Service Engineering (3M4SE), IEEE Computer Society, 2010.

[SBL+12] D. Schumm, J. Barzen, F. Leymann, L. Ellrich, *A Pattern Language for Costumes in Films*. In: Proceedings of the 17th European Conference on Pattern Languages of Programs (EuroPLoP 2012), ACM, 2012.

[SCF+11] D. Schumm, J. Cai, C. Fehling, D. Karastoyanova, F. Leymann, M. Weidmann, *Composite Process View Transformation*. In: Proceedings of the 12th International Conference on Electronic Commerce and Web Technologies (EC-Web), Springer, 2011.

[Sch10] D. Schumm, *Process Views and their Application in Compliance Management*. In: The 4th Summer School on Service Oriented Computing, Poster, 2010.

[Sch11a] D. Schumm, *Fragmento – Documentation*. 2011. [Online] Verfügbar: http://www.iaas.uni-stuttgart.de/forschung/projects/fragmento/

[Sch11b] D. Schumm, *Information Design for Business Process Management*. In: The 5th Summer School on Service Oriented Computing, Poster, 2011.

[Sch14] D. Schleicher, *Modellierung regelkonformer Geschäftsprozesse*. Dissertation, Universität Stuttgart, 2014.

[SDH+12] D. Schumm, D. Dentsas, M. Hahn, D. Karastoyanova, F. Leymann, M. Sonntag, *Web Service Composition Reuse through Shared Process Fragment*

Libraries. In: Demo Proceedings of the 12th International Conference on Web Engineering (ICWE), Springer, 2012.

[SDK+95] M. Shaw, R. DeLine, D. Klein, T. Ross, D. Young, G. Zelesnik, *Abstractions for Software Architecture and Tools to Support them*. In: IEEE Transactions on Software Engineering, 21(4):314–335, IEEE, 1995.

[Sel11] B. Selic, *A Short Catalogue of Abstraction Patterns for Model-Based Software Engineering*. In: International Journal of Software and Informatics, 5(1-2 Part2):313–334, 2011.

[SFG+11] D. Schleicher, C. Fehling, S. Grohe, F. Leymann, A. Nowak, P. Schneider, D. Schumm, *Compliance Domains: A Means to Model Data-Restrictions in Cloud Environments*. In: Proceedings of the 15th IEEE Enterprise Distributed Object Computing Conference (EDOC), IEEE, 2011.

[SFK+12] D. Schumm, C. Fehling, D. Karastoyanova, F. Leymann, J. Rütschlin, *Processes for Human Integration in Automated Cloud Application Management*. Technischer Bericht Nr. 2012/02, Universität Stuttgart, 2012.

[SFW11] S. Smirnov, A. Farahni, M. Weske, *State Propagation in Abstracted Business Processes*. Technischer Bericht Nr. 47 des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam, Universitätsverlag Potsdam, Juli 2011.

[SGK+11] M. Sonntag, K. Görlach, D. Karastoyanova, F. Leymann, P. Malets, D. Schumm, *Views on Scientific Workflows*. In: Proceedings of the 10th International Conference on Perspectives in Business Informatics Research (BIR), Springer, 2011.

[SGL+11] D. Schleicher, S. Grohe, F. Leymann, P. Schneider, D. Schumm, T. Wolf, *An Approach to Combine Data-Related and Control Flow-Related Compliance Rules*. In: Proceedings of Service Oriented Computing and Applications (SOCA), IEEE Computer Society, 2011.

[SGN07] S. Sadiq, G. Governatori, K. Namiri, *Modeling Control Objectives for Business Process Compliance*. In: Proceedings of the 5th International Conference on Business Process Management (BPM'07), Springer, 2007.

[Sig13] Signavio GmbH, *Signavio Process Editor QuickModel*. 2013. [Online] Verfügbar: http://www.signavio.com/de/products/signavio-process-editor/quickmodel-de/

[Sil09] B. Silver, *BPMN Method and Style: A Levels-based Methodology for BPM Process Modeling and Improvement using BPMN 2.0*. Cody-Cassidy Press, 2009.

[SK08] S. Sackmann, M. Kähmer, *ExPDT: A Policy-based Approach for Automating Compliance*. In: Wirtschaftsinformatik (WI), 50(5):366–374, Gabler, 2008.

[SK11] D. Schumm, D. Karastoyanova, *Integrating Humans in Scientific Workflows: Integrate, Register & Communicate*. In: The 4th SimTech Status Seminar, Poster, 2011.

[SKK+11] D. Schumm, D. Karastoyanova, O. Kopp, F. Leymann, M. Sonntag, S. Strauch, *Process Fragment Libraries for Easier and Faster Development of Process-based Applications*. In: Journal of Systems Integration. 2(1):39–55, Online, 2011.

[SKL+09] D. Schumm, D. Karastoyanova, F. Leymann, J. Nitzsche, *On Visualizing and Modelling BPEL with BPMN*. In: Proceedings of the 4th International Workshop on Workflow Management (IWWM), IEEE Computer Society, 2009.

[SKL+10] D. Schumm, D. Karastoyanova, F. Leymann, S. Strauch, *Fragmento: Advanced Process Fragment Library*. In: Proceedings of the 19th International Conference on Information Systems Development (ISD), Springer, 2010.

[SKL+11] D. Schumm, D. Karastoyanova, F. Leymann, S. Lie, *Propagation of States from BPEL Process Instances to Chevron Models*. Technischer Bericht Nr. 2011/06, Universität Stuttgart, 2011.

[SL09] T. Scheibler, F. Leymann, *From Modelling to Execution of Enterprise Integration Scenarios: the GENIUS Tool*. In: Proceedings of the 16th Conference on Communication in Distributed Systems (KiVS'09), 2009.

[SL10] D. Schumm, F. Leymann, *Schwerpunktbeitrag Geschäftsprozesstechnologie*. In: Gabler Wirtschaftslexikon, Springer Gabler, 2010. [Online] Verfügbar: http://wirtschaftslexikon.gabler.de/Archiv/569797/geschaeftsprozesstechnologie-v8.html

[SLL11] D. Schumm, G. Latuske, F. Leymann, *A Prototype for View-based Monitoring of BPEL Processes*. Technischer Bericht Nr. 2011/04, Universität Stuttgart, 2011.

[SLL+11] D. Schumm, G. Latuske, F. Leymann, R. Mietzner, T. Scheibler, *State Propagation for Business Process Monitoring on Different Levels of Abstraction*. In: Proceedings of the 19th European Conference on Information Systems (ECIS), AIS Library, 2011.

[SLM+10] D. Schumm, F. Leymann, Z. Ma, T. Scheibler, S. Strauch, *Integrating Compliance into Business Processes: Process Fragments as Reusable Compliance Controls*. In: Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI), Universitätsverlag Göttingen, 2010.

[SLS10a] D. Schumm, F. Leymann, A. Streule, *Process Viewing Patterns*. In: Proceedings of the 14th IEEE Enterprise Distributed Object Computing Conference (EDOC), IEEE Computer Society, 2010.

[SLS10b] D. Schumm, F. Leymann, A. Streule, *Process Views to Support Compliance Management in Business Processes*. In: Proceedings of the 11th International Conference on Electronic Commerce and Web Technologies (EC-Web), Springer, 2010.

[Smi12] S. Smirnov, *Business Process Model Abstraction*. Dissertation, Hasso Plattner Institut, Universität Potsdam, 2012.

[SML09] T. Scheibler, R. Mietzner, F. Leymann, *EMod: Platform Independent Modelling, Description and Enactment of Parameterisable EAI Patterns*. In: Enterprise Information Systems, 3(3):299–317, Taylor & Francis, 2009.

[SO00] W. Sadiq, M. Orlowska, *Analyzing Process Models using Graph Reduction Techniques.* In: Information Systems, 25(2):117–134, Elsevier, 2000.

[Sof11] Software AG, *ARIS Business Architect Version 7.2 - Schnelleinstieg*. Product Documentation, Oktober 2011.

[Spa13] Sparx Systems, *Enterprise Architect 10*. 2013. [Online] Verfügbar: http://www.sparxsystems.com.au/products/ea/index.html

[Spr11] SpringSource, *Spring Framework*. 2011. [Online] Verfügbar: http://projects.spring.io/spring-framework/

[SRC+09] P. Silveira, C. Rodríguez, F. Casati, F. Daniel, V. D'Andrea, C. Worledge, Z. Taheri, *On the Design of Compliance Governance Dashboards for Effective Compliance and Audit Management*. In: Proceedings of the Non Functional Properties and Service Level Agreements Management in Service Oriented Computing Workshop (NFPSLAM-SOC2009), Springer, 2009.

[SRH+06] S. Seidel, M. Rosemann, A. ter Hofstede, L. Bradford, *Developing a Business Process Reference Model for the Screen Business – A Design Science Research Case Study*. In: Proceedings of the 17th Australasian Conference on Information Systems (ACIS'06), 2006.

[SSO01] S. Sadiq, W. Sadiq, M. Orlowska, *Pockets of Flexibility in Workflow Specification*. In: Proceedings of the 20th International Conference on Conceptual Modeling, Springer, 2001.

[STK+10] D. Schumm, O. Türetken, N. Kokash, A. Elgammal, F. Leymann, W.-J. van den Heuvel, *Business Process Compliance through Reusable Units of Compliant Processes*. In: Proceedings of the 1st Workshop on Engineering SOA and the Web (ESW), Springer, 2010.

[Str09] A. Streule, *Abstract Views on BPEL Processes.* Diplomarbeit Nr. 2889, Universität Stuttgart, 2009.

[Stu10] Studienprojekt A Softwaretechnik Universität Stuttgart, *decidR*. 2010. [Online] Verfügbar: http://www.decidr.eu

[Sun12] L. Sun, *Web Services for Human Interaction*. Diplomarbeit Nr. 3275, Universität Stuttgart, 2012.

[SWL+10] D. Schleicher, M. Weidmann, F. Leymann, D. Schumm, *Compliance Scopes: Extending the BPMN 2.0 Meta Model to Specify Compliance Requirements*. In: Proceedings of Service Oriented Computing and Applications (SOCA), IEEE Computer Society, 2010.

[TBR00] The Business Rules Group, *Defining Business Rules - What Are They Really?* GUIDE Business Rules Project Final Report, 2000.

[TEH+12] O. Türetken, A. Elgammal, W.-J. van den Heuvel, M. Papazoglou, *Capturing Compliance Requirements: A Pattern-Based Approach*. In: IEEE Software, Mai/Juni 2012, 28–36, IEEE, 2012.

[Thö12] M. Thöne, *Die Symbolik der Farben*. 2012. [Online] Verfügbar: http://www.mara-thoene.de/html/farbensymbolik.html

[TRI09] L. Thom, M. Reichert, C. Iochpe, *Activity Patterns in Process-aware Information Systems: Basic Concepts and Empirical Evidence*. In: International Journal of Business Process Integration and Management (IJBPIM), 2009.

[TRK+08] G. Thieberger, M. Rosenfeld, M. Karasik, K. Rotberg, *Methods for Utilizing User Emotional State in a Business Process*. United States Patent Application Publication US 2008/0091515 A1, April 2008.

[TZD07] H. Tran, U. Zdun, S. Dustdar, *View-based and Model-driven Approach for Reducing the Development Complexity in Process-driven SOA*. In: Proceedings of the International Working Conference on Business Process and Services Computing (BPSC), Gesellschaft für Informatik (GI), 2007.

[TZD11] H. Tran, U. Zdun, S. Dustdar, *Name-based View Integration for Enhancing the Reusability in Process-driven SOAs*. In: International Journal of Business Process Integration and Management, 5(3):229–239, InderScience, 2011.

[Var01] M. Vardi, *Branching vs. Linear Time: Final Showdown*. In: Proceedings of the 2001 Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2001.

[Ver10] Vereofy-Group TU-Dresden, *Vereofy Model Checker*. 2010. [Online] Verfügbar: http://www.vereofy.de/

[VVL07] J. Vanhatalo, H. Völzer, F. Leymann. *Faster and more focused Control-flow Analysis for Business Process Models though SESE Decomposition*. In: Proceedings of the 5th International Conference on Service-Oriented Computing (ICSOC), Springer, 2007.

[W3C01] World Wide Web Consortium (W3C), *Web Services Description Language (WSDL) 1.1*. W3C Note 15 March 2001. [Online] Verfügbar: http://www.w3.org/TR/wsdl

[W3C03] World Wide Web Consortium (W3C), *SOAP Version 1.2 Part 1: Messaging Framework*. W3C Proposed Recommendation 07 May 2003. [Online] Verfügbar: http://www.w3.org/TR/2003/PR-soap12-part1-20030507/

[W3C05] World Wide Web Consortium (W3C), *Scalable Vector Graphics (SVG) Full 1.2 Specification*. 2005. [Online] Verfügbar: http://www.w3.org/TR/SVG12/

[W3C13] World Wide Web Consortium (W3C), *Cascading Style Sheets*. 2013. [Online] Verfügbar: http://www.w3.org/Style/CSS/

[WAK+11] M. Weidmann, M. Alvi, F. Kötter, F. Leymann, R. Renner, D. Schumm, *Business Process Change Management based on Process Model Synchronization of Multiple Abstraction Levels*. In: Proceedings of Service Oriented Computing and Applications (SOCA), IEEE Computer Society, 2011.

[WDG+08] M. Weidlich, G. Decker, A. Großkopf, M. Weske, *BPEL to BPMN: The Myth of a Straight-Forward Mapping*. In: Proceedings of the 16th International Conference on Cooperative Information Systems, Springer, 2008.

[Wes07] M. Weske, *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007.

[WFM95] Workflow Management Coalition, *The Workflow Reference Model*. Document Number TC00-1003, 1995. [Online] Verfügbar: http://www.wfmc.org/standards/docs/tc003v11.pdf

[WFM99] Workflow Management Coalition, *Terminology & Glossary*. Document Number WFMC-TC-1011, 1999.

[Wil07] S. Wilkens, *Effizientes Nachhaltigkeitsmanagement*. Deutscher Universitätsverlag / Gabler, 2007.

[WR08] B. Weber, M. Reichert, *Refactoring Process Models in Large Process Repositories*. In: Proceedings 20th International Conference on Advanced Information Systems Engineering (CAiSE'08), Springer, 2008.

[WRR08] B. Weber, M. Reichert, S. Rinderle-Ma, *Change Patterns and Change Support Features - Enhancing Flexibility in Process-aware Information Systems*. In: Data & Knowledge Engineering, 66(3):438–466, Elsevier, 2008.

[WS07] C. Wolter, A. Schaad, *Modeling of Task-Based Authorization Constraints in BPMN*. In: Proceedings of the 5th International Conference on Business Process Management (BPM'07), Springer, 2007.

[WSL09] B. Wetzstein, S. Strauch, F. Leymann, *Measuring Performance Metrics of WS-BPEL Service Compositions*. In: Proceedings of the 5th International Conference on Networking and Services, IEEE Computer Society, 2009.

[WSO12] WSO2, *WSO2 Business Process Server*. 2012. [Online] Verfügbar: http://wso2.com/products/business-process-server/

[YMH+06] J. Yu, T. Manh, J. Han, Y. Jin, Y. Han, J. Wang, *Pattern-Based Property Specification and Verification for Service Composition*. In: Proceedings of the 7th International Conference on Web Information Systems Engineering (WISE'06), Springer, 2006.

[Zen10] Z. Zeng, *Anwendungsspezifische Prozessvisualisierung*. Diplomarbeit Nr. 2967, Universität Stuttgart, 2010.

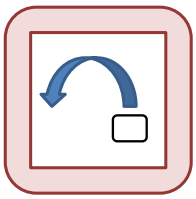
[Zim95] W. Zimmer, *Relationships between Design Patterns*. In: Pattern Languages of Program Design, Addison-Wesley, 1995.

[ZLS11] S. Zor, F. Leymann, D. Schumm, *A Proposal of BPMN Extensions for the Manufacturing Domain*. In: Proceedings of the 44th CIRP Conference on Manufacturing Systems (ICMS), 2011.

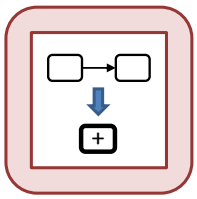
[ZLS+09] X. Zhao, C. Liu, W. Sadiq, M. Kowalkiewicz, S. Yongchareon, *WS-BPEL Business Process Abstraction and Concretisation*. In: Proceedings of the 14th International Conference on Database Systems for Advanced Applications, Springer, 2009.

14.2 Kompaktdarstellung des Musterkatalogs

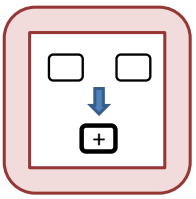
Muster zur Strukturveränderung (Reduzierung)



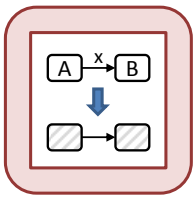
Aktivitäten entfernen (M1)



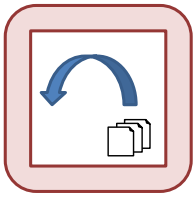
Verbundene Aktivitäten aggregieren (M2)



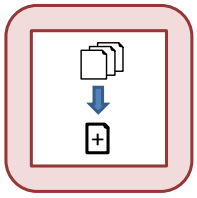
Unverbundene Aktivitäten aggregieren (M3)



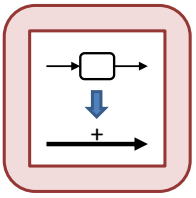
Eigenschaften abstrahieren (M4)



Datenelemente entfernen (M5)



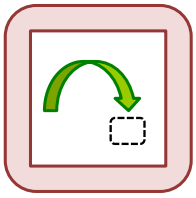
Datenelemente aggregieren (M6)



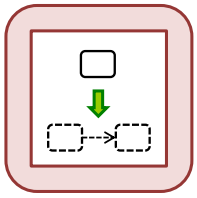
Pfade aggregieren (M7)

Strukturen aufräumen (M8)

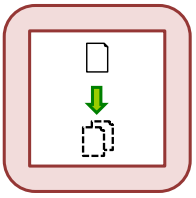
Muster zur Strukturveränderung (Erweiterung)



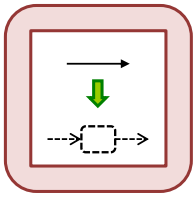
Aktivitäten einfügen (M9)



Aktivitäten zerlegen (M10)



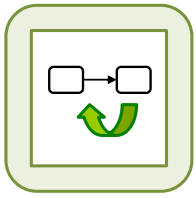
Datenelemente zerlegen (M11)



Pfade zerlegen (M12)

Muster zur Informationsanreicherung

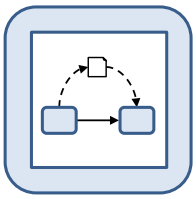
Nutzergestützt anreichern (M13)



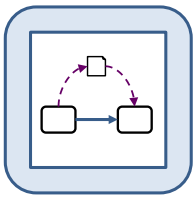
Statisch anreichern (M14)

Dynamisch anreichern (M15)

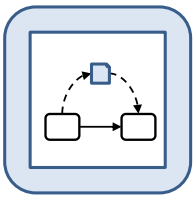
Muster zur Darstellungsanpassung (Erscheinungsbild)



Erscheinungsbild von Aktivitäten anpassen (M16)

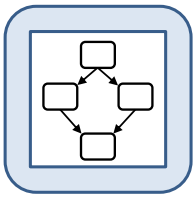


Erscheinungsbild von Verbindungen anpassen (M17)

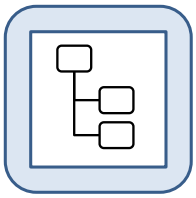


Erscheinungsbild von Datenelementen anpassen (M18)

Muster zur Darstellungsanpassung (Schema)

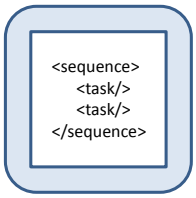


Graphschema verwenden (M19)

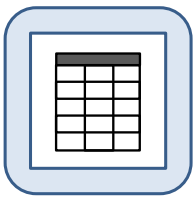


Baumschema verwenden (M20)

Blockschema verwenden (M21)

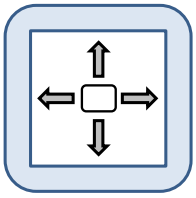


Codeschema verwenden (M22)

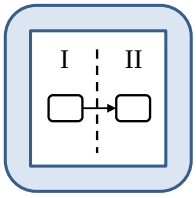


Tabellenschema verwenden (M23)

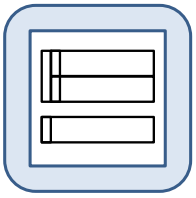
Muster zur Darstellungsanpassung (Layout)



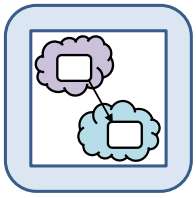
Orientierung festlegen (M24)



Phasenweise auslegen (M25)



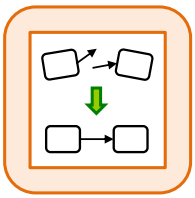
Organisatorisch auslegen (M26)



Räumlich auslegen (M27)

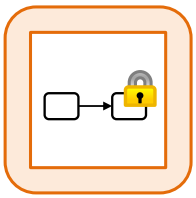
Maßgeschneidert auslegen (M28)

Muster von Vorgaben zur Durchführung von Transformationen



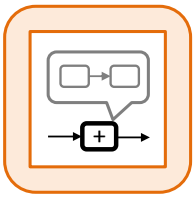
Konsistenz bewahren (M29)

Ausführbarkeit bewahren (M30)



Strukturen bewahren (M31)

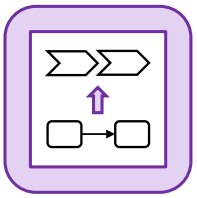
Transformationen protokollieren (M32)



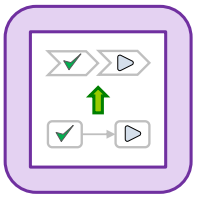
Nachvollziehbarkeit gewährleisten (M33)

14.3 Hilfselemente

Sprachübergreifende Transformationen

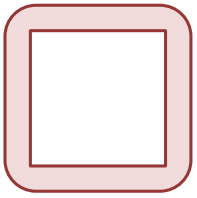


Prozess übersetzen (S1)

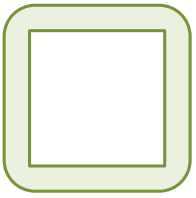


Zustand projizieren (S2)

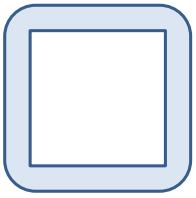
Vorlagen für nutzerdefinierte Transformationen



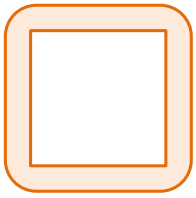
Struktur verändern



Mit Informationen anreichern



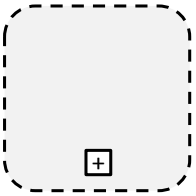
Darstellung anpassen



Vorgabe machen

Weitere Hilfselemente

Gruppierung



Gruppierung (aggregiert)

Platzhalter

14.4 Vorgehensweise zur Erläuterung des musterbasierten Sichtenentwurfs

Musterauswahl. Arbeiten zur Abstraktion (vgl. Abschnitt 2.2) haben vielfach gemein, dass Strukturen weggelassen (M1) oder aggregiert werden (M2). Die Anpassung des Erscheinungsbildes stellt ebenfalls eine gängige Technik dar (M16). Um das Wichtige vom Unwichtigen zu unterscheiden, können neben den im Prozess enthaltenen Informationen individuelle Kenntnisse und Einschätzungen herangezogen werden (M13). Neben diesen elementaren Mustern wurden in den Sichtenentwürfen in der vorliegenden Dissertation häufig dynamische Anreicherung (M15), Bewahrung von Konsistenz (M29) und Aufräumen (M8) eingesetzt. Diese Musterauswahl ist in Abbildung 14.1 zusammengefasst.

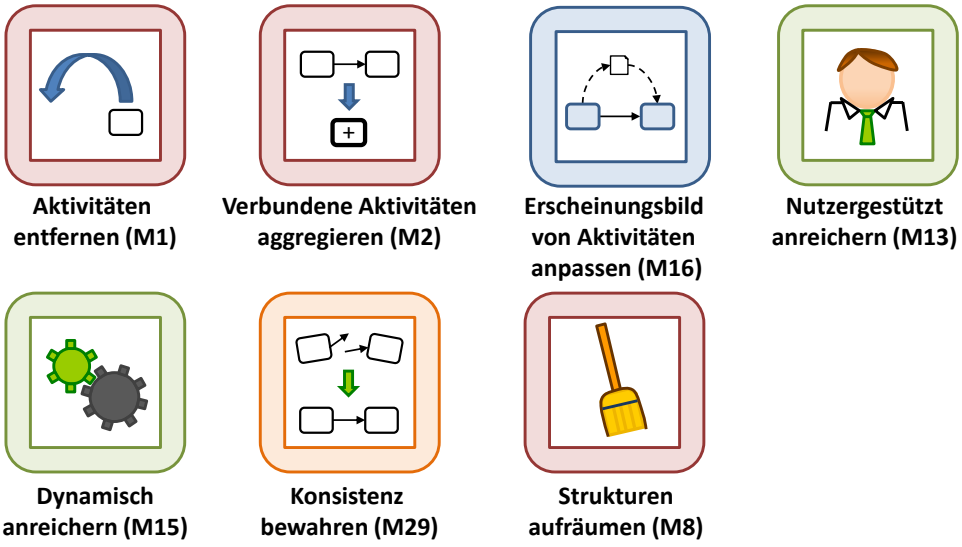


Abbildung 14.1: Auswahl von Mustern zur Validierung des musterbasierten Sichtenentwurfs.

Vorstellung der Wirkungsweise. Anhand des Kreditvergabeprozesses (Abbildung 11.1) werden die Wirkungsweisen der ausgewählten Muster praktisch und an konkreten Beispielen erläutert. Die Muster werden den Probanden gedruckt und ausgeschnitten zur Verfügung gestellt. Sichtenentwürfe können dadurch auf einem Blatt Papier auf einfache Weise gelegt, beschrieben und geändert werden. Nach der Erläuterung der Wirkung der einzelnen Muster werden die unterschiedlichen Farbcodierungen erklärt. Nach diesem Schritt sollte ein Proband in der Lage sein, die ausgewählten Muster, deren Wirkungsweise und die unterschiedlichen Musterkategorien in eigenen Worten zu beschreiben.

Vorstellung der Musterkomposition. Als Vorgehensweise zur Ordnung einer Musterkomposition wird vorgeschlagen, zuerst Muster zur Vorgabe und Informationsanreicherung zu verwenden, anschließend strukturverändernde Muster zu nutzen und am Schluss Muster zur Darstellungsanpassung einzusetzen. Unter Verwendung der ausgewählten Muster kann ein einfacher Sichtenentwurf diskutiert werden, der in Abbildung 14.2 dargestellt ist. Dieser Entwurf dient dazu, ein Verständnis für die Anwendung von Mustern und deren Zusammenwirken in einer Komposition zu vermitteln. Nach diesem Schritt sollte ein Proband in der Lage sein, den Sichtenentwurf in eigenen Worten zu beschreiben.

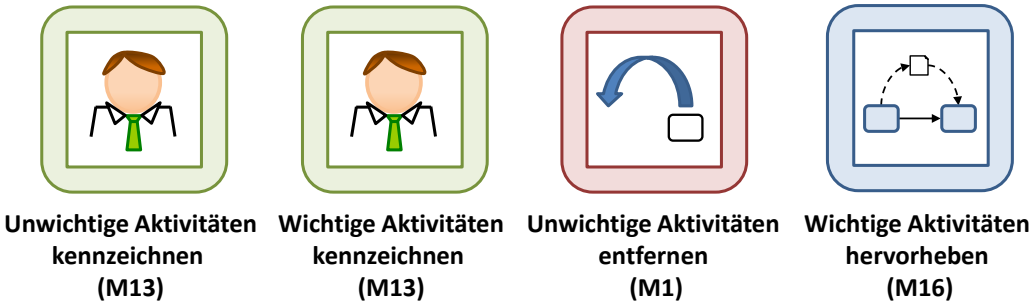


Abbildung 14.2: Einfacher Sichtenentwurf auf Basis der ausgewählten Muster.

Anpassung einer Musterkomposition. Der gezeigte Sichtenentwurf suggeriert, dass der Prozess beim Entfernen unwichtiger Aktivitäten zusammengehalten wird. Um dieses Verhalten explizit in den Entwurf aufzunehmen, wird die Komposition dahingehend angepasst, dass vor dem Entfernen unwichtiger Aktivitäten (M1) die Anwendung des Musters Konsistenz bewahren (M29) erfolgt. Des Weiteren kann der Prozess durch eine Aggregation von Strukturen (M2) weiter vereinfacht werden. Das Aufräumen (M8) kann dazu eingesetzt werden, um überflüssige Kontrollverbindungen und Datenelemente zu entfernen. Nun sollte ein Proband in der Lage sein, einen Sichtenentwurf nach eigenem Ermessen zu verändern.

Eigener Sichtenentwurf. Auf Basis von Anforderungen soll ein Proband anschließend eine eigene Sicht entwerfen. Die Sicht soll es einem Sachbearbeiter einer Bank ermöglichen, den Ablauf eines früheren Kreditvergabefalles eines Kunden nachzuvollziehen zu können. Der Kunde hatte sich über die lange Bearbeitungsdauer beschwert und der Sachbearbeiter hat nun die Aufgabe, den Grund für die lange Dauer zu identifizieren. Ein beispielhafter Sichtenentwurf für dieses Szenario ist in Abbildung 14.3 dargestellt, wobei die Aktivitätsdauer bspw. durch eine Farbcodierung des Rahmens von Aktivitäten dargestellt werden könnte.

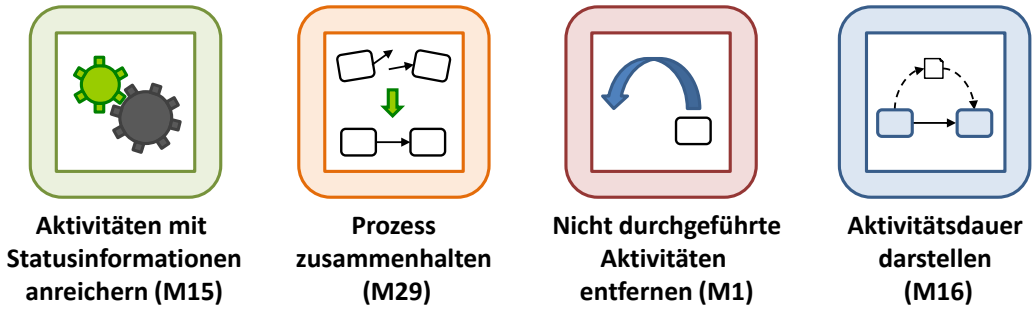


Abbildung 14.3: Beispielhafte Lösungsmöglichkeit für den geforderten Sichtenentwurf.

Überblick Musterkatalog und Sichtenentwürfe. Ein Überblick über die verschiedenen Muster im Musterkatalog mit den zugehörigen Beschreibungen und Skizzen zeigt dem Proband die Möglichkeiten des Sichtentwurfs auf. Der Ansatz wird erläutert, jeweils nur einen bestimmten Teil der Muster für den Entwurf zur Verfügung zu stellen, bspw. abhängig von der Aufgabenstellung oder den gegebenen technischen Möglichkeiten. Anschließend werden verschiedene Sichtenentwürfe vorgestellt und die Konzepte zur Wiederverwendung von Entwürfen erläutert. Abschließend sollte der Proband einen persönlichen Eindruck vom musterbasierten Sichtenentwurf haben.

14.5 Abbildungsverzeichnis

Abbildung 1.1: Beispielhafte Prozesssicht und deren schrittweise Verfeinerung, vgl. [SCF+11]. 20

Abbildung 2.1: Infrastruktur zur Automatisierung von Geschäftsprozessen, vgl. [SL10]. 28

Abbildung 2.2: Phasenverständnis des Lebenszyklus von Geschäftsprozessen, modelliert nach [SLS10b]. 30

Abbildung 2.3: Aktivitätszustandsmodell von PI-Graphen, vgl. [KKS+06]. 33

Abbildung 2.4: Prozessinstanzzustandsmodell von PI-Graphen, vgl. [KKS+06]. 33

Abbildung 2.5: Grafische Notation zur Darstellung von PM-Graphen, orientiert an BPMN [OMG11b]. 34

Abbildung 2.6: Aktivitätszustandssymbole für die Darstellung von PI-Graphen. 34

Abbildung 2.7: Beispiel eines PI-Graphen. 34

Abbildung 2.8: Grundlagen von Sichten auf Datenbanken, modelliert nach [AHV95]. 38

Abbildung 2.9: Abstraktionsebenen im Stand der Technik. 46

Abbildung 3.1: Vorgehensweise zum musterbasierten Entwurf und der dienstorientierten Implementierung von Sichten auf Geschäftsprozesse. 65

Abbildung 3.2: Konzeptionelles Modell zum musterbasieren Entwurf von Sichten auf Geschäftsprozesse. 66

Abbildung 3.3: Begriffe im musterbasieren Entwurf von Sichten auf Geschäftsprozesse. 67

Abbildung 3.4: Elemente zur Veranschaulichung der Anwendung von Transformationen, vgl. [SLS10a]. 68

Abbildung 3.5: Elemente der grafischen Notation zur Darstellung von Prozessskizzen (a) und Erweiterungen zur Unterscheidung von ursprünglichen und transformierten Elementen (b), vgl. [SLS10a]. 68

Abbildung 3.6: Beispiel der Transformation eines PM-Graphen in einen erweiterten PM-Graphen. 69

Abbildung 3.7: Aktivitätszustände (a) im PI-Graphen, (b) sichtenbezogene Erweiterungen der Aktivitätszustände und (c) sichtenbezogene Erweiterungen der Kontrollverbindungszustände. 69

Abbildung 3.8: Beispiel eines erweiterten PI-Graphen. 70

Abbildung 3.9: Unterscheidung der Musterkategorien durch Farbcodierung der Rahmenfarbe der Symbole. 71

Abbildung 3.10: Informationszusammensetzung in den Symbolen der Sichtenentwurfsmuster. 71

Abbildung 3.11: Kategorisierung und Untergliederung der Sichtenentwurfsmuster im Musterkatalog. 73

Abbildung 3.12: Übersicht des Musterkatalogs nach dem Verständnis eines Fließbandes, auf dem ein Eingabeprozess an verschiedenen Stationen bearbeitet werden kann und schrittweise eine Prozesssicht angefertigt wird. 74

Abbildung 3.13: Symbol zu „Aktivitäten entfernen (M1)“. 75

Abbildung 3.14: Skizze zu „Aktivitäten entfernen (M1)“. 75

Abbildung 3.15: Beispiel für eine Mehrdeutigkeit beim Entfernen von Aktivitäten. 76

Abbildung 3.16: Symbol zu „Verbundene Aktivitäten aggregieren (M2)“. 77

Abbildung 3.17: Skizze zu „Verbundene Aktivitäten aggregieren (M2)“. 77

Abbildung 3.18: Beispiel für die Entstehung von Zyklen bei der Aggregation von Aktivitäten. 78

Abbildung 3.19: Symbol zu „Unverbundene Aktivitäten aggregieren (M3)“. 78

Abbildung 3.20: Skizze zu „Unverbundene Aktivitäten aggregieren (M3)“. 79

Abbildung 3.21: Symbol zu „Eigenschaften abstrahieren (M4)“. 79

Abbildung 3.22: Skizze zu „Eigenschaften abstrahieren (M4)“ mit Eingabeprozess (a), Veranschaulichung der Anwendungsbereiche der mehrfachen Musteranwendung (b) und dem Ausgabeprozess (c). 80

Abbildung 3.23: Feingranulare Abstraktion von Eigenschaften unter Einsatz einer Taxonomiezuordnung (T_1) mit anschließender Aggregation verbundener Aktivitäten mit wertgleichen Eigenschaften (T_2). 81

Abbildung 3.24: Symbol zu „Datenelemente entfernen (M5)“. 81

Abbildung 3.25: Skizze zu „Datenelemente entfernen (M5)“. 81

Abbildung 3.26: Symbol zu „Datenelemente aggregieren (M6)“. 82

Abbildung 3.27: Skizze zu „Datenelemente aggregieren (M6)“ mit (a) einem Eingabeprozess, (b) zwei Anwendungsbereichen für eine mehrfache Musteranwendung und (c) dem entsprechenden Ausgabeprozess. 82

Abbildung 3.28: Symbol zu „Pfade aggregieren (M7)“. 83

Abbildung 3.29: Skizze zu „Pfade aggregieren (M7)“. 83

Abbildung 3.30: Symbol zu „Strukturen aufräumen (M8)“. 84

Abbildung 3.31: Skizze zu „Strukturen aufräumen (M8)“. 84

Abbildung 3.32: Symbol zu „Aktivitäten einfügen (M9)“. 85

Abbildung 3.33: Skizze zu „Aktivitäten einfügen (M9)“. 86

Abbildung 3.34: Symbol zu „Aktivitäten zerlegen (M10)“. 86

Abbildung 3.35: Skizze zu „Aktivitäten zerlegen (M10)“. 86

Abbildung 3.36: Symbol zu „Datenelemente zerlegen (M11)“. 87

Abbildung 3.37: Skizze zu „Datenelemente zerlegen (M11)“. 87

Abbildung 3.38: Symbol zu „Pfade zerlegen (M12)“. 88

Abbildung 3.39: Skizze zu „Pfade zerlegen (M12)“. 88

Abbildung 3.40: Symbol zu „Nutzergestützt anreichern (M13)“. 89

Abbildung 3.41: Skizze zu „Nutzergestützt anreichern (M13)“ für einen Eingabeprozess (a) mit Anwendung auf den gesamten Prozess (b) zur Markierung von Favoriten eines bestimmten Nutzers (c). 89

Abbildung 3.42: Symbol zu „Statisch anreichern (M14)“. 90

Abbildung 3.43: Skizze zu „Statisch anreichern (M14)“. 90

Abbildung 3.44: Symbol zu „Dynamisch anreichern (M15)“. 91

Abbildung 3.45: Skizze zu „Dynamisch anreichern (M15)“ mit einer Anreicherung von Aktivitätszuständen und Werten von Datenelementen. 92

Abbildung 3.46: Symbol zu „Erscheinungsbild von Aktivitäten anpassen (M16)“. 93

Abbildung 3.47: Stand der Technik in der Darstellung von Aktivitäten, vgl. [SLS10b]. 93

Abbildung 3.48: Skizze zu „Erscheinungsbild von Aktivitäten anpassen (M16)“. 94

Abbildung 3.49: Symbol zu „Erscheinungsbild von Verbindungen anpassen (M17)“. 94

Abbildung 3.50: Stand der Technik in der Darstellung von Verbindungen. 95

Abbildung 3.51: Skizze zu „Erscheinungsbild von Verbindungen anpassen (M17)“. 95

Abbildung 3.52: Symbol zu „Erscheinungsbild von Datenelementen anpassen (M18)“. 95

Abbildung 3.53: Skizze zu „Erscheinungsbild von Datenelementen anpassen (M18)“. 96

Abbildung 3.54: Symbol zu „Graphschema verwenden (M19)“. 97

Abbildung 3.55: Skizze zu „Graphschema verwenden (M19)“. 97

Abbildung 3.56: Symbol zu „Baumschema verwenden (M20)“. 97

Abbildung 3.57: Skizze zu „Baumschema verwenden (M20)“. 98

Abbildung 3.58: Symbol zu „Blockschema verwenden (M21)“. 98

Abbildung 3.59: Skizze zu „Blockschema verwenden (M21)“. 99

Abbildung 3.60: Symbol zu „Codeschema verwenden (M22)“. 99

Abbildung 3.61: Skizze zu „Codeschema verwenden (M22)“. 99

Abbildung 3.62: Symbol zu „Tabellenschema verwenden (M23)“. 100

Abbildung 3.63: Skizze zu „Tabellenschema verwenden (M23)“. 100

Abbildung 3.64: Symbol zu „Orientierung festlegen (M24)“. 101

Abbildung 3.65: Skizze zu „Orientierung festlegen (M24)“. 101

Abbildung 3.66: Symbol zu „Phasenweise auslegen (M25)“. 102

Abbildung 3.67: Skizze zu „Phasenweise auslegen (M25)“. 102

Abbildung 3.68: Symbol zu „Organisatorisch auslegen (M26)“. 103

Abbildung 3.69: Skizze zu „Organisatorisch auslegen (M26)“. 103

Abbildung 3.70: Symbol zu „Räumlich auslegen (M27)“. 104

Abbildung 3.71: Skizze zu „Räumlich auslegen (M27)“. 104

Abbildung 3.72: Symbol zu „Maßgeschneidert auslegen (M28)“. 104

Abbildung 3.73: Skizze zu „Maßgeschneidert auslegen (M28)“. 105

Abbildung 3.74: Symbol zu „Konsistenz bewahren (M29)“. 106

Abbildung 3.75: Skizze zu „Konsistenz bewahren (M29)“ mit Entfernen der Aktivität *C*, wobei (b) den Ausgabeprozess unter Einhaltung der Vorgabe und (c) ohne dessen Einhaltung darstellt. 106

Abbildung 3.76: Symbol zu „Ausführbarkeit bewahren (M30)“. 107

Abbildung 3.77: Skizze zu „Ausführbarkeit bewahren (M30)“. 107

Abbildung 3.78: Symbol zu „Strukturen bewahren (M31)“. 108

Abbildung 3.79: Skizze zu „Strukturen bewahren (M31)“. 108

Abbildung 3.80: Symbol zu „Transformationen protokollieren (M32)“. 109

Abbildung 3.81: Skizze zu „Transformationen protokollieren (M32)“. 109

Abbildung 3.82: Symbol zu „Nachvollziehbarkeit gewährleisten (M33)“. 109

Abbildung 3.83: Skizze zu „Nachvollziehbarkeit gewährleisten (M33)“. 110

Abbildung 3.84: Beziehungsgraph der Sichtenentwurfsmuster mit den Beziehungen *gut kombinierbar* (grau), *ähnlich* (grün) und *verwendet* (blau gestrichelt). 111

Abbildung 3.85: Mögliche Erweiterung des Musterkatalogs am Beispiel „Kontrollverbindungen entfernen“. 115

Abbildung 3.86: Unveränderte Merkmale (links) und maximal veränderte Merkmale (rechts). 119

Abbildung 3.87: Beispiel der Mustersequenz eines Grobentwurfs mit einer einfachen Skizze. 122

Abbildung 3.88: Beispiel der Mustersequenz eines verfeinerten Grobentwurfs mit detaillierter Skizze. 124

Abbildung 3.89: Mustergraph zur Darstellung der Abhängigkeiten der Musteranwendungen eines musterbasierten Sichtenentwurfs. 125

Abbildung 3.90: Gruppierung (b) und Aggregation (c) eines Segments (a) zur Komplexitätsreduzierung. 126

Abbildung 3.91: Parametrisierung eines Mustersegments zur Unterstützung der Wiederverwendung. 127

Abbildung 3.92: Beispiel einer Vorlage als Mittel der Vereinfachung und Standardisierung des Entwurfs. 127

Abbildung 3.93: Beispiel einer Entwurfsvorlage mit werkzeugspezifisch eingeschränkten Platzhaltern. 128

Abbildung 3.94: Informationsdesign für Geschäftsprozesse als Schnittstelle zwischen Anforderungen und der Bereitstellung von Informationen im Prozesszusammenhang. 129

Abbildung 3.95: Informationsdesign im Lebenszyklus von Geschäftsprozessen. 130

Abbildung 4.1: Klassifikation des Entwurfs. 131

Abbildung 4.2: Mustergruppierung zur Informationserweiterung. 132

Abbildung 4.3: Mustersequenz zum Entfernen von Details. 132

Abbildung 4.4: Skizze zur Informationserweiterung (b) und dem anschließenden Entfernen von Aktivitäten und Datenelementen mit Relevanz #2 (c). 133

Abbildung 4.5: Klassifikation des Entwurfs. 133

Abbildung 4.6: Mustersequenz zum Zusammenfassen bekannter Sachverhalte. 134

Abbildung 4.7: Skizze zur Annotation bekannter Sachverhalte (b) und deren Zusammenfassung (c). 134

Abbildung 4.8: Klassifikation des Entwurfs. 135

Abbildung 4.9: Eine Größenanpassung stellt die Ausprägung eines bestimmten Aspekts dar. 135

Abbildung 4.10: Mustersequenz zur Hervorhebung bestimmter Aspekte. 136

Abbildung 4.11: Skizze zur Hervorhebung bestimmter Aspekte. 136

Abbildung 4.12: Skizze zur farblichen Hervorhebung eines als wichtig ausgezeichneten Schwerpunkts. 136

Abbildung 4.13: Klassifikation des Entwurfs. 137

Abbildung 4.14: Mustersequenz zum Informationsschutz. 137

Abbildung 4.15: Skizze zum Informationsschutz. 138

Abbildung 4.16: Klassifikation des Entwurfs. 138

Abbildung 4.17: Auflistung von Mustern zur Unterstützung variierender Granularität (Abstraktion) 139

Abbildung 4.18: Auflistung von Mustern zur Unterstützung variierender Granularität (Verfeinerung). 140

Abbildung 4.19: Skizze zur Angleichung der Granularität eines Ursprungsprozesses durch Abstraktion (a) und Verfeinerung (c). 140

Abbildung 4.20: Klassifikation des Entwurfs. 141

Abbildung 4.21: Mustersequenz für eine Sicht auf die Ausführungskonfiguration eines Prozesses 141

Abbildung 4.22: Skizze zur Sicht auf die Ausführungskonfiguration eines Prozesses. 142

Abbildung 4.23: Klassifikation des Entwurfs. 142

Abbildung 4.24: Mustersequenz für eine vereinfachende Sicht auf eine Prozessinstanz. 143

Abbildung 4.25: Skizze für eine vereinfachende Sicht auf eine Prozessinstanz. 143

Abbildung 4.26: Klassifikation des Entwurfs. 143

Abbildung 4.27: Mustersequenz für eine einfache Zeitanalyse. 144

Abbildung 4.28: Skizze für eine einfache Zeitanalyse. 144

Abbildung 5.1: Symbol zu „Prozess übersetzen (S1)“. 148

Abbildung 5.2: Skizze zu „Prozess übersetzten (S1)“. 149

Abbildung 5.3: Symbol zu „Zustand projizieren (S2)“. 149

Abbildung 5.4: Skizze für (a) Zustandsübertragungen von einem PM-Graphen auf ein Prozessmodell in einer anderen Prozesssprache, (b) Zustandsübertragungsregeln und (c) die Durchführung der Zustandsprojektion. 150

Abbildung 5.5: Übersetzung eines Prozessmodells mit vorangehender Vereinfachung. 151

Abbildung 5.6: Skizze zur Übersetzung eines vereinfachten Prozessmodells. 152

Abbildung 5.7: Auf übersetzte Prozessmodelle können weitere Muster angewendet werden. 152

Abbildung 5.8: Skizze zur Anwendung einer Sicht auf einen vereinfachten und übersetzten Eingabeprozess. 153

Abbildung 5.9: Konzeptionelles Modell von Chevron. 154

Abbildung 5.10: Möglichkeiten zur Anpassung des Erscheinungsbildes einer Phase oder Aktivität in Chevron, angepasst nach [SKL+11]. 155

Abbildung 5.11: Darstellung eines Prozesses in Chevron mit drei Phasen, angepasst nach [SKL+11]. 156

Abbildung 5.12: Zustandsmodell für Instanzen, Phasen und Aktivitäten. 156

Abbildung 5.13: Klassifikation des Entwurfs. 157

Abbildung 5.14: Mustersequenz für eine Sprachemulation durch Chevron. 158

Abbildung 5.15: Skizze für eine für eine Sprachemulation durch Chevron. 158

Abbildung 5.16: Klassifikation des Entwurfs. 160

Abbildung 5.17: Erforderliche Erweiterungen für die Sprachemulation einer Prozessinstanz mit Chevron. 160

Abbildung 5.18: Mustersequenz für die Sprachemulation einer Prozessinstanz durch Chevron. 161

Abbildung 5.19: Skizze für eine Sprachemulation einer Prozessinstanz mit Chevron. 162

Abbildung 5.20: Mustersequenz für eine Zustandsprojektion mit beispielhafter Darstellung von Eingabeprozess und Ausgabeprozess. 165

Abbildung 5.21: Zustand direkt übertragen (Z1). 165

Abbildung 5.22: Zustand ändern (Z2). 166

Abbildung 5.23: Zustände kombinieren (Z3). 166

Abbildung 5.24: Zustand ableiten (Z4). 167

Abbildung 5.25: Übergang bilden (Z5). 167

Abbildung 5.26: Zustände sammeln (Z6). 168

Abbildung 6.1: Compliance-Instrumente im Compliance Management-Modell, vgl. [SAL+10a]. 172

Abbildung 6.2: Auswahl einer Option zur Realisierung regelkonformen Verhaltens in einem Prozess. 174

Abbildung 6.3: Lebenszyklus von Geschäftsprozessen mit besonderer Betrachtung von zusätzlichen Phasen für Compliance (in Orange). 175

Abbildung 6.4: Informationsdesign im Lebenszyklus von Geschäftsprozessen (in Blau) mit besonderer Betrachtung von Compliance (in Orange). 175

Abbildung 6.5: Konzeptionelles Modell zur Verknüpfung von Compliance-Quellen, Compliance-Anforderungen, Risiken und Mitteln zu deren Berücksichtigung in Prozessen, angepasst nach [STK+10]. 178

Abbildung 6.6: Auswahl an Symbolen aus BPMN zu Darstellung von rr-Fragmenten mit Erweiterungen für Platzhalter und Constraints, angepasst und erweitert nach [OMG11b] und [SKK+11]. 179

Abbildung 6.7: rr-Fragment für revisionssichere Ablaufdokumentation. 180

Abbildung 6.8: rr-Fragment für die Einbindung eines Compliance-Beauftragten in einen Prozess. 181

Abbildung 7.1: Klassifikation des Entwurfs. 182

Abbildung 7.2: Mustersequenz für das Ausschneiden eines rr-Fragments. 183

Abbildung 7.3: Skizze für ein Ausschneiden von rr-Fragmenten. 183

Abbildung 7.4: Klassifikation des Entwurfs. 184

Abbildung 7.5: Mustersequenz für ein Ausblenden von rr-Fragmenten. 184

Abbildung 7.6: Skizze für ein Ausblenden von rr-Fragmenten. 185

Abbildung 7.7: Klassifikation des Entwurfs. 185

Abbildung 7.8: Mustersequenz zur Hervorhebung von rr-Fragmenten. 186

Abbildung 7.9: Skizze für das Hervorheben von rr-Fragmenten. 186

Abbildung 7.10: Klassifikation des Entwurfs. 187

Abbildung 7.11: Voraussetzung für eine Überprüfung von Transformationen. 187

Abbildung 7.12: Mustersequenz für eine Sicht zur Überprüfung von Transformationen. 188

Abbildung 7.13: Skizze für eine Sicht, deren Transformationen überprüft werden sollen. .. 188

Abbildung 7.14: Skizze für eine Sicht zur Überprüfung von Transformationen. 189

Abbildung 7.15: Klassifikation des Entwurfs. 189

Abbildung 7.16: Mustersequenz für die Analyse von Verantwortlichkeiten. 190

Abbildung 7.17: Skizze für die Analyse von Verantwortlichkeiten. 190

Abbildung 8.1: Vorgehensweise zur Integration von rr-Fragmenten in einen Prozess. 194

Abbildung 8.2: Bestandteile eines rr-Fragments zur Realisierung von Genehmigungsanforderungen, angepasst nach [SAL+10a]. 196

Abbildung 8.3: Ein rr-Fragment (a) wird in einen Geschäftsprozess (b) integriert und dieser Geschäftsprozess somit um regelrealisierendes Verhalten erweitert (c), angepasst nach [SAL+10a]. 197

Abbildung 8.4: Vorgehensweise zur Prüfung der Integration eines rr-Fragments, modelliert nach [STK+10]. 199

Abbildung 8.5: Abstraktes rr-Fragment zur Realisierung von Genehmigungsanforderungen mit den Beziehungen zu den Aussagenvariablen der abstrakten Regel. 203

Abbildung 8.6: Beispielprozess nach Integration und Konkretisierung des rr-Fragments mit den Beziehungen zu den Aussagenvariablen der konkreten Regel. 204

Abbildung 9.1: Eingaben und Ausgaben einer modularen Transformationsfunktion. 209

Abbildung 9.2: Generische Struktur von Transformationsanweisungen, vgl. [SCF+11]..... 210

Abbildung 9.3: Konzeptionelle Kapselung einer Transformationsfunktion als Dienst (a) mit Bezug zu den Elementen einer Schnittstellenbeschreibung in WSDL (b). 211

Abbildung 9.4: Konzeptionelles Modell einer DSL für die grafische Konfiguration einer Prozesssicht, angepasst nach [NKL+12]. 214

Abbildung 9.5: Festlegung des Datenflusses bei einer Komposition von Transformationsschritten durch Dienstaufrufe, angepasst und erweitert nach [SCF+11]. 216

Abbildung 9.6: Struktur eines aus einer Sichtendefinition generierten BPEL-Prozesses zur Komposition von Transformationsdiensten nach [SCF+11] in einer Sprachemulation von BPEL durch BPMN. 218

Abbildung 9.7: Konzeptionelles Modell zur Spezifikation einer Zustandsprojektion. 219

Abbildung 9.8: Grafische Darstellung einer beispielhaften Übertragungsvorlage, um den Aktivitätszustand von einem PI-Graphen (links) zu Chevron (rechts) zu ändern (Anwendung von Übertragungsmuster Z2). 220

Abbildung 9.9: Vorgehensweise zur Durchführung einer Zustandsprojektion, angepasst nach [SKL+11]. 221

Abbildung 9.10: Rendering einer zustandsbehafteten Chevron-Aktivität in einem Webbrowser. 223

Abbildung 9.11: Kennzeichnung von Spracherweiterungen von BPEL für rr-Fragmente am Beispiel eines rr-Fragments zur Realisierung einer Genehmigungsanforderung. 224

Abbildung 10.1: Legende zur Architekturbeschreibung. 228

Abbildung 10.2: Architekturüberblick und Zusammenhang der Komponenten. 229

Abbildung 10.3: Architektur der Dienstkomposition zur Realisierung von Sichten, angepasst nach [SCF+11]. 230

Abbildung 10.4: Screenshot zur Sichtendefinition in der Implementierung nach [Cai11]....232

Abbildung 10.5: Screenshot zu Parametern eines Dienstaufrufs in der Implementierung nach [Cai11]. 233

Abbildung 10.6: Architektur der Bibliothek für Prozessfragmente, angepasst nach [SKL+10]. 234

Abbildung 10.7: Erstellen eines Prozessfragments in der Bibliothek für Prozesse und Prozessfragmente. 235

Abbildung 10.8: Architektur der Modellierungsumgebung für Prozesse. 237

Abbildung 10.9: Screenshot des Eclipse BPEL Designer [Ecl11] für die musterbasierte Werkzeuganalyse. 239

Abbildung 10.10: Screenshot der erweiterten Modellierungsumgebung für Prozesse [SDH+12]. 240

Abbildung 10.11: Architektur der Monitoring-Plattform für Prozesse, angepasst und erweitert nach [SLL11] und [SKL+11]. 243

Abbildung 10.12: Screenshot einer Sicht auf eine Prozessinstanz in der Monitoring-Plattform in der Implementierung nach [Lat10]. 245

Abbildung 10.13: Effektive Mustersequenz für Sichten in der Monitoring-Plattform. 246

Abbildung 11.1: Kreditvergabeprozessmodell aus COMPAS als Beispielszenario zur Validierung. 251

Abbildung 11.2: Diskussion der Effekte und Einsatzmöglichkeiten der Sichtenentwurfsmuster am Beispielszenario im Rahmen der Validierung. 251

Abbildung 11.3: Darstellung einer abstrahierten Prozessinstanz im Kompaktmodus mit hervorgehobenen rr-Fragmenten [COM10b]. 256

Abbildung 14.1: Auswahl von Mustern zur Validierung des musterbasierten Sichtenentwurfs. 289

Abbildung 14.2: Einfacher Sichtenentwurf auf Basis der ausgewählten Muster. 290

Abbildung 14.3: Beispielhafte Lösungsmöglichkeit für den geforderten Sichtenentwurf. ...290

14.6 Listingverzeichnis

Listing 9.1: Transformationsanweisung zum Ausschneiden eines rr-Fragments [SLS10b]..212

Listing 9.2: Regelvorlage als Grundlage zum Entfernen eines rr-Fragments [SLS10b]. 213

Listing 9.3: Fallabfragen einer beispielhaften Übertragungsregel, mit der Zustände kombiniert werden (Z3). 220

Listing 9.4: Beschreibung einer Übertragungsvorlage in XML, angepasst nach [Lie11]. 222

Listing 9.5: Beschreibung einer Übertragungsregel in XML, angepasst nach [Lie11]. 222

Listing 9.6: Ausschnitt eines zustandsbehafteten Chevron-Modells in XML, angepasst nach [Lie11]. 223

Listing 9.7: Ausschnitt eines zustandsbehafteten Chevron-Modells in SVG, angepasst nach [Lie11]. 223

Listing 9.8: Erweiterung von BPEL für den Eingang eines rr-Fragments [SAL+10a]. 225

Listing 9.9: Erweiterung von BPEL für den Ausgang eines rr-Fragments [SAL+10a].....225

Listing 9.10: Erweiterung von BPEL für Platzhalter [SAL+10a].....226

Listing 9.11: Erweiterung von BPEL für eindeutige Identifikationsmerkmale [SAL+10a]..226

Listing 9.12: Erweiterung von BPEL für den Fragmentkontext [SAL+10a].....227

Listing 9.13: Erweiterung von BPEL für graphbasierte rr-Fragmente mit mehreren Eingängen oder Ausgängen [SAL+10a]. 227

14.7 Tabellenverzeichnis

Tabelle 1.1: Zusammenfassung der identifizierten Forschungsprobleme und entsprechender Beiträge.....23

Tabelle 2.1: Zusammenfassung der Anforderungen an die Lösungsansätze.....59

Tabelle 2.2: Zusammenfassung der allgemeinen Feststellungen zum Rahmen dieser Arbeit.60

Tabelle 2.3: Zusammenfassung der Voraussetzungen der entwickelten Ansätze. 60

Tabelle 3.1: Musterbeziehungen der Sichtenentwurfsmuster, wobei die Beziehungstypen *kombinierbar* in Schwarz mit Kennzeichnung der Reihenfolge, *ähnlich* in Grün und *verwendet* in Blau codiert sind. 112

Tabelle 3.2: Tabellarische Form einer präzisierten verfeinerten Mustersequenz eines Sichtenentwurfs. 126

Tabelle 4.1: Verwendung der Sichtenentwurfsmuster in den Sichtenentwürfen.....145

Tabelle 7.1: Grobspezifikation zur Anpassung des Erscheinungsbildes von Aktivitäten, Datenelementen und Verbindungen für eine Sicht zur Überprüfung von Transformationen. 188

Tabelle 7.2: Verwendung der Sichtenentwurfsmuster in den Sichtenentwürfen mit Bezug auf Compliance.....192

Tabelle 11.1: Aufstellung der Rückmeldungen der befragten Teilnehmer.....252

Tabelle 11.2: Auswertung der Rückmeldungen mit Differenzierung von Forschung und Industrie.....252

Danksagung

Abschließend möchte ich mich bei denjenigen bedanken, die mich während der Erarbeitung und Erstellung der vorliegenden Dissertation begleitet haben.

Ich möchte mich bei meinem Doktorvater Prof. Dr. Frank Leymann bedanken, ohne dessen Motivationsfähigkeit und Enthusiasmus ich mich dieser Herausforderung nicht gestellt hätte. Mein Dank geht auch an meine Mentorin Dr. Dimka Karastoyanova, die mich stets gut beraten hat. Ferner bedanke ich mich bei meinem Zweitgutachter Univ. Prof. Dr. Schahram Dustdar.

Ich möchte mich auch bei meiner Familie bedanken, die mir stets zur Seite gestanden ist.

Danken möchte ich auch den Kollegen am Institut für Architektur von Anwendungssystemen für die guten Ratschläge und Diskussionen. Im Besonderen geht mein Dank an Daniel Schleicher, Alexander Nowak, Mirko Sonntag, Christoph Fehling, Oliver Kopp, Sema Zor, Tobias Anstett, Ralph Retter, Thorsten Scheibler, Katharina Görlach und Johanna Barzen.

Ebenfalls möchte ich mich bei den Partnern aus Forschung und Industrie für die gute Zusammenarbeit in dem Projekt COMPAS bedanken. An dieser Stelle möchte ich besonders Huy Tran, Ta'id Holmes, Uwe Zdun, Florian Daniel und Oktay Türetken erwähnen.

Mein Dank geht außerdem an die Studenten, die meine Forschung mit wissenschaftlichen Arbeiten und Implementierungstätigkeiten unterstützt haben.

Zu guter Letzt möchte ich mich bei den Teilnehmern der Validierungs-Workshops für ihr Interesse und ihre Aufgeschlossenheit bedanken.

Ehrenwörtliche Erklärung

Ehrenwörtliche Erklärung zu meiner Dissertation mit dem Titel:

„Sichten auf Geschäftsprozesse mit besonderer Betrachtung von Compliance“

Sehr geehrte Damen und Herren,

hiermit erkläre ich, dass ich die beigefügte Dissertation selbstständig verfasst habe. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet.

Ich versichere außerdem, dass ich die beigefügte Dissertation nur in diesem und keinem anderen Promotionsverfahren eingereicht habe und, dass diesem Promotionsverfahren keine endgültig gescheiterten Promotionsverfahren vorausgegangen sind.

_____ _____

Ort, Datum Unterschrift