Institut für Architektur von Anwendungssystemen
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3134

Adaptive Prozessmodellierung mittels mehrerer Abstraktionsschichten

Modood Ahmad Alvi

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Frank Leymann
Betreuer:	Monika Weidmann, M. Sc. Dipl.-Inf. David Schumm
begonnen am:	4. Februar 2011
beendet am:	4. August 2011
CR-Klassifikation:	H.4.1, H.5.3

Zusammenfassung

Geschäftsprozesse werden typischerweise von verschiedenen Prozessverantwortlichen modelliert. Das führt zu Modellen, die sich inhaltlich unterscheiden und auf verschiedenen Abstraktionsschichten liegen. Insbesondere existieren je nach Prozessperspektive mehrere Modelle eines einzelnen Geschäftsprozesses. Daraus ergeben sich drei Anforderungen: Erstens, die Zuordnung der Prozesselemente in den Abstraktionsschichten. Zweitens, Änderungen in einem Prozessmodell müssen an alle anderen verwandten Modelle weitergereicht werden. Drittens, die Einführung einer Notation für Konsistenz zwischen Prozessmodellen.

In dieser Arbeit wird ein Konzept zur adaptiven Modellierung von Geschäftsprozessen mittels mehrerer Abstraktionsschichten entwickelt. Dazu werden Änderungen an einem Prozessmodell in Änderungsoperationen ausgedrückt, die wiederum auf einer Modellzuordnung operieren. Diese Änderungsoperationen werden in einem Änderungsprotokoll abgelegt und bei Bedarf den verwandten Modellen weitergereicht. Jede Operation definiert dabei Bedingungen, die erfüllt sein müssen, um die Änderung zu übernehmen. Zwei Prozessmodelle sind demnach konsistent, nachdem alle Änderungsoperationen übernommen wurden. Die Übernahme einer Änderung kann wiederum eine Änderungsoperation des adaptierenden Prozessmodells generieren.

Die entwickelten Konzepte werden für den webbasierten Editor Oryx umgesetzt.

Abstract

Business process modeling is typically performed by various stakeholders resulting in models that differ in content and level of abstraction. In particular, there exist several models of the same business process depending on the perspective the modeler assumes. Given all these models, three main challenges arise: First, the alignment of process models in different abstraction levels. Second, changes in one model have to be propagated to all related models accordingly. Third, an introduction of a notion for consistency between given process models.

This thesis presents a concept for adaptive business process modeling in several abstraction levels. To accomplish this, changes performed on a process model are expressed as change operations that in turn operate on the structure of the model correspondence. These operations are stored in a change queue and flushed on demand to all related process models. Every change operation carries along a completion condition which has to be satisfied by the related process to adopt the change. Consequently, two process models are in sync when all change operations are adopted properly. The adoption of a change operation might in turn result into a new change operation of the adopting process model.

These concepts are implemented through a prototype using the web-based editor Oryx.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

Verzeichnis der Algorithmen

1 Einleitung

1.1 Problemstellung

Die Modellierung von Geschäftsprozessen gehört zu den grundlegenden Aufgaben des Prozessmanagements eines Unternehmens. An der Gestaltung eines Prozessmodells sind verschiedene Akteure beteiligt und deshalb existieren für den gleichen Geschäftsprozess verschiedene Modelle. Geht es dem Manager darum, eine unternehmensweite Prozesslandkarte zu erstellen und sie als Mittel der Analyse zu verwenden, wird ein Sachbearbeiter eher daran interessiert sein, in welcher Abfolge die einzelnen Prozessschritte ausgeführt werden, welche Mitarbeiterrollen ihnen zugeordnet sind und welche Bedingung vor und nach ihrer Ausführung gelten.

Ziel der Modellierung ist oft die (teil-) automatische Ausführung des Geschäftsprozesses oder zumindest eine ausreichend genaue Spezifikation des prozessunterstützenden IT-Systems. Ein Modell durchläuft in der Modellierung verschiedene Abstraktionsschichten und erfährt dadurch eine kontinuierliche Konkretisierung. Aus abstrakten Aktivitäten werden konkrete Systemkomponenten oder Web Service Aufrufe. Die unfreiwillige Vermischung von konzeptionellen, fachlichen und technischen Aspekten des Geschäftsprozesses führt zu überladenen, unübersichtlichen und unpraktischen Prozessmodellen.

Geschäftsprozesse sind ständig neuen Anforderungen unterworfen. Das macht es notwendig, Prozessmodelle entsprechend anzupassen. Muss in einem Modell lediglich eine Aktivität umbenannt werden, ändert sich in einem anderen Modell die Prozessschrittfolge. Prozessmodelle müssen angepasst werden, und zwar so, dass die verschiedenen Modelle eines Prozesses untereinander konsistent bleiben. Wichtig ist in in diesem Zusammenhang, inwieweit sich Änderungen in einem Prozessmodell auf verwandte Modelle auswirken.

Es existieren somit verschiedene Modelle eines Geschäftsprozesses auf verschiedenen Abstraktionsschichten, die untereinander konsistent gehalten werden müssen. Insbesondere muss man in der Lage sein, alle Modelle schnell an Änderungen anzupassen. Eine Änderung in einem Modell muss an alle verwandte Modelle weitergereicht werden. Abbildung 1.1 verdeutlicht die Ausgangssituation, wobei die verwendeten Begriffe im Kapitel 2 genauer erläutert werden.

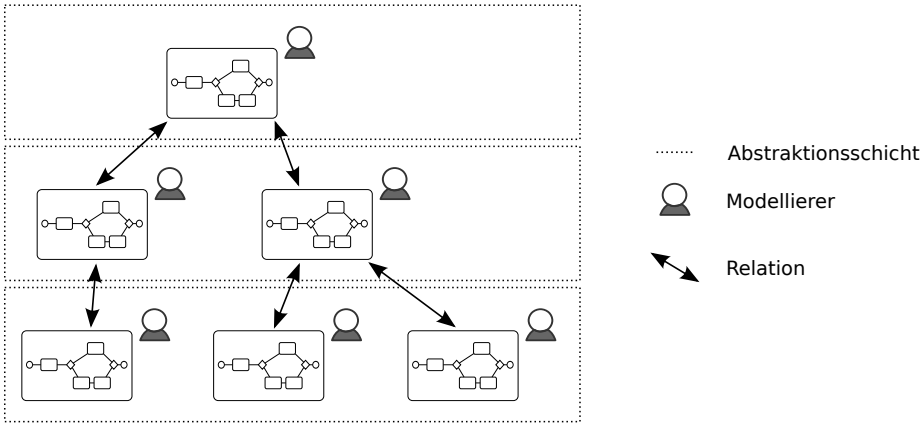

Abbildung 1.1: Modellierung in mehreren Abstraktionsschichten

1.2 Aufgabenstellung und Abgrenzung

Das Ziel der Arbeit ist es, ein Konzept zur adaptiven Geschäftsprozessmodellierung in mehreren Abstraktionsschichten zu entwickeln. Dem geht eine Analyse der vorhandenen Methoden zur Beschreibung von Komplexitätsstufen im Umfeld der adaptiven Geschäftsprozessmodellierung voraus. Zur Validierung des erarbeiteten Konzepts wird ein Prototyp implementiert.

Daraus ergeben sich konkret folgende Aufgaben:

- Analyse aktueller Ansätze zur Anwendung von Abstraktionsschichten bei der Prozessmodellierung
- Konzeption eines Modells zur adaptiven Modellierung in mehreren Schichten
- Konzept zur schichtenübergreifenden Synchronisierung von Änderungen
- Anbindung an eine Infrastruktur zur Verwaltung von adaptiven Prozessmodellen auf verschiedenen Abstraktionsschichten
- Implementierung eines Prototyps auf Basis des OpenSource-Editors Oryx [1] sowie Dokumentation der Architektur und Benutzerschnittstelle

Der Zweck eines Geschäftsprozessmodells ist meistens seine automatische Ausführung. Dabei kann ein Prozess mehrere Instanzen besitzen, die parallel und unabhängig voneinander existieren. Eine Änderung im Prozessmodell kann unter Umständen Änderungen in den einzelnen Instanzen mit sich bringen. Der Schwerpunkt dieser Arbeit liegt allerdings bei der adaptiven Modellierung von Geschäftsprozessen und berücksichtigt keine Konzepte zur Manipulation von bereits existierenden Instanzen eines Prozessmodells.

[1] Auf den OpenSorce-Editor Oryx wird im Kapitel 6 – Implementierung näher eingegangen

Geschäftsprozesse lassen sich in verschiedenen Notationen modellieren [Gad07]. Zur Veranschaulichung des Konzepts wählen wir BPMN 2.0 als Modellierungssprache [OMG11]. Weitere Details zu BPMN 2.0 finden sich unter Kapitel 2 - Grundlagen der adaptiven Prozessmodellierung.

1.3 Gliederung der Arbeit

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – Grundlagen der adaptiven Prozessmodellierung: Hier werden die grundlegenden Begriffe dieser Arbeit beschrieben.

Kapitel 3 – Anforderungen: Aus der Einleitung (Kapitel 1) und der Begriffsbestimmung der adaptiven Prozessmodellierung (Kapitel 2) ergeben sich Anforderungen, die in diesem Kapitel vorgestellt werden.

Kapitel 4 – Verwandte Arbeiten: Führt verwandte Arbeiten auf, die sich mit den Anforderungen aus dem vorherigen Kapitel (Kapitel 3) auseinandergesetzt haben.

Kapitel 5 – Modellierungskonzept: Dieses Kapitel stellt das Konzept der adaptiven Modellierung mittels mehrerer Abstraktionsschichten vor.

Kapitel 6 – Implementierung: Die Umsetzung des Konzepts und die Implementierung eines Prototypen sind Themas dieses Kapitels.

Kapitel 7 – Zusammenfassung und Ausblick: Zum Abschluss eine zusammenfassende Betrachtung der Arbeit und mögliche weitere Fragestellungen.

2 Grundlagen der adaptiven Prozessmodellierung

Dieser Abschnitt erklärt grundlegende Begriffe, die zum weiteren Verständnis der Arbeit notwendig sind. Er dient auch zur Abgrenzung zu anderen Begriffen, die im Zusammenhang der Geschäftsprozessmodellierung auftreten. Die Begriffserklärungen erheben nicht den Anspruch Definitionen zu sein, sondern sollen lediglich dem Verständnis dienen.

2.1 Adaptive Geschäftsprozesse

Man unterscheidet zwischen adaptiven Geschäftsprozessen und adaptiver Geschäftsprozessmodellierung [All95]. In diesem Abschnitt wird nicht auf die adaptiven Geschäftsprozesse eingegangen. Der Fokus liegt mehr auf der Modellierung und den damit verbundenen Tätigkeiten und Begrifflichkeiten. Lediglich der Übersicht wegen wird an dieser Stelle eine Begriffsklärung für adaptive Geschäftsprozesse wiedergegeben:

Adaptive Geschäftsprozesse „Adaptive Geschäftsprozesse besitzen die Eigenschaft, sich selbsttätig zu verbessern und an geänderte Anforderungen und Rahmenbedingungen anzupassen. Im Gegensatz zu umfassenden Reengineering-Projekten ist dabei keine Top-Management-Entscheidung erforderlich, sondern der Prozeß ist bereits so strukturiert, daß notwendige Änderungen im laufenden Geschäft erkannt werden und der Prozeß daran angepaßt wird.“ [All95]

2.2 Abstraktionsschichten

Es liegt in der Natur eines Modells, dass es bestimmte Attribute des Originals präteriert [Sta73]. Was auf den ersten Blick als Nachteil erscheint, ist die eigentliche Stärke des Modells: Es abstrahiert von irrelevanten Details und wird dadurch erst zweckdienlich [Lud02].

Der Modellbildung als geistige Tätigkeit geht also immer die Fähigkeit der Abstraktion voraus. Abstraktionsschichten geben das Maß der Abstraktion vor in der ein Modell abgebildet wird. Die Beschreibung der Abstraktionsschichten kann informell, semi-formal oder formal erfolgen. Beispielsweise kann man mit steigendem Abstraktionsgrad die Granularität des Modells einschränken oder jeder Abstraktionsschicht eine Teilmenge von Modellelementen zuordnen.

Existiert zu einem Prozessmodell ein Meta-Modell das für die Modellierung eine Menge von Modellelementen zu Verfügung stellt, dann bilden alle Prozessmodelle, die aus einer echten Teilmenge der Modellelemente modelliert wurden, eine Abstraktionsschicht.

Abstraktionsschicht Eine Abstraktionsschicht ist eine Menge von Prozessmodellen, deren Modellelemente eine echte Teilmenge zu den Modellelementen des Meta-Modells bilden.

Davon zu unterscheiden sind sogenannte Sichten („process views“) auf ein Prozessmodell. Es folgt eine informelle Begriffsbestimmung, die lediglich dem Verständnis dienen soll.

Sicht auf ein Prozessmodell Eine Sicht auf ein Prozessmodell ist wie eine Schablone, die auf das Modell gelegt wird und so unerwünschte Details ausblendet und erwünschte Zusammenhänge aufzeigt [RBRB06].

2.3 Adaptive Geschäftsprozessmodellierung

Grundsätzlich ist jedes Modell an veränderte Bedingungen anpassbar. Die Herausforderung besteht allerdings darin das Modell so zu gestalten, dass der Aufwand einer späteren Anpassung so gering wie möglich gehalten wird. Dazu bedarf es nicht nur der Weitsicht des Modellierers, sondern auch Methoden und Werkzeuge zur Gestaltung und Bearbeitung leicht anpassbarer Prozessmodelle.

Adaptive Geschäftsprozessmodellierung Unter adaptiver Geschäftsprozessmodellierung verstehen wir die Modellierung von Geschäftsprozessen unter dem Aspekt der leichten Anpassungsfähigkeit der Modelle. Leicht anpassbar ist ein Modell, wenn der Aufwand einer Anpassung weit unter der einer Neumodellierung liegt.

Adaptive Geschäftsprozessmodellierung in mehreren Abstraktionsschichten Bezeichnet die adaptive Geschäftsprozessmodellierung unter dem Aspekt der leichten Anpassungsfähigkeit der Modelle in allen Abstraktionsschichten.

Prozessmodelle erfahren bei der Modellierung in Abstraktionsstufen eine schrittweise Verfeinerung. Die Granularität des Modells nimmt mit sinkendem Abstraktionsgrad zu. Unterprozesse werden genauer festgelegt, Ausnahmen werden behandelt und Ein- und Ausgabedaten der Aktivitäten spezifiziert.

Process refinement A process refinement is a process description in a more fine-grained representation [RGL+09].

Wir halten uns an folgende Beschreibung.

Verfeinerung Eine Verfeinerung eines Prozessmodells ist ein Prozessmodell dass den gleichen Prozess auf einem niedrigeren Abstraktionsgrad beschreibt.

2.4 Modellangleich

Beschreiben zwei Prozessmodelle den gleichen Sachverhalt in mehreren Abstraktionsschichten, bedarf es einer Zuordnung der Aktivitäten des einen Modells zu den Aktivitäten des anderen Modells.

Process alignment I „Given a pair of process models, the goal is to automatically establish a relation between the elements of one model (e.g. tasks or events) and the elements in the other model.“ [DDGBK09]

Process alignment II „Given two structures (e.g., ontologies or Petri nets), aligning one structure with another one means that for each entity (e.g., concepts and relations, or places and transitions) in the first structure, one tries to find a corresponding entity, which has the same intended meaning, in the second structure.“ [BEK+06]

Für unsere Zwecke soll die folgende Erklärung genügen.

Modellangleich Modellangleich bezeichnet die Zuordnung von Elementen eines Prozessmodells zu den Elementen eines anderen Prozessmodell. Wir nennen den Modellangleich *horizontal*, wenn beide Prozessmodelle in derselben Abstraktionsschicht liegen und *vertikal*, wenn sie in unterschiedlichen Abstraktionsschichten liegen.

2.5 BPMN

Für die Beschreibung und Darstellung von Geschäftsprozessmodellen gibt es eine Vielzahl von Sprachen [Gad07], die abhängig von der Domäne des Modells unterschiedliche Aufgaben erfüllen. Eine dieser Sprachen zur Beschreibung von Geschäftsprozessen ist die Business Process Modeling Notation (kurz BPMN).

Abbildung 2.1 zeigt die „BPMN Core Elements“ der BPMN [WWW05], eine detaillierte Einführung in BPMN findet sich unter [All10]. Abbildung 2.2 zeigt schematisch ein Beispiel für einen Prozess in BPMN.

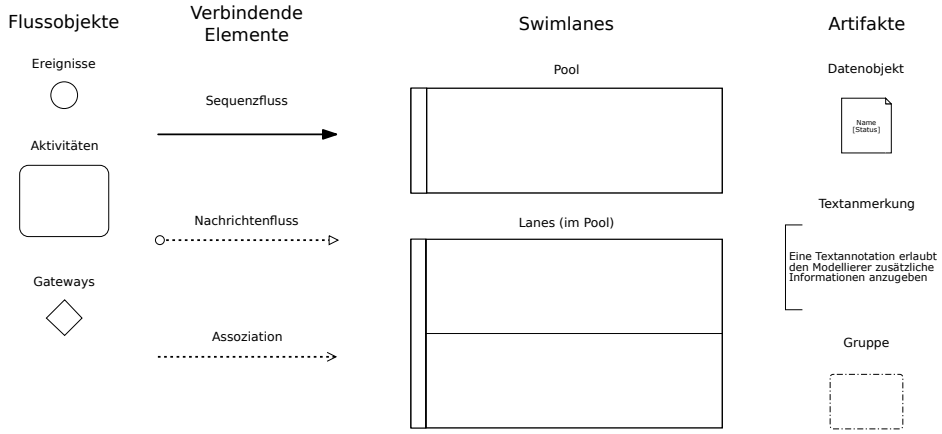

Abbildung 2.1: Kernelemente der BPMN nach [WWW05]

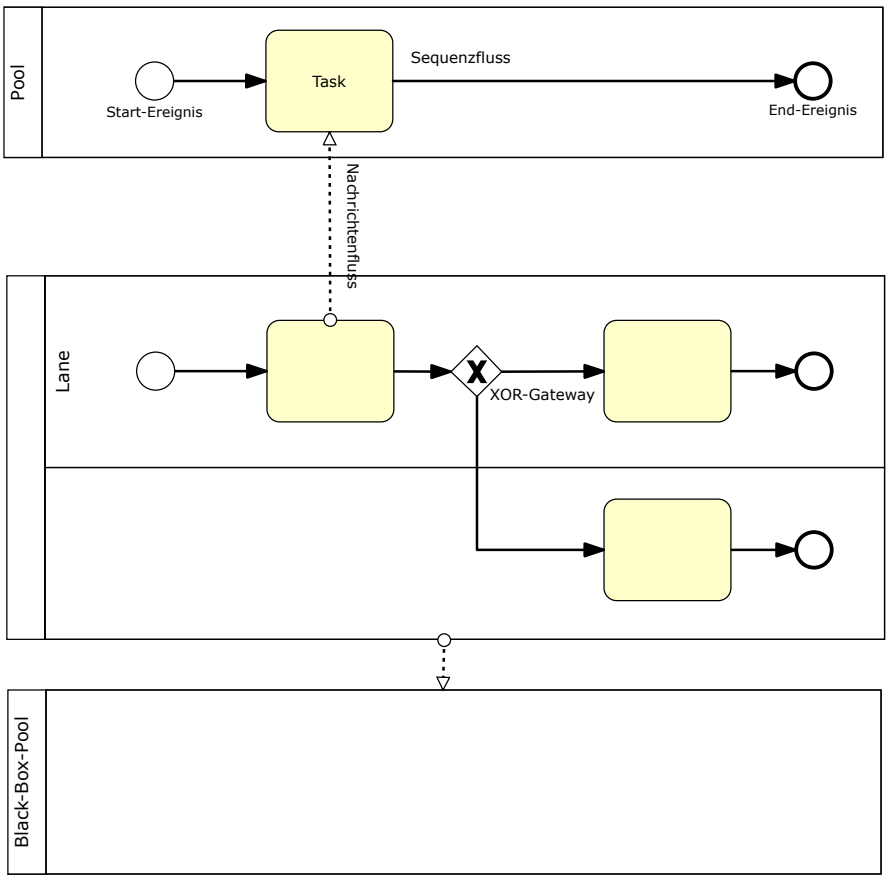

Abbildung 2.2: Beispieldiagramm mit BPMN

3 Anforderungen

Aus dem Kapitel 1 – Einleitung und dem Kapitel 2 – Grundlagen der adaptiven Prozessmodellierung ergeben sich Anforderungen an ein Modellierungskonzept der adaptiven Geschäftsprozessmodellierung. Es folgt in diesem Abschnitt eine genaue Untersuchung der Anforderungen.

3.1 Modellangleich

Zwei Prozessmodelle, die den gleichen Geschäftsprozess auf unterschiedlichen Abstraktionsschichten repräsentieren, müssen miteinander in Beziehung gesetzt werden. Im Allgemeinen Fall kann man von einer n:m Beziehung zwischen den Aktivitäten ausgehen. Dabei können folgende Sonderfälle auftreten:

1:1 Beziehung Einer Aktivität in Prozess A entspricht eine Aktivität in Prozess B. Das ist zum Beispiel der Fall, wenn zwei Aktivitäten semantisch äquivalent sind, ihre Bezeichner sich aber unterscheiden.

1:n Beziehung (n>1) Einer Aktivität in Prozess A entsprechen n Aktivitäten in Prozess B. Zum Beispiel können n Aktivitäten in Prozess B eine Verfeinerung der Aktivität in Prozess A darstellen.

n:1 Beziehung (n>1) n Aktivitäten in Prozess A entsprechen einer Aktivität in Prozess B. Dieser Fall tritt zum Beispiel auf, wenn ein technischer Prozess B eine Aktivität enthält, die n Aktivitäten von Prozess A implementiert.

0:1 Es gibt Aktivitäten in Prozess B zu denen es keine Entsprechungen in Prozess A gibt. Das kann zum Beispiel der Fall sein, wenn ein Prozess B eine Aktivität enthält, die unabhängig vom Prozess, technische Aspekte des Systems berücksichtigt, so zum Beispiel den Prozessfortschritt protokolliert.

1:0 Es gibt Aktivitäten in Prozess A zu denen es keine Entsprechungen in Prozess B gibt. Das kann immer dann auftreten, wenn zwei Prozessmodelle zwar den gleichen Prozess, aber vollkommen unterschiedliche Aspekte modellieren und keine der oben erwähnten Beziehungen zutrifft.

Ein Problem beim Aufbau einer Beziehung zwischen Aktivitäten ist die Bestimmung der Beziehungsgrundlage. Zwei Aktivitäten können in Relation gesetzt werden, weil sie syntaktisch ähnliche Bezeichner haben, eine ähnliche Bedeutung nach einer vorgegebenen Semantik besitzen, strukturelle ähnlich sind, im ähnlichen Kontext stehen oder andere ähnliche, die

Relation definierende Merkmale aufweisen. Eine manuelle Zuordnung der Aktivitäten ist praxisfern. Eine automatische und eindeutige Erkennung der Zuordnung ist jedoch genauso unrealistisch, was das Beispiel in Abbildung 3.1 plausibel macht.

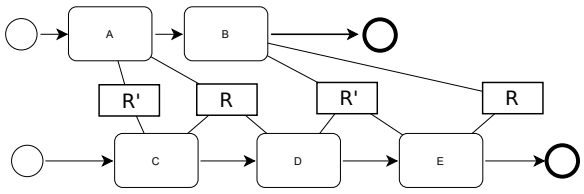

Abbildung 3.1: Es gibt mindestens zwei Möglichkeiten die Aktivitäten in Relation zu setzen

Die Aufgabe besteht nun darin eine Definition für den Modellangleich zu finden, die alle oben genannten Arten von Beziehungen abdeckt. Keine Anforderung ist, eine Definition für Ähnlichkeit von Prozessmodellen zu finden oder einen Algorithmus zu entwickeln, der die Zuordnung automatisiert.

3.2 Synchronisierung

Geschäftsprozesse sind oft Änderungen unterworfen. Änderungen in einem Prozessmodell müssen auf verwandte Prozessmodelle übertragen werden - die Prozessmodelle müssen synchron gehalten werden. Da es sich oft um ein Netz verwandter Prozessmodelle handelt, müssen die Änderungen sukzessive in alle Modelle weitergereicht werden (Abbildung 3.2). Befinden sich die Modelle in verschiedenen Abstraktionsschichten, ist zum Beispiel sowohl eine Top-Down- als auch Bottom-Up-Propagation der Änderungen nötig.

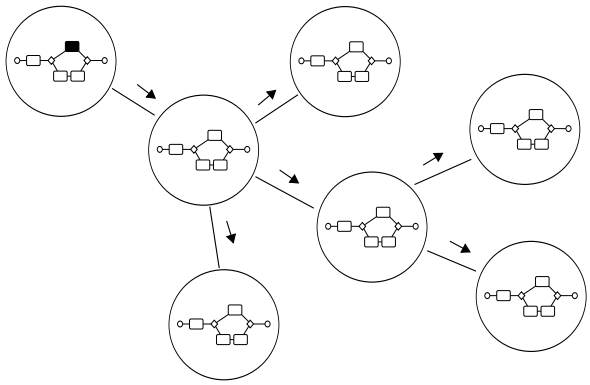

Abbildung 3.2: Weitergabe einer Änderung in einem Netz von Prozessmodellen

Im Idealfall übernimmt jeder Prozess die Änderung automatisch, das heißt die Prozesse sind derart miteinander verknüpft und möglicherweise mit einer Semantik hinterlegt, sodass jeder Prozess *weiß*, wie er die Änderung zu übernehmen hat. Das allerdings scheint ohne

eine starke Einschränkung des Modellierers unrealistisch zu sein. Es ist zum Beispiel nicht immer nötig eine Änderungen auch wirklich zu übernehmen, da die Änderungen einen Prozessabschnitt betrifft, der für den verwandten Prozess irrelevant ist und somit nicht im Fokus der Modellierung steht. Um das *Wie* und *Wann* einer Übernahme zu entscheiden, braucht es den Sachverstand eines Menschen.

Eine Anforderung an das Konzept ist, einen Mechanismus zu entwickeln mit dem Änderungen in einem Prozessmodell an alle anderen verwandten Modelle weitergereicht werden. Es muss ersichtlich werden, um was für eine Änderung es sich handelt, welche Aktivitäten davon betroffen sind und was der Modellierer damit bezweckt. Des Weiteren muss das Konzept erklären, was es heißt, wenn zwei Prozessmodelle *synchron* sind.

3.3 Konsistenz zwischen Prozessmodellen

Beschreiben zwei Prozessmodelle den gleichen Zusammenhang auf verschiedenen Ebenen der Abstraktion, ist es wichtig, dass beide Modelle zueinander konsistent sind und konsistent gehalten werden.

Dazu kann man zwei Ansätze für Konsistenz verfolgen:

- Zwei Prozessmodelle sind zueinander konsistent, wenn sie *tatsächlich* ein und denselben Prozess auf verschiedenen Abstraktionsstufen beschreiben. Betrachten wir hierzu Abbildung 3.3: Der Unterschied zwischen den Prozessen liegt in der Reihenfolge der Aktivitäten A und B. Hinterlegen wir beispielsweise beide Modelle mit einer Ausführungssemantik, dann sind diese Modelle *nicht* konsistent.
- Zwei Prozessmodelle sind konsistent zueinander, wenn sie synchron sind, das heißt es gibt keine Änderung in einem Prozessmodell, die nicht im anderen Modell übernommen wurde.

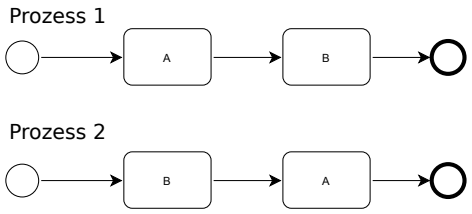

Abbildung 3.3: Die Prozesse unterscheiden sich in der Ausführungsreihenfolge ihrer Aktivitäten. Sind sie konsistent zueinander?

Eine automatische Überprüfung der Konsistenz ist jedoch nur für den letzteren Ansatz denkbar. Der erste Ansatz scheint demgegenüber sehr schwer umsetzbar zu sein. Selbst wenn ein System eine Aktivität in einem semantischen Kontext stellen kann, ist es im Allgemeinen unmöglich festzustellen, ob zum Beispiel bestimmte Aktivitäten *tatsächlich* eine Verfeinerung

darstellen. Abbildung 3.4 macht diese Problematik deutlich. Die Aktivität „Baue Schloss“ kann je nach Zweck und Verständnis theoretisch in beliebige Unteraktivitäten unterteilt werden. Daraus ergeben sich ganz unterschiedliche Prozessmodelle, die alle konsistent sein können.

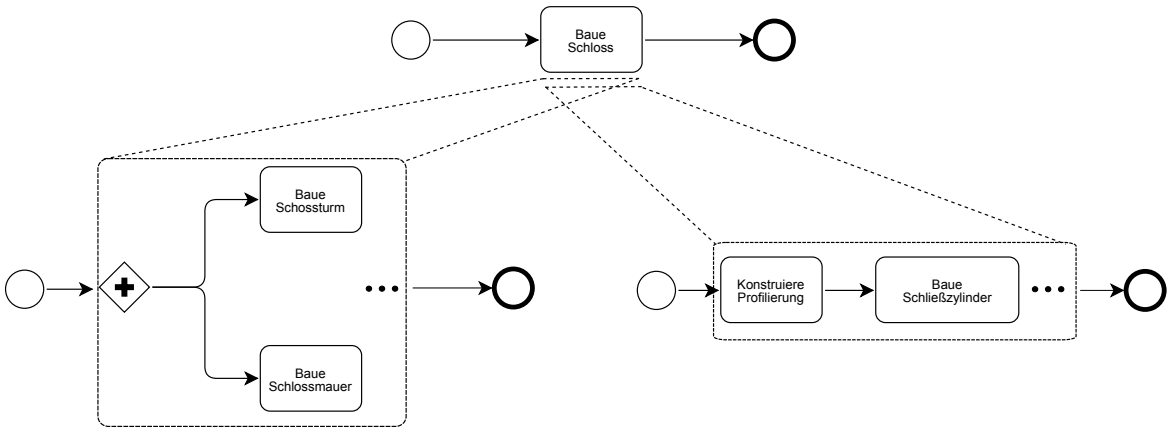

Abbildung 3.4: Welches Prozessmodell ist eine konsistente Verfeinerung?

Es braucht also eine Definition von Konsistenz zwischen Prozessmodellen, die Auskunft darüber gibt, ob und wie zwei Prozessmodelle miteinander in Relation stehen. Es muss zum Beispiel geklärt sein, ob zwei Prozessmodelle zueinander konsistent sind, wenn es in beiden Modellen Aktivitäten gibt, die in keiner Relation stehen. Die Prüfung der Korrektheit dieser Relation aus semantischer Sicht ist allerdings weder umsetzbar noch gefordert.

4 Verwandte Arbeiten

Dieses Kapitel gibt einen Überblick über Arbeiten, die sich mit Teilaspekten des gestellten Problems befassen und so zu (Teil-) Lösungen beitragen können. Die Auswahl der Arbeiten orientiert sich an die im letzten Kapitel erläuterten Anforderungen der adaptiven Geschäftsprozessmodellierung:

1. Welche Arbeiten gibt es zu Modellierung von Geschäftsprozessen in mehreren Abstraktionsschichten? Wie werden Abstraktionsschichten eingeführt und wie werden sie genutzt?
2. Welche Ansätze gibt es um zwei Prozessmodelle in Relation zu setzen?
3. Welche Ansätze gibt es um mit Änderungen auf der Modellebene umzugehen?
4. Wie können Geschäftsprozesse möglichst adaptiv gestaltet werden?
5. Gibt es Ansätze zur Synchronisierung von Prozessmodellen?
6. Wie lassen sich Prozessmodelle zueinander konsistent halten?

4.1 Abstraktionsschichten bei der Prozessmodellierung

[LS10] befasst sich mit der Frage wie ein schichtenbasiertes Denken in der Modellierung von Geschäftsprozessen dazu beitragen kann, eine Verbindungen zwischen Geschäftsprozessen und IT-Systemen herzustellen. Die Autoren entwickeln dazu ein Framework mit fünf Abstraktionsschichten und kommen zu dem Ergebnis, dass die Modellierungen von Geschäftsprozessen in mehreren Abstraktionsschichten eine mögliche Lösung des Problems zur Überbrückung der Diskrepanz zwischen Geschäftsprozessen und IT-System darstellt. Abbildung 4.1 zeigt die verschiedenen Schichten des Frameworks.

Robert Shapiro, Mitglied der OMG BPMN 2.0 Finalization Task Force, schlägt eine Einteilung der BPMN-Elemente in vier Klassen vor: SIMPLE, DESCRIPTIVE, DODAF und COMPLETE [Sha10]. SIMPLE enthält 8 Elemente, die sich dafür eignen, einen Prozess in seiner Gesamtheit zu erfassen. Ein typisches Szenario für den Einsatz von SIMPLE wäre eine Sitzung in der ein Analyst mit dem Prozesseigentümer den Prozess mündlich bespricht und ihn dabei erfasst. Eine schematische Darstellung der vier Klassen ist in Abbildung 4.2 abgebildet.

In seinem Buch „BPMN Method and Style“ beschreibt Bruce Silver drei Abstraktionsschichten für Prozessmodelle, die auf die verschiedenen Typen von Anwendern von BPMN zugeschnitten sind [Silo9]. Level 1 Modelle sind deskriptive Modelle. Sie dienen zur Erfassung und

Abbildung 4.1: Schichten-Pyramide nach [LS10]

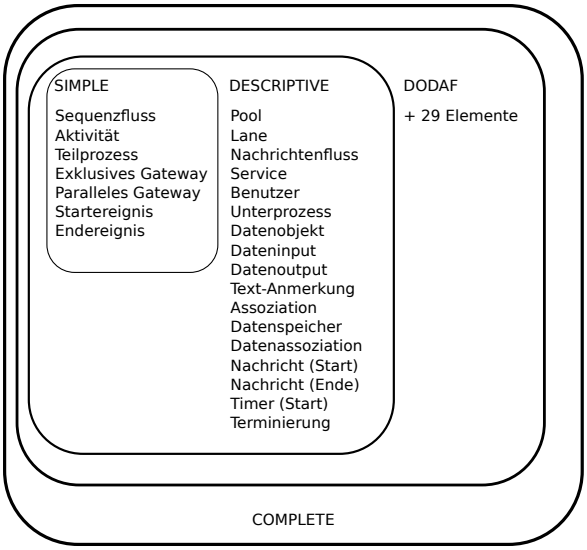

Abbildung 4.2: Schematische Darstellung der Shapiro-Klassen aus [Sha10]

Dokumentation von betrieblichen Abläufen. Der Modellierer beschränkt sich auf eine Grundmenge von BPMN Elementen und kann spezielle Ausnahmefälle außer Acht lassen. Level 2 stellt die analytische Sicht auf den Geschäftsprozess. Obwohl Level 2 Modelle nicht „ausführbar“ sind, bilden sie die Schnittstelle zur IT und sind mit einer „ausführenden“ Semantik hinterlegt. Level 3 macht den Prozess ausführbar. Dazu werden die Ein- und Ausgabewerte der Aktivitäten sowie die aufzurufenden Systemkomponenten genau festgelegt.

Die BPMN 2.0 Spezifikation selbst unterteilt ihre Elemente in 10 Schichten, die teilweise aufeinander aufbauen [OMG11]. Die angegebenen Schichten sind mit all ihren Elementen, Abhängigkeiten und Attributen zwar selbst keine Abstraktionsschichten wie man sie

unter dem Aspekt der Modellierung einsetzen würde, können aber zur Herleitung eines schichtenbasierten Konzepts der Modellierung verwendet werden. Abbildung 4.3 zeigt die Schichtenstruktur. Das begleitende Dokument „BPMN 2.0 by Example“ [OMG10] zeigt im Kapitel 6 „Incident management“ wie BPMN 2.0 dazu genutzt werden kann, in mehreren Abstraktionsschichten ein und den selben Prozess zu modellieren.

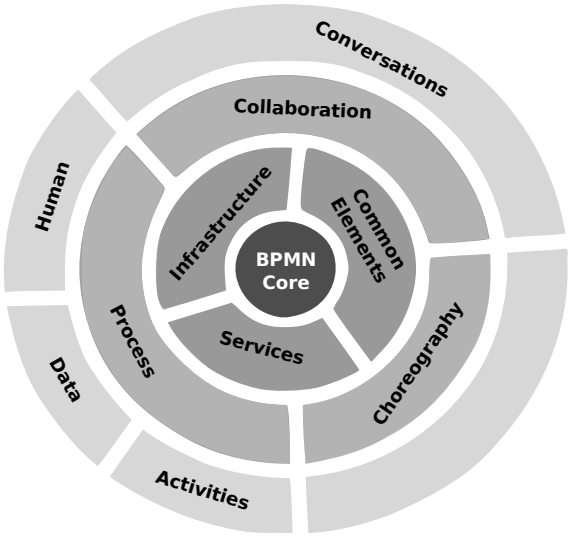

Abbildung 4.3: Schichtenstruktur des BPMN 2.0 Standards nach [OMG11]

Alle genannten Arbeiten erwähnen zwar Abstraktionsschichten als Bestandteil der Modellierung von Geschäftsprozessen, allerdings nicht als integralen Bestandteil eines Konzepts zur Modellierung in mehreren Abstraktionsschichten, das den Anforderungen (Kapitel 3 - Anforderungen) der adaptiven Geschäftsprozessmodellierung genügt. Insbesondere adressieren sie nicht das Problem (a) wie die Modelle in Relation zu setzen sind, (b) die Synchronisierung der Modelle funktionieren soll und (c) die Modelle untereinander konsistent gehalten werden können.

4.2 Modell-zu-Modell-Transformationen

„Business-driven development“ ist eine Methode zur Abbildung von Geschäftsprozessen auf IT-Lösungen. Dabei werden Geschäftsprozesse auf domänenspezifische Modelle abgebildet und anschließend durch mehrere (teil-) automatische Modell-zu-Modell-Transformationen in (bestehende) IT-Lösungen überführt [KHK+08]. Der Vorteil dieser Vorgehensweise liegt in der (teil-) automatischen Übertragung von Änderungen in der Modellebene auf die IT-Ebene, da die IT-Systeme oder zumindest ihre Spezifikationen aus den Modellen generiert werden [KHK+08].

[All07] entwickelt einen Ansatz, „mit dem sich aus grobgranularen fachlichen Geschäftsprozessmodellen detaillierte, ggf. auch technisch verfeinerte, ausführbare Modelle in verschiedenen Zielnotationen erzeugen lassen“. Ausgangspunkt der Transformation ist ein grobgranularer Prozesss, der eine Instanz eines vorgegebenen Musters darstellt. Die Transformation bildet nun diesen abstrakten Prozess mit Hilfe von Transformationsregeln auf einen feingranularen Prozess ab. Abbildung 4.4 verdeutlicht die generelle Vorgehensweise. Die Beziehung zwischen den Modellelementen im Muster und denen in der Ausprägung wird mit Annotationen und Stereotypen aus der UML-Notation hergestellt.

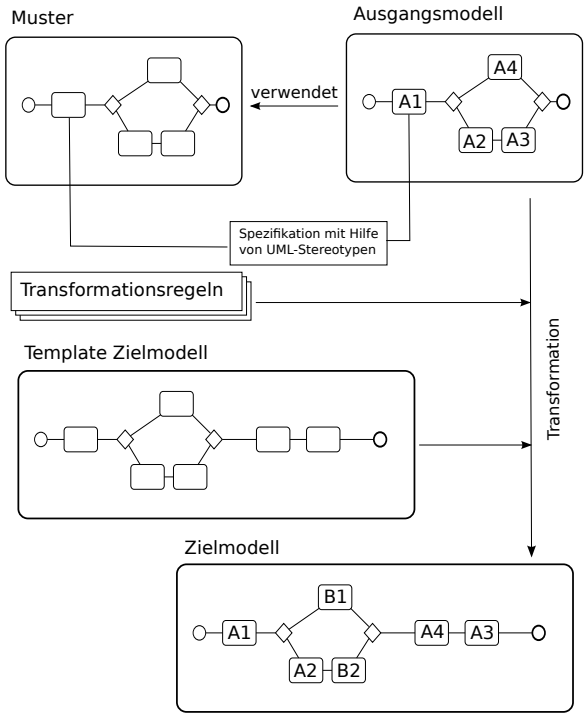

Abbildung 4.4: Modelltransformation nach Allweyer

In [KHK+03] stellen die Autoren eine Transformations-Engine vor, die aus ADF-Modellen [DBB+02] und UML-Aktivitätsdiagrammen [RQZ07] einen Prozessgraphen erzeugt und diesen dann in einen Strukturgraphen übersetzt. Aus dem Strukturgraphen lassen sich dann direkt WS-BPEL Artifakte [Org07] erzeugen. Abbildung 4.5 zeigt eine Übersicht der Transformation.

Einen Einblick in das vielseitig einsetzbare Konzept der Modell-zu-Modell Transformation geben folgende Arbeiten: [BFR+] reichert BPMN-Modelle mit Meta-Daten an, um diese in WS-BPEL-Prozesse umzuwandeln, [SO00] führt Änderungsoperationen auf Prozessmodellen ein, die garantieren, dass bestimmte Bedingungen bei der Transformation nicht verletzt werden und [ABL08] annotiert Petri-Netze mit WS-BPEL Aktivitäten, unterteilt sie in Komponenten und überführt sie mit einem iterativen Reduktionsalgorithmus in einen lesbaren WS-BPEL Prozess.

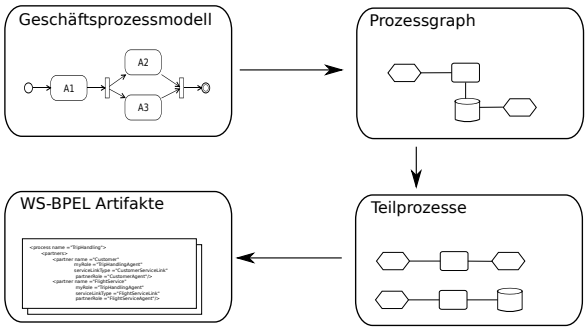

Abbildung 4.5: Transformationsmodell mit Prozessgraphen

Die vorgestellten Arbeiten zeigen, dass bei der Modell-zu-Modell Transformation das Zielmodell von dem Ausgangsmodell bestimmt wird. Jeder Transformationschritt lässt sich zwar parametrisieren, die Auswahl an möglichen Zielmodellen bleibt aber dennoch begrenzt. Bei der adaptiven Modellierungen mittels mehrerer Abstraktionsschichten gehen wir allerdings davon aus, dass (a) mehrere Prozessmodelle parallel existieren, (b) mehrere Akteure daran beteiligt sind, (c) jeder Modellierer sein Modell theoretisch beliebig gestalten kann, das heißt er entscheidet selbst welchen Aspekt des abstrakten Prozesses er genauer modellieren möchte und (d) Prozessmodelle teilweise zirkulär in Relation stehen. In Abbildung 4.6 werden die Unterschiede nochmals verdeutlicht.

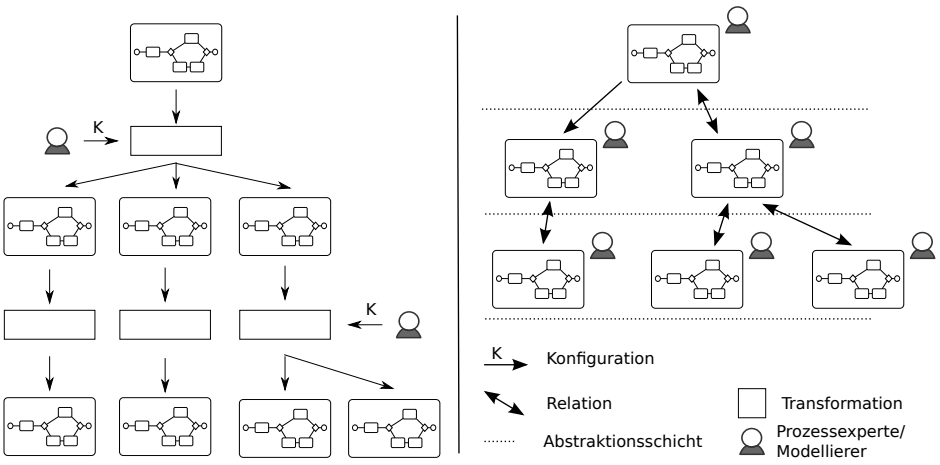

Abbildung 4.6: Unterschied M2M-Transformation (links) und adaptive Modellierung (rechts): Generierte Zwischenmodelle auf der einen Seite und verschieden granulare Prozessmodelle auf der anderen Seite.

4.3 Modellübereinstimmung und Modellangleich

Der horizontale und vertikale Modellangleich von Prozessmodellen ist Gegenstand von vielen Arbeiten um das Thema „Process Alignment“. Die Schwierigkeit eine adäquate Verbindung zwischen beschreibenden Prozessmodellen und den dazugehörigen ausführbaren Modellen oder den Prozess ausführenden IT-Systemen herzustellen, wird gemeinhin als „Business-IT-Gap“ bezeichnet [WBMW]. In diesem Zusammenhang fallen auch Arbeiten, die sich mit der Übereinstimmung von Prozessmodellen beschäftigen („Model Correspondence“). Andere Arbeiten untersuchen wie Zusammenhänge zwischen Aktivitäten von Prozessmodellen erkannt und dokumentiert werden können.

[DDGBK09] stellt zwei Techniken vor anhand der man Prozessgraphen vergleicht und eine Zuordnung zwischen den Elementen herstellt. Die erste Technik vergleicht Knotenbeschriftungen der Prozessgraphen mit Hilfe einer Abstandsfunktion auf Zeichenketten [1] und stellt eine Zuordnung her, bei der die Summe der Abstände der zugeordneten Elemente minimal ist. Die zweite Technik durchläuft alle möglichen Zuordnungen und wendet jeweils eine Gütefunktion an, die die topologische Struktur der Prozessgraphen vergleicht. Der Vorgang stoppt, wenn eine Zuordnung gefunden wurde, deren Güte einen bestimmten Wert unterschreitet.

[KGFE08] stellt einen Ansatz vor, mit dem Unterschiede zwischen Prozessmodellen erkannt und aufgelöst werden. Dazu werden die Modelle in Fragmente zerlegt und in einem Strukturbaum (Process Structure Tree [VVK09]) organisiert. Anhand von Modellverknüpfungen werden Unterschiede in den Prozessmodellen definiert und daraus ein Änderungsprotokoll erstellt.

In [BBR11] führen die Autoren das Business-IT-Mapping-Model (BIMM) ein, um eine Verbindung zwischen einem Geschäftsprozessmodell wie er von Geschäftsleuten modelliert wird, und einen Systemprozess, der den gleichen Prozess auf technischer Ebene darstellt, herzustellen. Jede Aktivität des Geschäftsprozessmodells wird mit Hilfe von Basisoperationen auf Aktivitäten des Systemprozess abgebildet. Im einfachsten Fall wird die Aktivität umbenannt. Abbildung 4.7 zeigt einen Geschäftsprozess und den dazugehörigen Systemprozess. Das BIMM stellt nun die Verbindung zwischen den beiden Modellen her.

In [HZJ04] wird ein Konzept vorgestellt mit dem Prozessmodelle in technische Modelle umgewandelt werden. Zuerst wird zwischen dem eigentlichen Prozess und dem technischen Prozess unterschieden. Der technische Prozess berücksichtigt sowohl die Geschäftslogik als auch die vorhandenen Systemkomponenten zur Realisierung des Geschäftsprozesses. Das Problem ist die Abbildung zwischen den beiden Modellen. Dazu werden sogenannte Realisierungstypen eingeführt, die ein technisches Modell aus einem Prozessmodell herleiten. Ein Realisierungstyp ist zum Beispiel die Aggregation, bei der eine Aktivität im Prozessmodell in mehrere Aktivitäten im technischen Prozess umgewandelt wird.

[1] Es handelt sich hier um die Levenshtein-Distanz von Zeichenketten

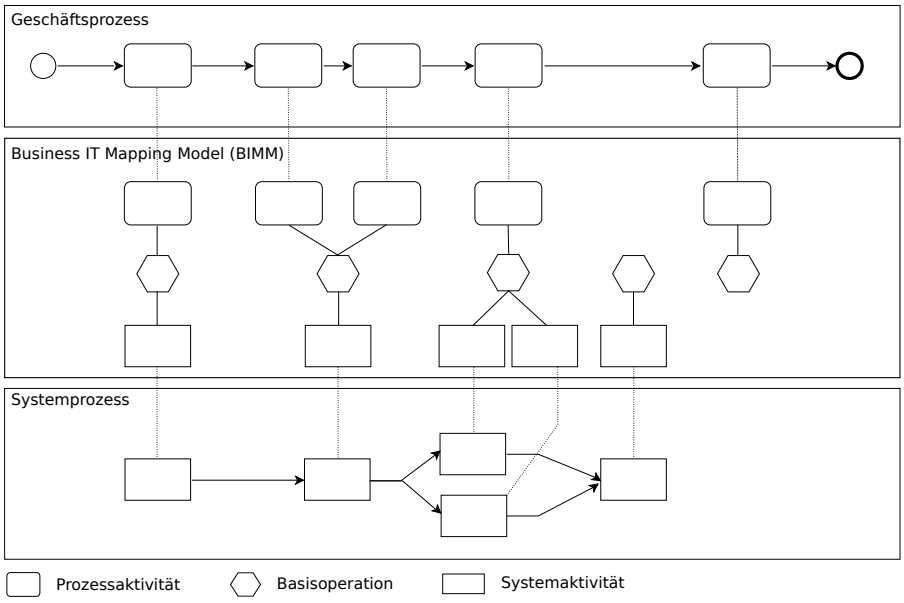

Abbildung 4.7: Business-IT-Mapping-Model nach [BBR11]

Einen ähnlichen Ansatz verfolgt [Deco6] und führt ein „Process Support Layer“ ein, um Inkompatibilitäten zwischen Geschäftsprozessmodellen und den ausführenden IT-Systemen zu überbrücken. Teil des Konzepts sind Lösungsmuster, die bei bestimmten Zuordnungsproblemen Anwendung finden. Realisiert beispielsweise eine einzige Systemfunktion AB zwei Aktivitäten A und B im Geschäftsprozess, dann braucht es eine zusätzlichen Systemfunktion C, die die Eingabewerte beider Aktivitäten A und B entgegennimmt und anschließend die eigentliche Systemfunktion B aufruft.

Die semantische Ausrichtung von Prozessmodellen ist das Thema von [BEK+06]. Die Autoren argumentieren, dass die ausschließlich syntaktische Ausrichtung zweier Prozessmodelle oft am unterschiedlichen Vokabular der Modelle scheitert. Darüber hinaus führen Homonyme und Polyseme in der natürlichen Sprache zu Missverständnissen. Deswegen schlagen sie eine semantische Überprüfung der Prozessmodelle vor. Prozessmodelle werden mit Ontologien beschrieben und jedes Modellelement des einen Prozesses mit den Modellelementen des anderen Prozesses verglichen. Zur Hilfe nimmt man eine übergeordnete Ontologie, die die Beziehungen zwischen den Begriffen beschreibt. Abbildung 4.8 zeigt eine schematische Darstellung der Vorgehensweise.

Weitere Arbeiten zu diesem Thema finden sich bei [GLKE10], [Dij08], [Dij], [DDM08], [GZ09], [MW06], [ZP09]. Eine gute Übersicht von verschiedenen Semantiken für Prozessmodelle und darauf aufbauenden Vergleichen von Prozessmodellen findet sich in [Gla01].

Wie bereits im Kapitel 3 – Anforderungen erwähnt, ist es unrealistisch anzunehmen, dass eine Relation zwischen Prozessmodellen automatisch hergestellt werden kann. Ansätze, die die Semantik und Struktur von Prozessmodellen berücksichtigen, sind zwar viel versprechend,

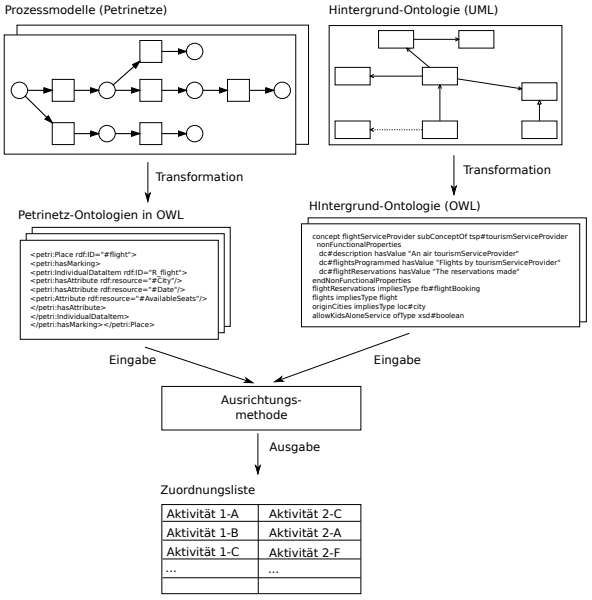

Abbildung 4.8: Semantische Ausrichtung von Prozessmodellen nach [BEK+06]

zwingen den Prozesseigentümer jedoch sein Modell mit einer vorgegebenen Semantik zu hinterlegen. Die vorgestellten Arbeiten beschreiben Techniken mit denen man Prozessmodelle vergleicht und eine Zuordnung der Elemente herstellt, nicht aber wie man (a) in mehreren Abstraktionsschichten modelliert und (b) die verschiedene Prozessmodelle synchron hält.

4.4 Adaptivität bei der Modellierung von Geschäftsprozessen

Folgende Arbeiten handeln von Adaptivität in der Modellierung von Geschäftsprozessen.

Die Publikation von [WSR09] zeigt Anforderungen auf, die an ein PAIS [2] gestellt werden, wenn es dynamische und adaptive Elemente der Prozessmodellierung und -ausführung bereitstellen möchte. Es werden Ansätze für die verschiedenen Lifecycle-Phasen (Design, Model, Execute, Monitor) aufgezeigt, wobei der Fokus auf den letzten drei Phasen liegt.

Adaptivität in der Modellierungsphase erreicht man dabei folgendermaßen:

1. Granularitätskontrolle: Subaktivitäten einer Hauptaktivität werden nicht festgelegt und können somit zu einem späteren Zeitpunkt je nach Anwendungsfall ausgewählt werden.

[2] PAIS steht für „process-aware information system“. Weitere Ausführungen dazu in [WSR09].

2. Flexibilität durch Aufzählung: Eine weitere Möglichkeit die Flexibilität schon in der Modellierungsphase zu erhöhen, besteht darin alle alternativen Ausführungspfade aufzulisten.
3. Prozesskonfiguration: Indem Prozessmodelle konfigurierbar gestaltet und anschließend mit einer Konfiguration belegt werden, lässt sich eine Anpassung der Modelle durch eine Konfigurationsänderung realisieren.
4. Späte Service-Anbindung: Die Anbindung von Web Services an Prozessaktivitäten geschieht so spät wie möglich, was eine Entkopplung der Geschäftslogik und zugleich eine Austauschbarkeit der Web Services zur Folge hat.
5. Prozessfragmente: Prozessfragmente liegen in einem Repository bereit und können je nach Anwendungsfall in das Prozessmodell integriert oder ausgetauscht werden.

Ein Prozessmodell kann auf verschiedene Art und Weisen angepasst werden. In [WRRM08] werden 14 typische Anpassungen an Prozessmodellen aufgezählt und in [RMRW08] ihre formale Semantik definiert. Eine Änderung an einem Prozessmodell wäre die Hinzunahme oder Wegnahme einer Aktivität. Abbildung 4.9 gibt alle 14 Anpassungsmuster wieder. Des Weiteren stellen die Autoren vier Anpassungen für „Changes in Predefined Regions“ vor; das sind vordefinierte Platzhalter im Prozessmodell, die nach folgenden Adaptionsmustern verändert werden können:

1. Späte Auswahl der Prozessfragmente: Prozessfragmente können zur Laufzeit an die Stelle des vordefinierten Platzhalter eingesetzt werden.
2. Späte Modellierung von Prozessfragmenten: Prozess stoppt in der Ausführung und wartet auf die Eingabe des Prozesseigentümers, der erst jetzt ein Prozessfragment anstelle des Platzhalters modelliert.
3. Späte Ausführungsreihenfolge: Die Reihenfolge der auszuführenden Prozessfragmente wird erst zur Laufzeit festgelegt.
4. Mehrfachausführung einer Aktivität: Die Häufigkeit mit der eine Aktivität ausgeführt wird, kann zur Laufzeit festgelegt werden.

Eine Möglichkeit schnell auf Anpassungen zu reagieren, ist die variable Gestaltung eines generischen Prozessmodells. Dazu führt man Variabilitätspunkte im Prozessmodell ein, für die alternative Ausführungen der Prozesselemente angegeben werden. Je nach Fall entscheidet man sich für eine Variante des sogenannten Referenzprozesses.

In [WKK+11] wird ein Konzept zur Variabilitätsmodellierung vorgestellt. Die Autoren erweitern BPMN um ein Modellelement „Variable Region“ und weisen diesen Elementen bei der Modellierung des Referenzprozesses jeweils eine Liste von möglichen einsetzbaren Prozessfragmenten zu. Bei der Erstellung einer Variante des Referenzprozesses werden die variablen Regionen durch einen ihrer zu Verfügung stehenden Prozessfragmente ersetzt.

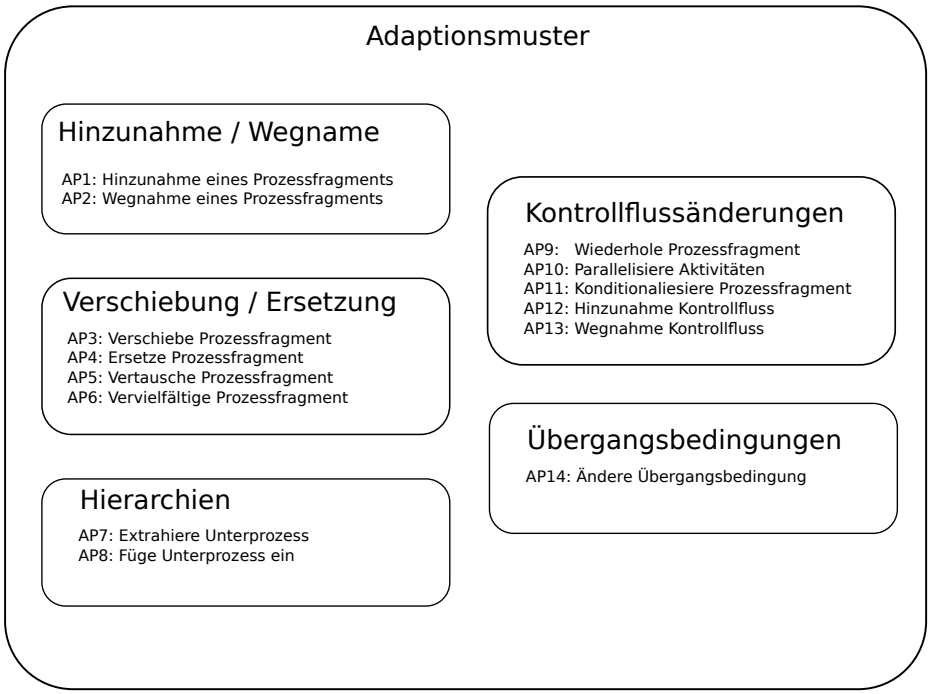

Abbildung 4.9: Adaptionsmuster nach [WRRM08]

Die genannten Arbeiten versuchen mit verschiedenen Ansätzen das Problem der schnellen Anpassung von Prozessmodellen zu lösen. Dabei verändern oder gestalten sie ein Prozessmodell derart, dass es den veränderten Anforderung der Umwelt genügt oder genügen kann. Unberücksichtigt, unter dem Aspekt der adaptiven Modellierung, bleibt jedoch (a) die Koexistenz der Modelle in den verschiedenen Abstraktionsschichten sowie (b) die Synchronisierung der Modelle.

4.5 Modellierungsrichtlinien

Die Beherrschung einer Programmiersprache ist keine Garantie für „gute“ Programme. So wie es in der Programmierung Handbücher gibt, die dem Programmierer anleiten sauberen und vor allem wartungsarmen Code zu schreiben, so braucht auch ein Modellierer einen Handreichung wie seine Prozessmodelle zu gestalten sind, damit sie zum Beispiel leicht anpassbar werden. Das Einhalten von Konventionen und die Einführung von Abstraktionsschichten sowie die Anwendung von Lösungsmuster sind weit verbreiteten Methoden um „gute“ Prozessmodelle zu entwerfen.

Die Autoren in [CT09] haben eine Modellierungsmethodik entwickelt, deren Anwendung zu vollständigen und korrekten Geschäftsprozessen führen soll. Die Modellierung wird in drei Phasen unterteilt. In der ersten Phase werden mit Hilfe von „Design Rules“ alle für ein

Modell relevanten Informationen und Aufgaben aus der natürlich sprachlichen Beschreibung der Prozesse identifiziert. In der zweiten Phase werden ausgehend von den Modellen der ersten Phase Regeln der Verfeinerung und Reduktion angewendet. Die Anwendung der Regeln aus der zweiten Phase garantierten, dass die Modelle anschließend korrekt und ökonomisch sind. In der dritten Phase werden die Modelle serialisierbar gemacht.

Auch [All05] befasst sich mit Modellierungskonventionen, die aufgestellt werden, um festzuhalten welche Details auf welcher Hierarchiestufe modelliert werden sollen und wie eine gewisse Einheitlichkeit erreicht werden kann.

Eine detaillierte Handreichung zur Methodik der Modellierung von Geschäftsprozessen in verschiedenen Abstraktionsschichten findet sich bei [Sil09]. Der Autor schlägt eine hierarchische Modellierung in Schichten vor: „Level 1 is limited to a basic working set of shapes and symbols familiar to business – similar to the new Descriptive conformance class of BPMN 2.0 – and useful for process documentation and qualitative analysis. Level 2 adds support for some additional elements, most notably intermediate events – the palette is similar to the Analytical conformance class of BPMN 2.0 – and represents the common process language shared by business and IT. It describes in precise detail the activity flow of a business process, but by itself it is not executable. Executable BPMN is what I call Level 3." [Sil]

Obwohl Modellierungskonventionen ein wichtiger Bestandteil bei der Gestaltung von Prozessmodellen sind, braucht es Werkzeuge und Techniken, die den Modellierer bei seiner Arbeit unterstützen. Zumal mit der mit Zunahme der Prozesskomplexität eine automatische Unterstützung unverzichtbar ist.

4.6 Synchronisierung von Prozessmodellen

[WBMW] führt an, dass eine automatische Synchronisierung von Prozessmodellen unrealistisch zu sein scheint, sodass allein die Angabe der betroffenen Prozesse oder einer Prozessregion vorteilhaft wäre.

Die Autoren in [WWM09] stellen ein Konzept zur Angabe einer Änderungsregion vor. Die Grundidee ist, graphenbasierte Prozessmodelle mit sogenannten „Behavioral Profiles" zu charakterisieren. Vereinfacht gesagt sind „Behavioral Profiles" eine Menge von Relationen auf der Knotenmenge der Prozessgraphen, die die Reihenfolge der Prozessschritte charakterisieren. Hinzu kommt eine Relation zwischen den Elementen der Prozessmodelle. Ändert sich ein Prozessmodell, lässt sich anhand der angeführten Charakteristika und einem Reduktionsalgorithmus ein Teilgraph in dem anderen Modell identifizieren, in denen die Änderung vollzogen werden kann.

Das vorgeschlagene Konzept aus [WWM09] kennzeichnet eine Region, in der die Änderung übernommen werden kann. Es macht aber auch gleichzeitig deutlich, dass eine automatische Änderungsübernahme wie man sie sich gemeinhin bei der Modellierung in mehreren Abstraktionsschichten wünscht, sehr schwierig, wenn nicht unmöglich ist.

4.7 Konsistenz zwischen Prozessmodellen

[LRF10] untersucht die Konsistenz zwischen verschiedenen Prozessmodellen. Die Autoren gehen der Frage nach, wie überprüft werden kann, ob die Verfeinerung eines Modells konsistent zum ursprünglichen Modell ist. Dazu schlagen sie zwei Ansätze vor: Validierung der Verfeinerung durch Petri-Netze und durch OWL-DL [3] Ontologien. Die Idee beim ersten Ansatz ist es, die Ausführung des abstrakten Prozess zu simulieren und den verfeinerten Prozess nachlaufen zu lassen. Dazu konstruiert man ein Petri-Netz beider Prozesse derart, dass es zu einem Deadlock kommt, wenn die Verfeinerung invalide ist. Im zweiten Fall werden die Modelle in OWD DL Ontologien übersetzt und anschließend miteinander verglichen.

[SEL+10] reichert Relationen zwischen Prozessmodellen mit einer Semantik an. Konsistenz kann überprüft werden, indem die Relationen auf Verletzungen von Konsistenzmerkmalen geprüft werden: (V1) Existenz von ungebundenen Aktivitäten, (V2) Verletzung der Ausführungsreihenfolge und (V3) fehlende Aktivitäten.

In [RGL+09] wird aus einem Prozessmodell ein sogenanntes „Exectution Set“ abgeleitet. Das sind vereinfacht gesagt, alle möglichen Pfade vom Start- bis zum Endknoten eines Prozessgraphen. Da der Vergleich aller Ausführungspfade ineffizient ist, werden diese durch Reduktion in Ontologien transformiert. Die Validierung einer Verfeinerung kann dann als eine Prüfung auf die Erfüllbarkeit eines Konzepts in einer Ontologie angesehen werden.

[3] Für weitere Informationen zur Web Ontology Language siehe http://www.w3.org/TR/owl-guide/

5 Modellierungskonzept

In diesem Kapitel wird das Modellierungskonzept für adaptive Prozessmodellierung in mehreren Abstraktionsschichten vorgestellt. Am Anfang steht eine allgemeine Vorstellung des Konzepts, gefolgt von den notwendigen Definitionen und Notationen für die formale Beschreibung von Prozessmodellen, Abstraktionsschichten und der Modellzuordnung. Im weiteren Verlauf werden Änderungsoperationen eingeführt und die Synchronisierung erklärt. Am Ende dieses Kapitels wird auf das Problem der Semantik eingegangen.

5.1 Vorgehensweise

Zur Veranschaulichung der Vorgehensweise gehen wir von zwei gegebenen Prozessmodellen aus. Zuerst muss eine eindeutige Beziehung zwischen den Aktivitäten der Modelle hergestellt werden. Das lässt sich beispielsweise mit Hilfe mathematischer Relationen realisieren. Für diesen Zweck fassen wir die Prozessmodelle als Prozessgraphen auf und können so die Beziehungen zwischen den Modellelementen als Relationen auf den Knotenmengen ausdrücken.

Als nächstes braucht es ein Konzept wie man mit Änderungen in den Prozessmodellen umgeht. Die Idee ist, Änderungen in Änderungsoperationen zu kategorisieren und ihnen bestimme Merkmale wie zum Beispiel Typ, Datum sowie die Menge der Modellelemente auf denen sie operieren, zuzuweisen. Änderungen können so in Form von diesen Operationen protokolliert werden.

Die Synchronisierung der Prozessmodelle lässt sich dann als Aufzeichnung und Weitergabe von Änderungsoperationen durchführen. Jede Änderungsoperation wird einem Änderungsprotokoll zugeführt, das den verwandten Prozessmodellen zugänglich gemacht wird. Der Modellierer des verwandten Prozessmodells kann anschließend das Änderungsprotokoll in seinem eigenen Prozess abarbeiten. Wurde beispielsweise eine Aktivität hinzugefügt, kann er auch eine Aktivität hinzufügen oder aber eine bestehende Aktivität der neu hinzugefügten Aktivität zuordnen. Seine Änderungen werden wiederum seinem Änderungsprotokoll zugeführt und an weitere verwandte Prozessmodelle weitergereicht.

Zwei Prozessmodelle sind dann synchron, wenn alle Modellelemente zueinander in Relation stehen und alle Änderungen übernommen wurden. In Abbildung 5.1 findet sich eine Veranschaulichung des vorgestellten Konzepts.

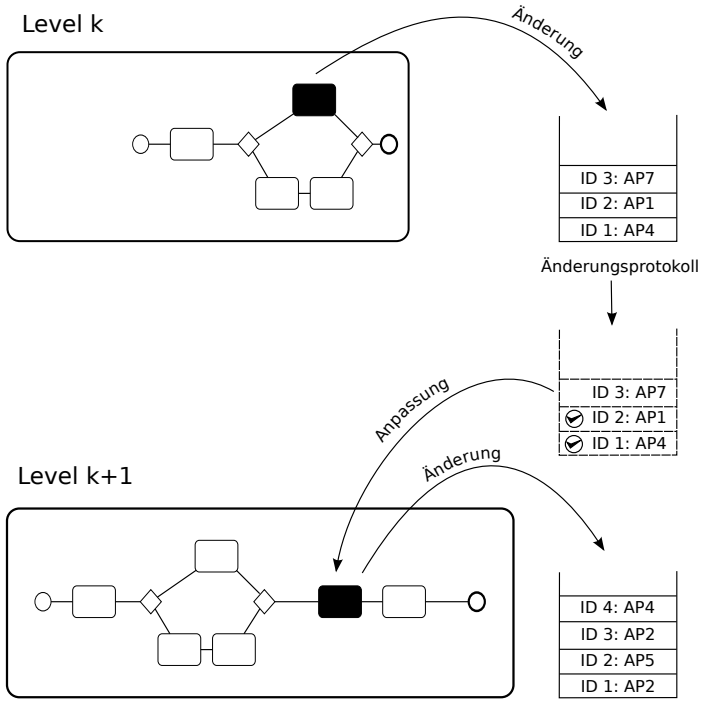

Abbildung 5.1: Konzept zur adaptiven Modellierung in mehreren Abstraktionsschichten

5.2 Prozessgraphen

Im Sinne der Allgemeinheit abstrahieren wir von aktuellen Modellierungssprachen und definieren ein Prozessmodell als einen gerichteten Graphen. Der Zweck dieser Definition ist die Herstellung von Beziehungen zwischen Prozessmodellen. Um später Änderungen an einer Aktivität so genau wie möglich zu lokalisieren, unterscheiden wir die Knotenmenge in Gateways und Aktivitäten.

Prozessgraph Ein Prozessgraph ist ein 7-Tupel $P = (\mathcal{A}, \mathcal{G}, \mathcal{E}, \mathcal{T}_\mathcal{A}, \mathcal{T}_\mathcal{G}, t_\mathcal{A}, t_\mathcal{G})$ mit

- $\mathcal{A}$ Menge der Aktivitäten
- $\mathcal{G}$ Menge der Gateways
- $\mathcal{E} \subseteq (\mathcal{A} \cup \mathcal{G}) \times (\mathcal{A} \cup \mathcal{G})$ Kantenmenge
- $\mathcal{T}_\mathcal{A}$ Typmenge der Aktivitäten
- $\mathcal{T}_\mathcal{G}$ Typmenge der Gateways
- Typzuordung $t_\mathcal{A} : \mathcal{A} \to \mathcal{T}_\mathcal{A}$
- Typzordnung $t_\mathcal{G} : \mathcal{G} \to \mathcal{T}_\mathcal{G}$

BPMN-Prozessgraph Ein BPMN-Prozessgraph ist ein Prozessgraph $P = (\mathcal{A}, \mathcal{G}, \mathcal{E}, \mathcal{T}_\mathcal{A}, \mathcal{T}_\mathcal{G}, t_\mathcal{A}, t_\mathcal{G})$ mit

- $\mathcal{T}_\mathcal{A} = BPMN_\mathcal{A} = \{$ Aktivität, Aktivität[Schleife], Aktivität[Teilprozess], Aktivität[Parallele Mehrfachaufführung], Aktivität[Sequentielle Mehrfachausführung], Aktivität[Ad Hoc], Aktivität[Kompensation], Aktivität[Senden], Aufgabe, Aufgabe[Benutzer], Aufgabe[Senden], Aufgabe[Empfangen], Aufgabe[Manuell], Aufgabe[Geschäftsregel], Aufgabe[Service], Aufgabe[Skript] $\}$
- $\mathcal{T}_\mathcal{G} = BPMN_\mathcal{G} = \{$ Gateway, Gateway[Parallel], Gateway[Exklusiv], Gateway[Inklusiv], Gateway [Komplex], Gateway[Ereignis], Gateway[Ereignis][Exklusiv], Gateway[Ereignis][Parallel] $\}$

5.3 Abstraktionsschichten

In Anlehnung an Abschnitt 2.2 werden Abstraktionsschichten eingeführt. Die Auswahl der Elemente für die Abstraktionsschichten kann je nach Aufgabe des Modells unterschiedliche ausfallen.

Abstraktionsschicht $\mathcal{L}$ Eine Abstraktionsschicht $\mathcal{L}$ ist eine Menge von Prozessmodellen, deren Modellelemente eine echte Teilmenge zu den Modellelementen des Meta-Modells bilden.

BPMN-Abstraktionsschicht $\mathcal{L}_{\mathcal{BPMN}}$ Eine BPMN-Abstraktionsschicht $\mathcal{L}_{\mathcal{BPMN}}$ ist eine Abstraktionsschicht für deren Elemente $P = (\mathcal{A}, \mathcal{G}, \mathcal{E}, \mathcal{T}_\mathcal{A}, \mathcal{T}_\mathcal{G}, t_\mathcal{A}, t_\mathcal{G})$ gilt: $\mathcal{T}_\mathcal{A} \subset BPMN_\mathcal{A}$ und $\mathcal{T}_\mathcal{G} \subset BPMN_\mathcal{G}$.

5.4 Modellangleich

Ein Modellangleich setzt die Elemente zweier Prozessgraphen in Beziehung. Es handelt sich um eine mathematische Relation auf der Knotenmenge der Prozessgraphen. Die Zuordnung der Aktivitäten durch eine Funktion genügt nicht den Anforderungen aus Abschnitt 3.1 [1].

Wie schon in Kapitel 3 - Anforderungen gezeigt, ist es schwer die Aktivitäten automatisch zu verbinden. Der Modellierer muss also manuell die Verknüpfungen zwischen den Aktivitäten herstellen.

Für die Herstellung von Beziehungen zwischen den Prozessmodellen werden nur Aktivitäten und nicht Gateways berücksichtigt, weil „Gateways nur die Logik eines Geschäftsprozesses repräsentieren, das heißt sie führen keine Aktionen aus und benötigen keine Zeit für ihren Durchlauf“ [All10]. Gateways würden die manuelle Zuordnung nur erschweren und brächten keinen wirklichen Mehrwert.

[1] Diese Funktion hätte auf Potenzmengen operieren müssen.

Modellzuordnung Eine Modellzuordnung zwischen zwei Prozessgraphen ist ein 4-Tupel $Z(P_1, P_2, R, U)$ mit

- $P_1 = (\mathcal{A}_1, \mathcal{G}_1, \mathcal{E}_1, \mathcal{T}_{\mathcal{A}_1}, \mathcal{T}_{\mathcal{G}_1}, t_{\mathcal{A}_1}, t_{\mathcal{G}_1})$ und $P_2 = (\mathcal{A}_2, \mathcal{G}_2, \mathcal{E}_2, \mathcal{T}_{\mathcal{A}_2}, \mathcal{T}_{\mathcal{G}_2}, t_{\mathcal{A}_2}, t_{\mathcal{G}_2})$ Prozessgraphen
- $(X, Y) \in R \subseteq (\mathcal{A}_1 \times \mathcal{A}_2) \cup (\mathcal{A}_2 \times \mathcal{A}_1) \Leftrightarrow$ Aktivität X steht in Beziehung zu Aktivität Y
- $U \subseteq (\mathcal{A}_1 \cup \mathcal{A}_2)$ Menge der Aktivitäten, die zu keiner Aktivität in Beziehung stehen

5.5 Änderungsoperationen

Die hier aufgelisteten Operationen orientieren sich an die in [WRRM08] genannten „Adaptation Patterns“ (siehe Abbildung 4.9 auf Seite 30). Da der Modellangleich keine Gateways berücksichtigt, fallen die Adaptionsmuster zu den Kontrollflussänderungen (AP9-AP13) und den Übergangsbedingungen (AP14) weg. Änderungen an Unterprozessen (AP7-AP8) werden durch den Modellangleich und die Abstraktionsschichten selbst abgedeckt.

Eine Änderungsoperation besteht aus (1) einer Beschreibung, (2) einer Vorbedingung, die vor der Anwendung der Operation erfüllt sein muss, (3) der eigentlichen Änderung der Modellzuordnung Z, (4) einer Menge V der veränderten Aktivitäten, die von der Änderung betroffen sind, (5) einer Ausgabe für das Protokoll und (6) einer Nachbedingung, die nach der Anwendung der Operation erfüllt sein muss.

5.5.1 Primitive Änderungsoperationen

Jede Änderung an einem Prozessgraphen kann durch die Hintereinanderausführung der folgenden primitiven Änderungsoperationen simuliert werden:

- Knoten hinzufügen
- Kante hinzufügen
- Knoten löschen
- Kante löschen

Die Aufnahme solcher primitiver Änderungsoperationen in das Änderungsprotokoll ist aber nicht zweckmäßig. Dem Modellierer muss im Änderungsprotokoll mitgeteilt werden *welche* Änderung vorgenommen wurde und nicht *wie* sie vorgenommen wurde.

5.5.2 Komplexe Änderungsoperationen

Komplexe Operationen bestehen aus der Hintereinanderausführung von primitiven Operationen. Das Löschen einer Aktivität ist insofern komplex, als dass man die ein- und ausgehenden Kanten des Knotens, sowie den Knoten selbst löschen muss. Im Konzept werden nur komplexe Änderungsoperationen behandelt, da aus ihnen ersichtlich wird *welche* Änderung stattgefunden hat. Folgende komplexe Änderungsoperationen wurden berücksichtigt:

Hinzunahme einer Aktivität Aktivität X vom Typ Y wird dem Prozessmodell hinzugefügt. Diese Operation kann primitiv oder komplex sein. Im zweiten Fall werden auch die ein- und ausgehenden Kanten zugefügt: Tabelle 5.1

Wegnahme einer Aktivität Aktivität X wird vom Prozessmodell entfernt. Mit ihr werden auch alle ein- und ausgehenden Kanten entfernt: Tabelle 5.2

Verschiebung einer Aktivität Aktivität X wird im Prozessgraphen an eine andere Stelle verschoben. Das beinhaltet auch die Modifikation der zugehörigen Kanten: Tabelle 5.3

Ersetzung einer Aktivität Eine Aktivität wird durch eine neue Aktivität ersetzt. Im Grunde ist das die Hintereinanderausführung der ersten und zweiten Änderungsoperation. Ohne diese Änderungsoperation würde nicht unmittelbar ersichtlich werden, warum zuerst eine Aktivität gelöscht und dann eine neue hinzugefügt wurde: Tabelle 5.4

Vertauschung zweier Aktivitäten Zwei Aktivitäten werden miteinander vertauscht. Werden die Aktivitäten allein durch ihre Bezeichner unterschieden, reicht es nur die Bezeichnung zu vertauschen. In allen anderen Fällen müssen die Knoten der Aktivitäten tatsächlich vertauscht werden: Tabelle 5.5

Typänderung einer Aktivität Der Typ einer Aktivität ändert sich. So kann zum Beispiel ein Task in einen Unterprozess umgewandelt werden: Tabelle 5.6

Hinzunahme einer Aktivität	
Beschreibung	Benutzer fügt Aktivität X zum Prozessgraphen hinzu.
Vorbedingung	$X \notin (\mathcal{A}_1 \cup \mathcal{A}_2)$
Zuordnungsänderung	Füge X zur Menge U hinzu.
Veränderte Aktivitäten	V = X
Protokoll	`AKTIVITAET X VOM TYP T WURDE EINGEFUEGT`
Nachbedingung	$X \in (\mathcal{A}_1 \cup \mathcal{A}_2)$ und $X \in U$

Tabelle 5.1: Änderungsoperation: Hinzunahme einer Aktivität

Wegnahme einer Aktivität	
Beschreibung	Benutzer entfernt Aktivität X vom Prozessgraphen
Vorbedingung	$X \in (\mathcal{A}_1 \cup \mathcal{A}_2)$
Zuordnungsänderung	Entferne alle Tupel (X,Y) und (Y,X) aus R und füge alle Aktivitäten Y, die mit keiner Aktivität verbunden sind, zur Menge U hinzu
Veränderte Aktivitäten	V = X
Protokoll	`AKTIVITAET X WURDE ENTFERNT`
Nachbedingung	$X \notin (\mathcal{A}_1 \cup \mathcal{A}_2)$ und $\{Y \mid !\exists X : (X,Y) \in R \vee (Y,X) \in R\} \subseteq U$

Tabelle 5.2: Änderungsoperation: Wegnahme einer Aktivität

Verschiebung einer Aktivität	
Beschreibung	Benutzer verschiebt Aktivität X im Prozessgraphen
Vorbedingung	$X \in (\mathcal{A}_1 \cup \mathcal{A}_2)$
Zuordnungsänderung	-
Veränderte Aktivitäten	V = X
Protokoll	`AKTIVITAET X WURDE VERSCHOBEN`
Nachbedingung	$X \in (\mathcal{A}_1 \cup \mathcal{A}_2)$

Tabelle 5.3: Änderungsoperation: Verschiebung einer Aktivität

Ersetzung einer Aktivität	
Beschreibung	Benutzer ersetzt Aktivität X mit Aktivität Y
Vorbedingung	$X \in (\mathcal{A}_1 \cup \mathcal{A}_2)$ und $Y \notin (\mathcal{A}_1 \cup \mathcal{A}_2)$
Zuordnungsänderung	Ersetze alle Tupel (X,Z) und (Z,X) aus R durch (Y,Z) bzw. (Z,Y)
Veränderte Aktivitäten	V = $\{X, Y\}$
Protokoll	`AKTIVITAET X WURDE DURCH AKTIVITAET Y VOM TYP(Y)ERSETZT`
Nachbedingung	$X \notin (\mathcal{A}_1 \cup \mathcal{A}_2)$ und $Y \in (\mathcal{A}_1 \cup \mathcal{A}_2)$

Tabelle 5.4: Änderungsoperation: Ersetzung einer Aktivität

Vertauschung zweier Aktivitäten	
Beschreibung	Benutzer vertauscht Aktivität X und Aktivität Y
Vorbedingung	$X, Y \in (\mathcal{A}_1 \cup \mathcal{A}_2)$
Zuordnungsänderung	Vertausche alle Tupel (X,Z) und (Z,X) durch (Y,Z) bzw. (Z,Y) und alle Tupel (Y,W) und (W,Y) durch (X,W) bzw. (W,X)
Veränderte Aktivitäten	V = $\{X, Y\}$
Protokoll	`AKTIVITAET X WURDE MIT AKTIVITAET Y VERTAUSCHT`
Nachbedingung	$X, Y \in (\mathcal{A}_1 \cup \mathcal{A}_2)$

Tabelle 5.5: Änderungsoperation: Vertauschung zweier Aktivitäten

Typänderung einer Aktivität	
Beschreibung	Benutzer ändert den Typ einer Aktivität X
Vorbedingung	$X \in (\mathcal{A}_1 \cup \mathcal{A}_2)$
Zuordnungsänderung	-
Veränderte Aktivitäten	V = X
Protokoll	`AKTIVITAET X WURDE AUF TYP T GEAENDERT`
Nachbedingung	$X \in (\mathcal{A}_1 \cup \mathcal{A}_2)$

Tabelle 5.6: Änderungsoperation: Typänderung einer Aktivität

5.6 Synchronisierung zwischen Prozessmodellen

Ändert ein Modellierer sein Prozessmodell, so werden die dazugehörigen Operationen in aufsteigender Nummerierung auf einem Stapel abgelegt. Möchte ein Modellierer die Änderungen an einem verknüpften Prozessmodells auf sein eigenes Prozessmodell adaptieren, so geht er Operation für Operation in aufsteigender Reihenfolge durch den verknüpften Stapel und übernimmt die Änderungen.

Eine Änderungsoperation gilt als übernommen, wenn alle Aktivitäten der Menge V in einer Relation sind, das heißt nicht in der Menge U enthalten sind. Die Abarbeitung kann jederzeit unterbrochen und später wieder aufgenommen werden. Zwei Prozessmodelle gelten als miteinander synchron, wenn alle Änderungsoperationen übernommen worden sind und die Menge U leer ist.

Änderungsliste S Ein Änderungsliste S ist eine Liste von aufsteigend nummerierten Änderungsoperationen.

Übernahme einer Änderungsoperation Eine Änderungsoperation gilt als übernommen, wenn die Menge V der veränderten Aktivitäten in einer Relation stehen.

Synchronitätszahl Wurden aus einer Liste von n Änderungsoperationen d Operationen übernommen, beträgt die Synchronitätszahl s = n-d.

Synchronität zwischen zwei Prozessmodellen Sei eine Modellzuordnung $Z(P_1, P_2, R, U)$ gegeben. Zwei Modelle gelten als miteinander synchron, wenn $U = \emptyset$ und s=o gilt.

5.7 Herausforderung Semantik

Das vorgestellte Konzept ignoriert bewusst eine semantische Überprüfung der Relation zweier Prozessmodelle. Dazu müsste es einen Weg geben, zu *wissen* was die Aktivitäten im Kontext des Geschäftsprozesses bedeuten. Zwei graphenbasierte Prozessmodelle können zum Beispiel strukturell völlig übereinstimmen und trotzdem zwei vollkommen verschiedene Geschäftsprozesse beschreiben. Auf der anderen Seite können Prozessmodelle, die sich strukturell unterscheiden, ein und denselben Prozess beschreiben.

Es gibt, wie in Kapitel 4 - Verwandte Arbeiten schon beschrieben, eine Vielzahl von Arbeiten, die sich mit dem Grad der Ähnlichkeit von Prozessmodellen und der semantischen Zuordnung von Modellelementen beschäftigen. Diese Arbeiten eignen sich ausgezeichnet, um vorhandene Modelle zu analysieren, nach ihnen in Prozessdatenbanken zu suchen, Zusammenhänge und Unterschiede zwischen Modellen festzustellen und aufzuzeigen, Modelle ineinander zu überführen oder Prozessverfeinerungen nach einer Semantik zu validieren.

Damit ließe sich die Übernahme einer Änderung in dem vorgestellten Konzept zum bestimmten Teil validieren. So könnte ein System den Modellierer warnen, wenn sich die Reihenfolge der Prozessschritte ändert (Abbildung 5.2) oder die verbundenen Aktivitäten auf verschiedenen Ausführungspfaden liegen (Abbildung 5.3). Allgemein könnte man Regeln der Verfeinerung aufstellen, die den Modellierer vor groben Fehlern warnen. Es ist jedoch weder möglich die Übernahme zu automatisieren noch eine Verfeinerung auf ihre *Zweckmäßigkeit* zu prüfen.

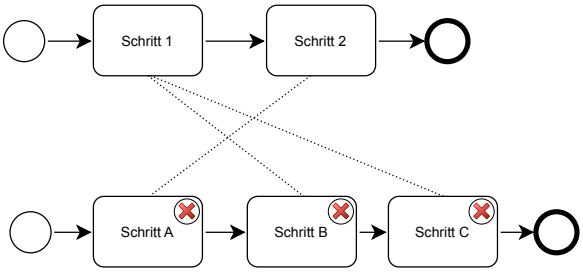

Abbildung 5.2: Die Reihenfolge der Prozessschritte hat sich in der Verfeinerung geändert.

Darüber hinaus wäre es auch denkbar, die Verfeinerungsmöglichkeiten eines Prozesses einzuschränken, um eine gewisse Konsistenz zu erzwingen. Diese Einschränkungen können die Verfeinerung oder die Zuordnungsrelation betreffen. Im folgenden ein paar Beispiele solcher Einschränkungen:

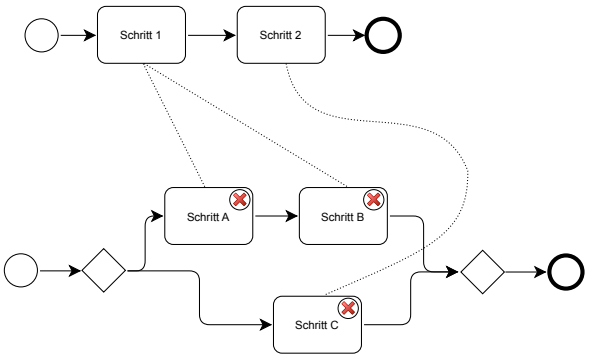

Abbildung 5.3: Simuliert man beide Prozess kann in der Verfeinerung der Schritt C wegfallen, im abstrakten Prozess wird Schritt 2 auf jeden Fall ausgeführt.

- Aktivität A vom Typ X darf nur von n Aktivitäten vom Typ Y verfeinert werden.
- Die Verfeinerung von Aktivität A muss das Modellelement Y enthalten.
- Ist $[..., A, ..., B, ...]$ ein Pfad vom Startknoten zum Endknoten eines Prozessgraphen und $(A, C) \in R$ und $(B, D) \in R$ dann muss es mindestens einen Pfad $[..., C, ..., D, ...]$ in verwandten Prozessgraphen geben.

Diese automatischen Prüfungen können den Modellierer bei seiner Arbeit lediglich unterstützen, ihn aber nicht davon abhalten falsche Modelle und Modellverfeinerungen zu entwerfen. Es ist nicht Ziel des Konzepts korrekte Modelle zu erzwingen, sondern dem Modellierer die Möglichkeit geben, schnell auf Anpassungen zu reagieren. Das schließt ein, dass er genügend Informationen zu einer Änderung zur Verfügung gestellt bekommt, um schließlich in seinem Ermessen die Anpassung vorzunehmen.

6 Implementierung

Gemäß Aufgabenstellung wird zur Validierung des erarbeiteten Konzepts ein Prototyp auf der Basis des OpenSource-Editor Oryx implementiert. Es folgt eine kurze Einführung in den Editor mit anschließender Beschreibung der Architektur des Prototypen. Anschließend wird auf die Umsetzung des Konzepts durch die Funktionen des Prototypen eingegangen.

6.1 Einführung Oryx

Oryx ist ein web-basierter Editor mit dem man Diagramme in verschiedenen Modellierungssprachen erstellen und anderen Benutzern zur Verfügung stellen kann. Oryx wurde am Hasso-Plattner-Institut in Potsdam entwickelt und steht unter der MIT Lizenz [1] [WWW].

Die Benutzeroberfläche des Editors ist in vier Bereiche eingeteilt. Im oberen Bereich befindet sich die Werkzeugleiste, im linken die Modellierungselemente des ausgewählten Diagrammtyps, in der Mitte die Modellierungsfläche und rechts die Eigenschaftspalette des aktuell ausgewählten Elements. Abbildung 6.1 zeigt die Benutzeroberfläche und Abbildung 6.2 die grobe Architektur des Editors.

Der Oryx-Editor besteht aus folgenden Hauptkomponenten:

Data Manager „Der Data Manager ist für den Zugriff und die Persistierung der Prozessdaten verantwortlich. Er stellt eine eigene Schicht in der Architektur dar, die auch von anderen Anwendungen benutzt werden kann. Er kapselt somit den Zugriff auf die Daten und stellt eine spezielle API dafür bereit. “ [Tsc07]

Core Diese Komponente setzt die Darstellung der Zeichenfläche und die grafischen Elementen des gewählten Diagrammtyps um. Außerdem beschreibt sie den Zugriff auf die Stencil Sets und den dazugehörigen SVG-Elementen.

Plugins Es gibt zwei Arten von Plugins: Client- und Server-Plugins. Client-Plugins erweitern die Funktionalität des Editors im Browser und sind in JavaScript implementiert. Sie greifen über eine Fassade auf die Funktionen des Editors zu und können so beispielsweise die Zeichenfläche oder die Werkzeugleiste manipulieren. Server-Plugins sind Java Servlets, die von clientseitigen Plugins über HTTP-Anfragen aufgerufen werden können.

[1] Für nähere Informationen zur MIT-Lizenz siehe http://www.opensource.org/licenses/mit-license.php

Editor „Die Editor Komponente ist für die Initialisierung der Zeichenfläche, der Plugins, der Stencil Sets und der Prozessmodelle verantwortlich. Den Plugins wird vom Editor eine Schnittstelle bereitgestellt, die es den Plugins ermöglicht, auf Kernfunktionen des Editors zuzugreifen. So lassen sich z.B. über diese Schnittstelle neue Modellelemente erzeugen oder vorhandene löschen.“ [Tsc07]

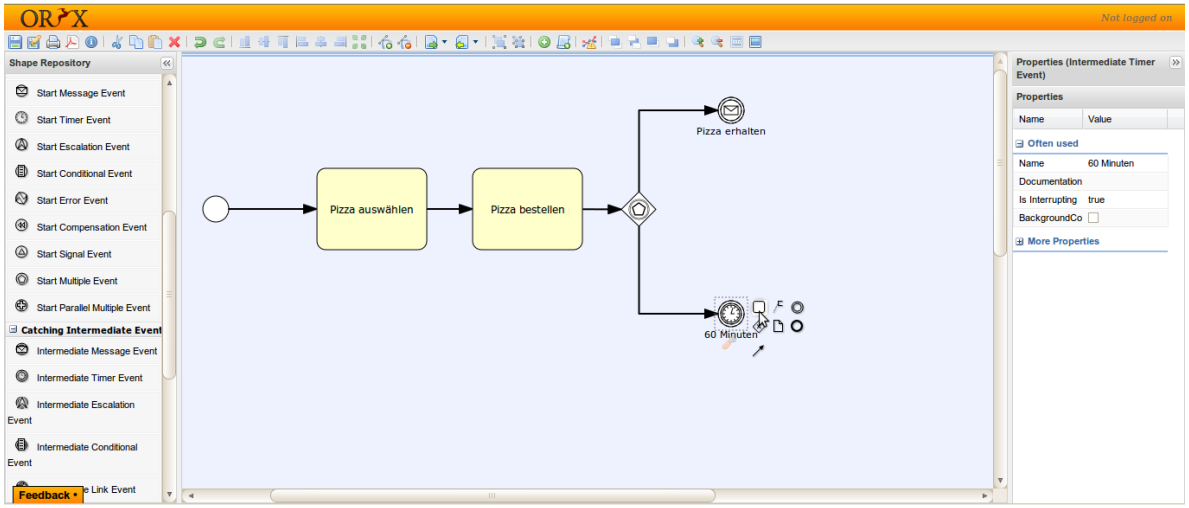

Abbildung 6.1: Benutzeroberfläche des Oryx-Editors

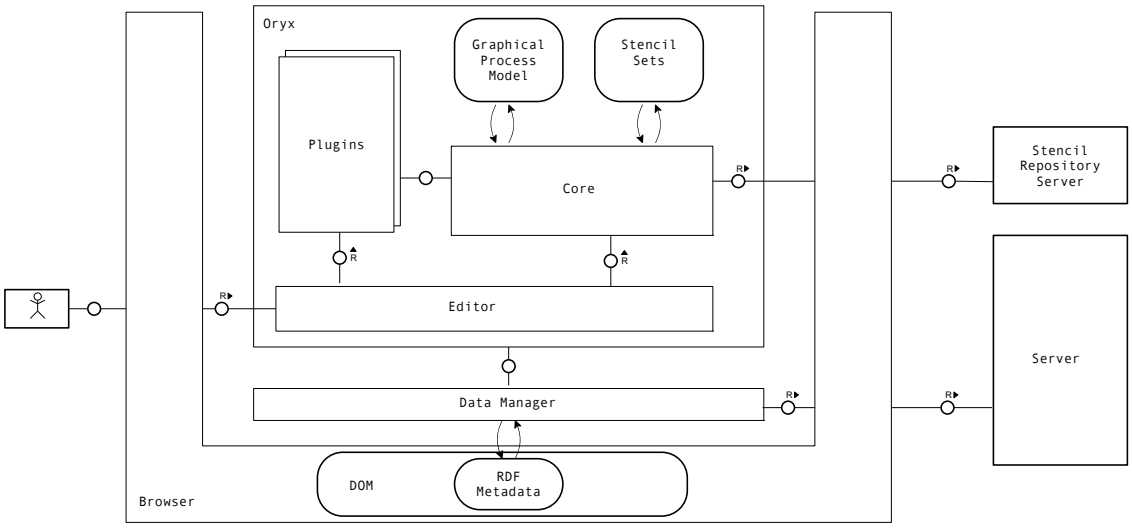

Abbildung 6.2: Architektur des Oryx-Editors nach [Tsc07]

Stencil Sets

Ein Stencil Set ist eine Menge von Stencils. Ein Stencil ist eine Beschreibung der grafischen Modellierungselmente eines Diagrammtyps. Man unterscheidet zwei Typen von Stencils: Node und Edge. Node sind die Elemente eines Diagrammtyps, die verbunden werden können und Edge sind die verbindenden Elemente. In Listing 6.1 ist ein Auszug einer Stencil-Beschreibung zu sehen. Ein Stencil wird durch Attribute wie ID, Typ, Titel usw. festgelegt und hat beliebige frei definierbare Eigenschaften. Die SVG-Datei im Verweis `view` gibt dem Stencil sein Aussehen.

Listing 6.1 Stencil Beschreibung in Oryx

```
{
                "type" : "node",
                "id": "node1",
                "title": "Node",
                "title_de" : "Knoten",
                "description" : "Example node.",
                "description_de" : "Beispiel-Knoten.",
                "view" : "node.svg",
                "icon" : "node.png",
                "roles" : [
                        "connection_node"
                ],
                "properties": [
                        {
                                "id":"name",
                                "type":"String",
                                "title":"Name",
                                "value":"",
                                "description":"Name of node.",
                                "description_de":"Name des Knotens.",
                                "readonly":false,
                                "optional":true,
                                "length":"",
                                "wrapLines" : true,
                                "refToView":"text_name"
                        },
                        ...
                ]

}
```

Nachrichtenkonzept in Oryx

Plugins kommunizieren über ein Nachrichtenkonzept miteinander. Eine Nachricht besteht aus einem Typ und einem beliebigen Inhalt. In Listing 6.2 wird die Callback-Funktion `alignmentComplete` an die Nachricht vom Typ `element_aligned` registriert. So kann Plugin A mit `raiseEvent` eine Nachricht erzeugen und Plugin B mit `registerOnEvent` die Nachricht abfangen und seine Callback-Funktion aufrufen.

Listing 6.2 Nachrichtenkonzept des Oryx-Editors

```
...
this.facade.registerOnEvent("element_aligned", this.alignmentComplete.bind(this));
...
this.facade.raiseEvent({type: "element_aligned", element:this.selectedElement});
...
alignmentComplete : function (event){
                        switch(this.operationInProgress.type){
                        case "ADD":
                            var alignedElements =
                                JSON.parse(event.element.properties["oryx-alignedelements"]);
                            ...
}
```

6.2 Erweiterungen des Oryx-Editors für den Prototypen

In diesen Abschnitt wird kurz auf die Erweiterungen eingegangen, die am Oryx-Editor durchgeführt wurden, um das Konzept zu validieren. Weitere Details zu der Umsetzung folgen in den nächsten Abschnitten.

Stencil Set Erweiterungen

Die Stencil Set Erweiterungen beschreiben die Abstraktionsschichten und enthalten Datenstrukturen für den Modellangleich, für das Änderungsprotokoll und der Synchronisierung.

BPMN 2.0 / Level-1-3 Elements Stencil Sets für die 1.-3. Abstraktionsschicht.

Clientseitige Plugins

Der Orxy-Editor wurde um vier Plugins erweitert; jedes davon deckt eine Anforderung aus dem Kapitel 3 - Anforderungen ab. Im Folgenden werden die Funktionen der Plugins kurz erklärt. Abbildung 6.3 zeigt die Einordnung der Plugins in die Architektur von Oryx und Abbildung 6.4 die Werkzeugleiste der Plugins.

AlignmentWizard Dieses Plugin ist dafür zuständig, die Prozesse und ihre Elemente miteinander zu verbinden. Zusätzlich bietet es die Möglichkeit zu überprüfen, ob alle Elemente verknüpft sind.

ChangeQueue Dieses Plugin schreibt, liest, und bearbeitet das Änderungsprotokoll. Es implementiert die Änderungsoperationen und fügt diese dem Änderungsprotokoll hinzu.

SynchronisationWizard Dieses Plugin lädt das Änderungsprotokoll des verbundenen Prozessmodells und ermöglicht seine Abarbeitung. Es synchronisiert somit die Modelle.

Serverseitige Plugins

SVG-Alignment Dieses serverseitiges Plugin lädt aus dem Repository die SVG-Datei eines Prozessmodells und markiert je nach Eingabeparameter diejenige Aktivitäten, die nicht verbunden sind.

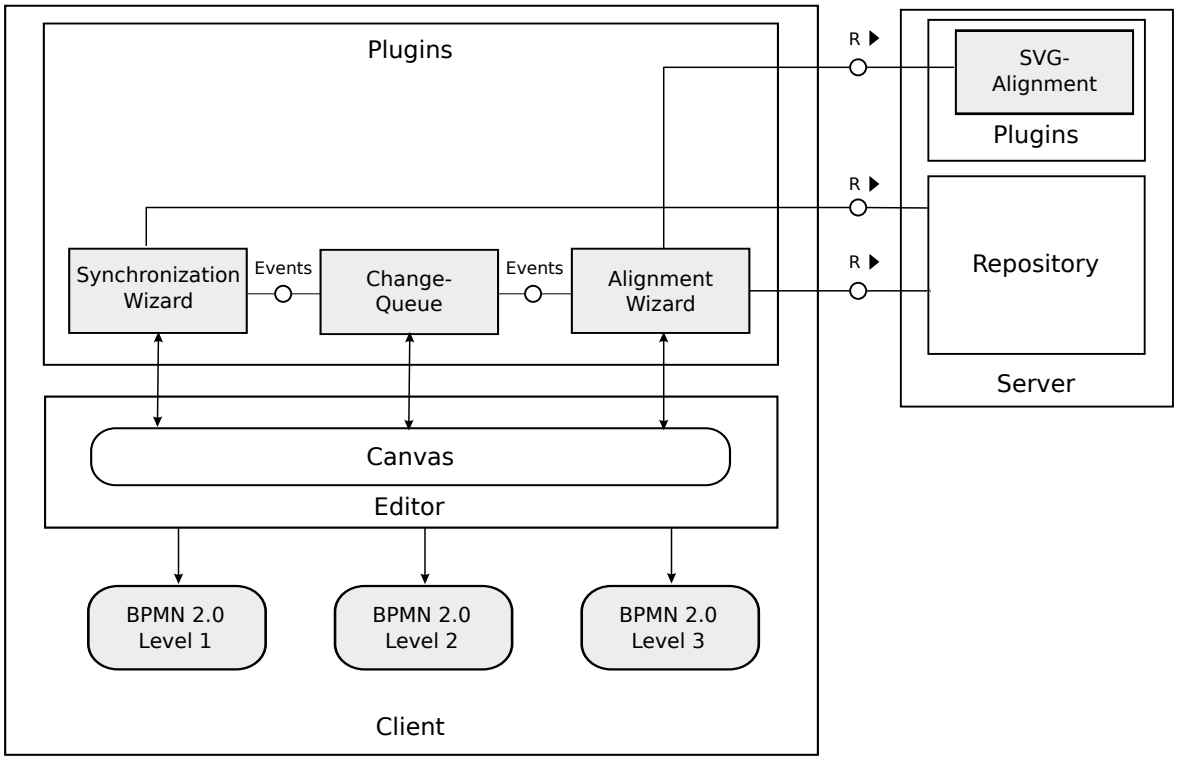

Abbildung 6.3: Architektur des Prototyps. Graue Artefakte sind Erweiterungen des Editors.

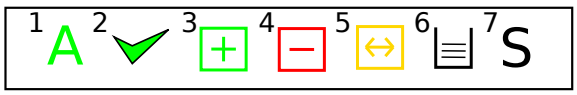

Abbildung 6.4: Plugin-Icons des Prototypen in Oryx: (1) Modellzuordnung, (2) Prüfung der Modellzuordnung, (3) Hinzunahme einer Aktivität, (4) Wegnahme einer Aktivität, (5) Vertauschung zweier Aktivitäten, (6) Änderungsprotokoll und (7) Synchronisation.

6.3 Umsetzung der Abstraktionsschichten

Auf Grundlage von praktischen Betrachtungen und Vorschlägen aus dem Abschnitt 4.1 - Abstraktionsschichten wird das vorgeschlagene Konzept mit drei Abstraktionsschichten umgesetzt.

Die Auswahl der Stencils der ersten Abstraktionsschicht orientiert sich an den Vorschlag von [Sha10] für die SIMPLE Klasse und den „Core Elements“ der Business Process Management Initiative [WWW05]. Das Stencil Set „BPMN 2.0 / Level-1 Elements“ ist eine Untermenge des BPMN 2.0 Stencil Sets, das bereits in Oryx integriert ist. Abbildung 6.5 zeigt alle Stencils der ersten Abstraktionsschicht.

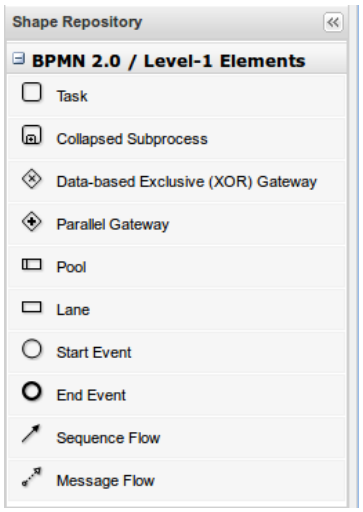

Abbildung 6.5: Level-1 Elemente der ersten Abstraktionsschicht

Die zweite Abstraktionsschicht ist eine Erweiterung der ersten und enthält zusätzliche Modellierungselemente; das zugehörige Stencil Set „BPMN 2.0 / Level-2 Elements“ ist dementsprechend auch eine Obermenge des ersten. Die Auswahl der Elemente für die zweite Abstraktionsschicht orientiert sich an die Klasse DESCRIPTIVE von [Sha10] und unterstützt bis auf Fehlerbehandlung die Level 2 Modellierung von [Silo9]. Abbildung 6.6 zeigt alle Stencils der zweiten Abstraktionsschicht. Die dritte Abstraktionsschicht enthält alle BPMN 2.0 Collaboration Elemente.

6.4 Modellierung des Modellangleichs

Jedes Stencil hat bestimmte Eigenschaften wie zum Beispiel ID, Typ, Hintergrundfarbe, Beschreibung. Für die Umsetzung des Modellangleichs wurden alle Task-Stencils zusätzlich mit der Eigenschaft `alignedElements` ausgestattet. In diesem Feld wird eine Liste von Prozesselementen des verbundenen Prozessmodells abgelegt.

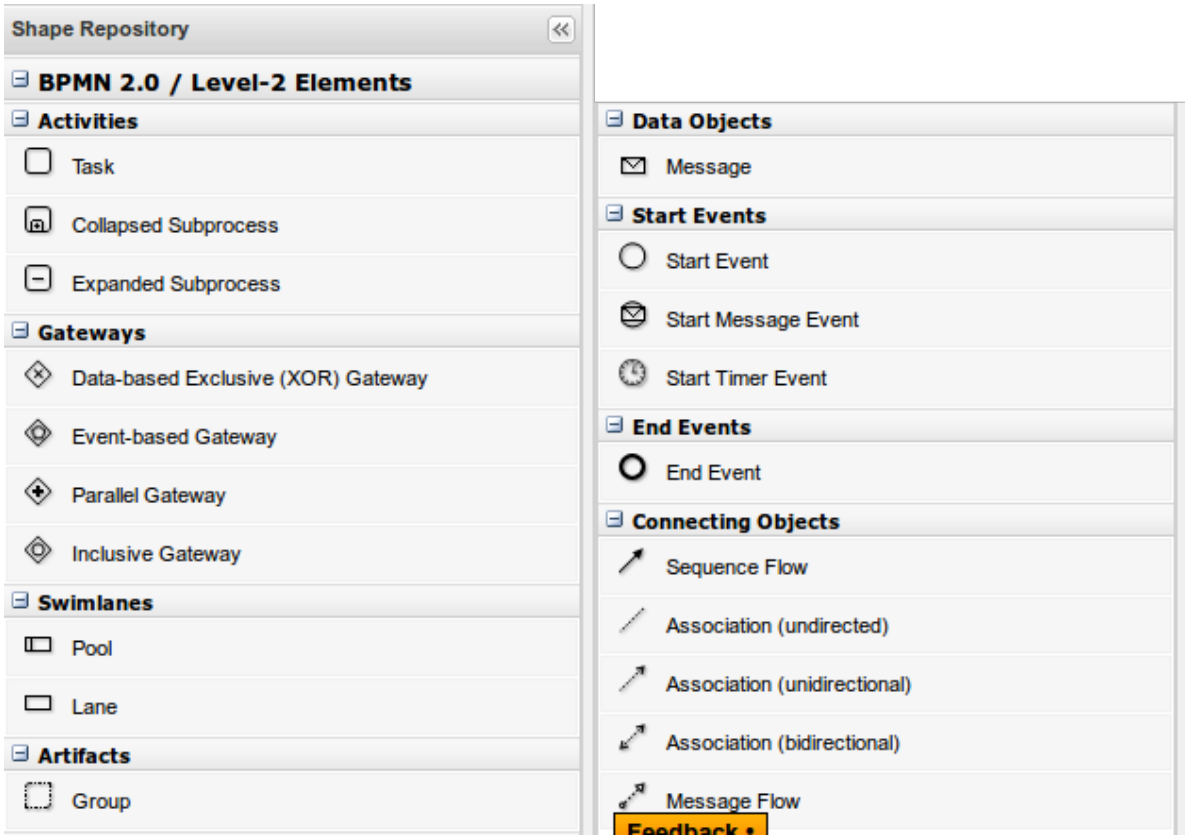

Abbildung 6.6: Level-2 Elemente der zweiten Abstraktionsschicht

Besitzt zum Beispiel Prozess A in Level-1 einen Task A1 mit der ID `oryx_123_ABCD` und einen Task A2 mit der ID `oryx_345_EFGH` und besitzt ein Prozess B in Level-2 einen Task B1, der mit den Tasks A1 und A2 des Prozesses A verbunden ist, dann steht im Eigenschaftsfeld `alignedElements` von B1 `{"totalCount":2,"items":[{"id":"oryx_123_ABCD"}, {"id":"oryx_345_EFGH"}]}`. Abbildung 6.7 macht diesen Zusammenhang deutlich.

In das Eigenschaftsfeld `alignedProcess` eines Prozessmodells wird die ID des verbundenen Prozesses eingetragen. Im Gegensatz zu `alignedElements` kann ein Prozess mit nur einem anderen Prozess verknüpft sein.

Die Wahl, verbundene Tasks in ein Eigenschaftsfeld zu schreiben und nicht eine separate Datenstruktur zu verwenden, begründet sich in der Einschränkung der Modelle durch den Oryx-Editor. Oryx persistiert nur Daten, die in den Stencil Set definiert wurden. Eine Modellzuordnung ist jedoch weder ein Knoten noch eine Kante (siehe Abschnitt 6.1), wodurch die Zuordnung als Eigenschaftsfelder der Task definiert wurde. Damit lässt sich eine n:m Beziehungen zwischen den Tasks realisieren.

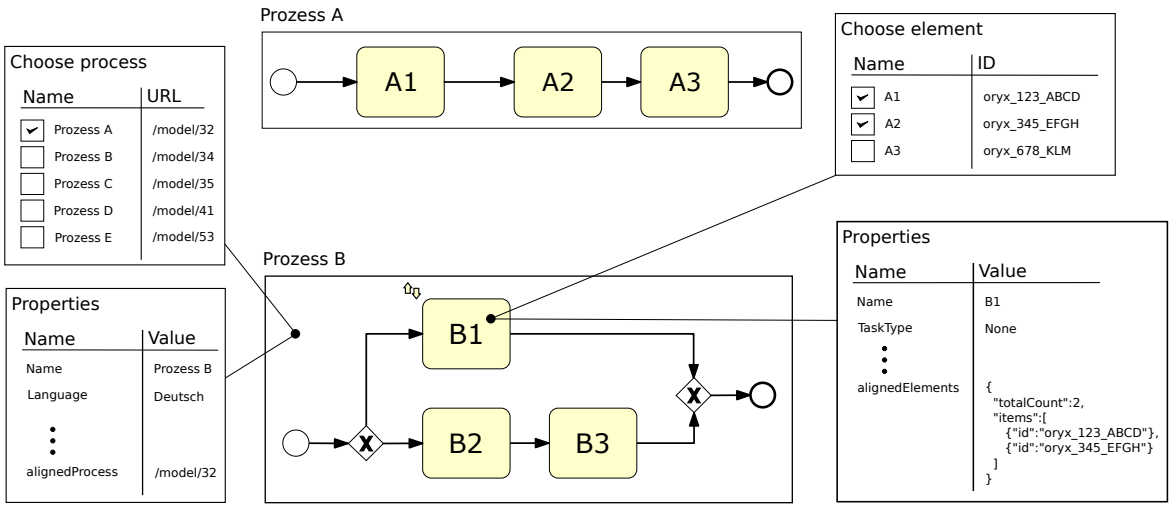

Abbildung 6.7: Modellzuordnung mit Oryx

Für die Zuordnung eines Prozesses ruft das `AlignmentWizard`-Plugin die Daten aller im Repository liegenden Prozesse ab und zeigt sie zur Auswahl in einem Dialogfenster an (Abbildung 6.8). Für die Zuordnung der Prozesselemente ruft es alle Task-Elemente des verknüpften Prozess ab und zeigt sie ebenfalls in einem Dialogfenster zu Auswahl an (Abbildung 6.8). Durch das Nachrichtenkonzept von Oryx teilt es anderen Plugins mit, wenn eine Zuordnung stattgefunden hat. Wurde eine Aktivität erfolgreich mit anderen Aktivitäten verbunden, wird sie markiert.

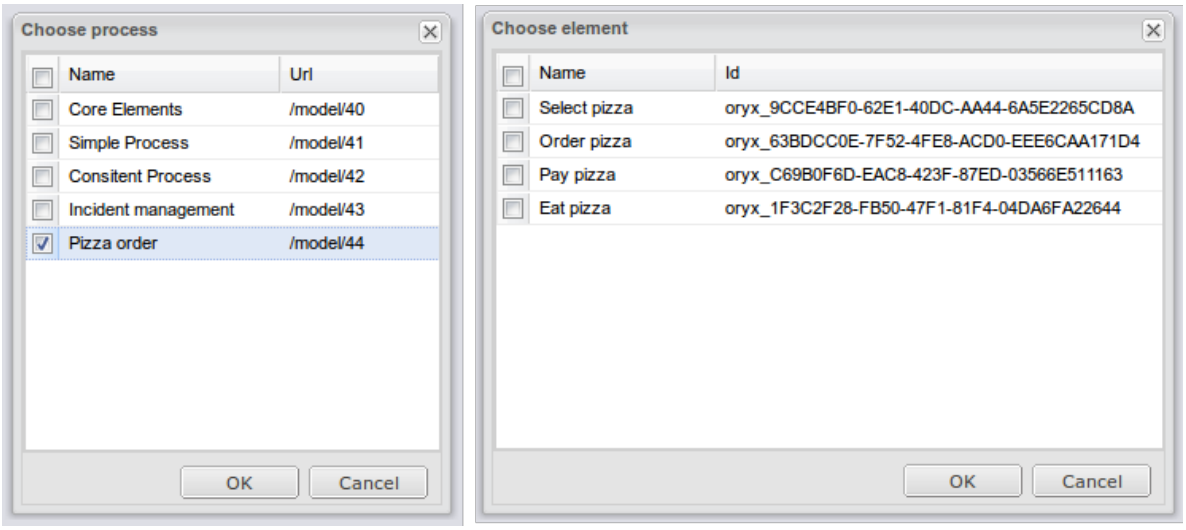

Abbildung 6.8: Dialog zur Auswahl von Prozessen und Prozesselementen

Zur Prüfung der Zuordnung ruft das `AlignmentWizard`-Plugin das serverseitige `SVG-Alignment`-Plugin auf und zeigt die veränderte SVG-Datein des verbundenen Modells an. Das `SVG-Alignment`-Plugin wiederum lädt aus dem Repository die SVG-Datei eines Prozessmodells und markiert je nach Eingabeparameter die nicht verbundenen Aktivitäten (Abbildung 6.9).

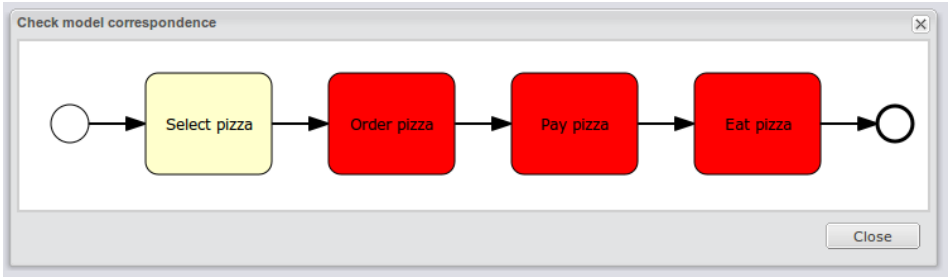

Abbildung 6.9: Prüfung der Modellzuordnung durch das `AlignmentWizard`-Plugin

Bei der Modellierung des Modellangleichs werden zwei Ausgangsszenarien unterschieden:

- Prozess A auf dem Level k ist fertig modelliert und Prozess B auf dem Level k+1 wird anhand von Prozess A erst modelliert. Der umgekehrte Fall ist auch denkbar und fällt in das gleiche Ausgangsszenario.
- Prozess A auf dem Level k und Prozess B auf dem Level k+1 entstehen oder existieren bereits unabhängig voneinander und sollen zuerst in Relation gebracht werden.

In beiden Fällen bietet es sich an, die Beziehungen zwischen den Modellen erst herzustellen, wenn die Modellierung von *beiden* Prozessmodellen abgeschlossen ist oder ein finales Stadium erreicht hat. Das hat den Vorteil, dass ständige Anpassungen, die im Entstehen eines Modells auftreten, das Änderungsprotokoll nicht zweckentfremdet überfüllen.

6.5 Umsetzung der Änderungsoperationen

Jede Änderung an den Aktivitäten eines Modells wird vom Plugin `Change-Queue` durchgeführt. Möchte der Modellierer zum Beispiel eine Aktivität löschen, markiert er die zu löschende Aktivität und benutzt aus der Werkzeugleiste die entsprechende Schaltfläche. Das Plugin entfernt die Aktivität und schreibt diese Operation in das Änderungsprotokoll (Abbildung 6.10) . Das `Change-Queue`-Plugin verändert also für den Modellierer sein Prozessmodell. Durch das Nachrichtenkonzept von Oryx teilt es anderen Plugins mit, wenn eine Änderungsoperation ausgeführt wurde.

Alle Änderungen, die nicht über das Plugin durchgeführt werden, können somit nicht in das Änderungsprotokoll aufgenommen werden. Der Modellierer entscheidet also welche Änderungen er protokollieren möchte und welche nicht. Das Konzept ist derart variabel gestaltet, dass es um beliebige Änderungsoperationen erweitert werden kann.

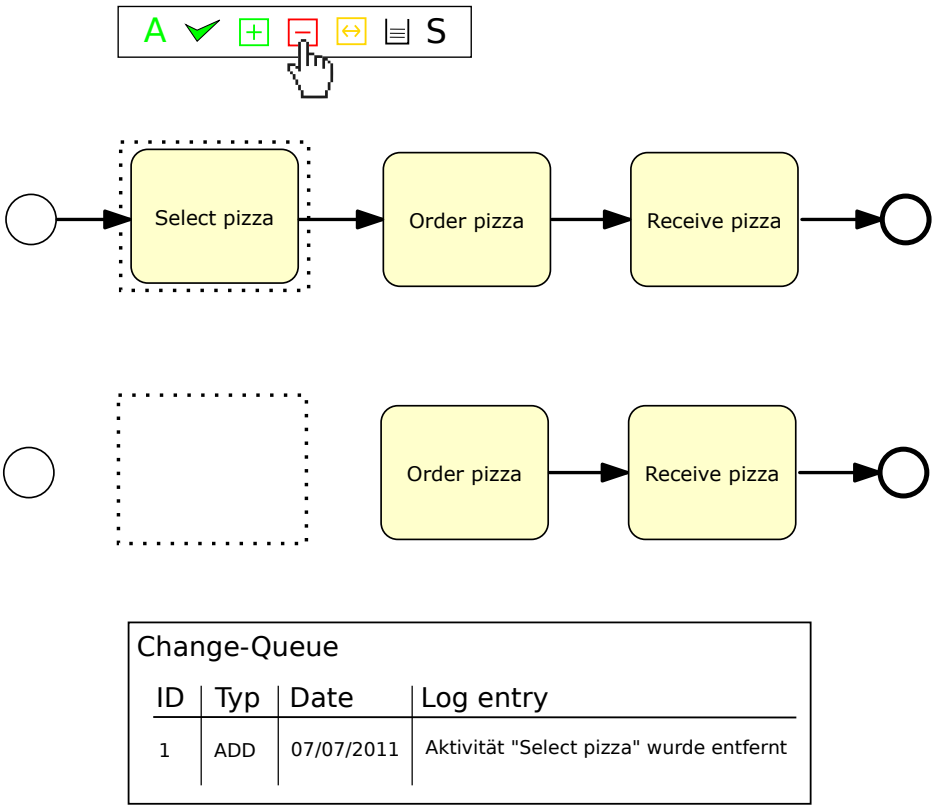

Abbildung 6.10: Der Benutzer wählt einen Aktivität aus und benutzt dann die Werkzeugleiste, um sie zu entfernen. Das Plugin `Change-Queue` schreibt die entsprechende Änderungsoperation in das Protokoll.

6.5.1 Erkennen von Änderungen

Theoretisch ist es möglich komplexe Änderungsoperationen automatisch zu erkennen. Dazu müsste ein Plugin alle Änderungen an einem Modell registrieren und sie bei Bedarf zu Änderungsoperationen zusammenfassen. Die Hintereinanderausführung von einfachen Änderungen ergäbe dann eine komplexe Operation wie man sie zum Beispiel bei `SWAP` gegeben hat.

Die Nachteile dieser Vorgehensweise liegen jedoch auf der Hand: Zum einen würde der Modellierer auch ungewollte Änderungen weitergeben und zum anderen könnte es zu Fehlern in der Erkennung der komplexen Operationen kommen. Das Problem liegt in der Überscheidung der komplexen Operationen. Damit die Erkennung fehlerfrei funktioniert und die Änderungsoperation eindeutig erkannt werden, müsste sich der Modellierer an eine bestimmte Reihenfolge halten. Das aber macht die automatische Erkennung obsolet.

6.5.2 Änderungsprotokoll

Das Eigenschaftsfeld `change-queue` eines Prozessmodells repräsentiert das Änderungsprotokoll und kann jederzeit über das `Change-Queue`-Plugin angezeigt werden (Abbildung 6.11). Änderungsoperationen werden in aufsteigender Nummerierung in dieses Feld abgelegt. Eine mögliches Änderungsprotokoll findet sich in Listing 6.3.

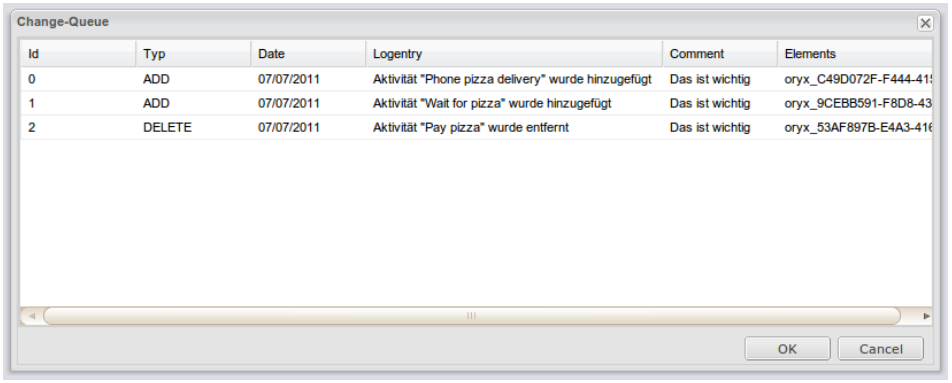

Abbildung 6.11: Dialog zur Anzeige des Änderungsprotokolls.

Listing 6.3 Änderungsprotokoll Beispiel

```
{
  "totalCount":2,
  "items":[
    {
      "id":12,
      "type":"ADD",
      "date":"07/18/11",
      "logentry":"AKTIVITAET "Rechnung vorbereiten" WURDE EINGEFUEGT",
      "comment":"Verbessert die Abrechnung.",
      "elements":"oryx_345_EFGH"
    },
    {
      "id":13,
      "type":"DELETE",
      "date":"07/18/11",
      "logentry":"AKTIVITAET "Papier bestellen" WURDE ENTFERNT",
      "comment":"Wird automatisch von "Rechnung vorbereiten" durchgefuehrt.",
      "elements":"oryx_678_IJKL"
    }
  ]
}
```

Änderungsoperationen werden ausschließlich dann erzeugt, wenn der Modellierer die Schaltflächen des `Change-Queue`-Plugins benutzt um sein Prozessmodell zu verändern. Tabelle 6.1 zeigt die Anatomie eines Eintrags einer Änderungsoperation im Protokoll.

Attribut	Beschreibung
ID	Identifiziert die Änderung.
Typ	Folgende Typen sind nach Abschnitt 5.5.2 möglich: `ADD`, `DELETE`, `MOVE`, `REPLACE`, `SWAP`, `TYPCHANGE`
Datum	Das Datum der Änderung.
Logeintrag	Protokolleintrag nach Abschnitt 5.5.2.
Kommentar	Der Modellierer kann seine Änderung kommentieren.
Elemente	Elemente, auf die die Änderung angewendet wurde.

Tabelle 6.1: Struktur einer Änderungsoperation in Oryx

6.6 Umsetzung des Synchronisationskonzepts

Das `Synchronization`-Plugin lädt das Änderungsprotokoll des verbundenen Prozessmodells und zeigt es in einem Dialogfenster an (Abbildung 6.12). Der Modellierer wählt eine zu übernehmende Operation aus und startet die Synchronisierung der ausgewählten Änderung. Abhängig von Typ der Änderung und den Nachbedingungen aus Abschnitt 5.5.2 verändert das `Synchronization`-Plugin die Eigenschaftsfelder `alignedElements` der einzelnen Aktiviäten und entfernt aus ihnen die entsprechenden Einträge.

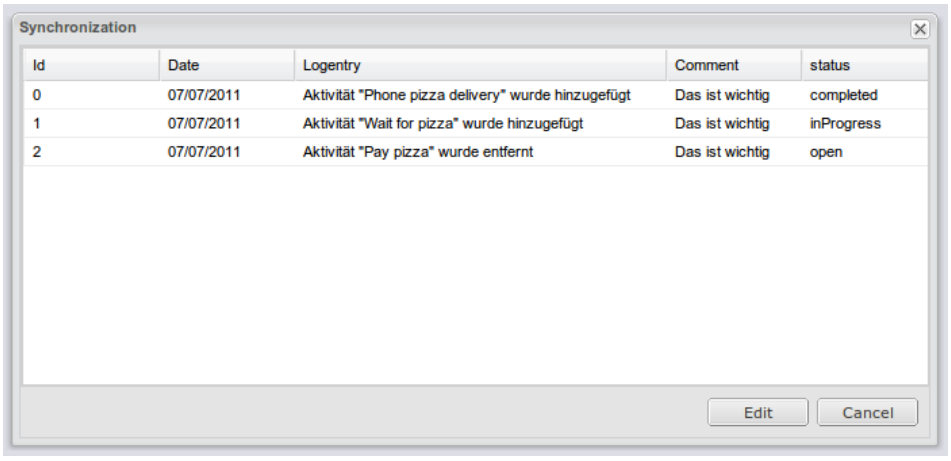

Abbildung 6.12: SynchronisationWizard: Dialog zur Synchronisation

Anschließend wartet es auf Ereignisse vom `AlignmentWizard`-Plugin bis der Modellierer den veränderten Modellelementen des verbundenen Prozessmodells neue Aktivitäten zugewiesen hat. Dazu kann er selbst sein Prozessmodell mit Hilfe vom `Change-Queue`-Plugin verändern und so neue Änderungsoperationen generieren.

6.6.1 Datenstruktur für die Synchronisation

Um festzustellen welchen Änderungsoperation bereits übernommen wurden, besitzt jedes Prozessmodell zudem ein Eigenschaftsfeld `synchronisation`, indem festgehalten wird welche Änderungsoperation des verwandten Prozessmodells gerade in Bearbeitung ist und welche letzte Operation abgeschlossen wurde. Die Datenstruktur erlaubt mit mehrerer Änderungsprotokollen zu synchronisieren. Ein Beispiel für einen Eintrag findet sich in Listing 6.4.

Listing 6.4 Synchronisationsdatenstruktur Beispiel

```
{
  "totalCount":1,
  "items":[
     {
       "processURI":"model/14",
       "inProgress":"17",
       "completed" :"16"
     }
   ]
}
```

7 Zusammenfassung und Ausblick

Ziel dieser Arbeit war der Entwurf eines Konzepts zur adaptiven Modellierung von Geschäftsprozessen mittels mehrerer Abstraktionsschichten. Das Resultat ist ein Mechanismus zur Weiterreichung von Änderungen auf Prozessmodellen mit Hilfe von Änderungsoperationen.

Zuerst war es wichtig festzustellen welche Arten von Änderungen auf Prozessmodellen vorkommen. Das führte zu einer Kategorisierung der Änderungen mit charakteristischen Merkmalen wie zum Beispiel der Menge der Modellelemente auf der die Änderung angewendet werden kann.

Um die Modelle in Relation zu setzen, wurde eine Modellzuordnung eingeführt. Änderungen operieren auf dieser Struktur und bilden somit die Änderungen ab. Damit konnte eine Definition für Konsistenz aufgestellt und automatisch geprüft werden.

Als nächstes mussten diese Änderungen weitergereicht werden. Dazu wurde ein Änderungsprotokoll eingeführt, das alle Änderungsoperationen aufzeichnet und verwandten Prozessmodellen zugänglich ist. Idealerweise sollten die Änderungen automatisch in andere Prozessmodelle übernommen werden, was sich jedoch als sehr schwierig herausstellte.

Das vorgeschlagene Konzept hat deswegen den Nachteil, dass Änderungen nur so schnell propagiert werden können, wie schnell die Modellierer auf diese Änderungen reagieren und sie bearbeiten. Bei zehn Abstraktionssichten und einer Bearbeitungsdauer von einem Tag pro Änderung, bräuchte eine Änderung auf der obersten Ebene zehn Tage bis sie am untersten Modell angekommen ist.

Momentan ist es möglich Prozessmodelle über mehrere Abstraktionsschichten hinweg zu verbinden. Das birgt die Gefahr, dass Änderungen nicht sachgerecht übernommen werden, weil der Unterschied der Abstraktionsgrade zu groß ist. Andrerseits ist es praktisch nicht immer möglich zu jeder Abstraktionsschicht ein entsprechendes Modell zu entwerfen.

Der große Vorteil des Ansatzes liegt in der Freiheit des Modellierers mit beliebigen Änderungen nach eigenem Ermessen umzugehen. So kann ein Prozess mit Hilfe der Änderungsoperationen in einen komplett anderen Prozessmodell umgewandelt werden und alle verwandten Modelle können diese Veränderung nachvollziehen und entsprechend synchronisieren. Erwartungen an einen Automatismus der Synchronisierung aller Prozessmodelle in den verschiedenen Abstraktionsschichten kann das Konzept nicht erfüllen, bietet jedoch eine Lösung, die allen Anforderungen der adaptiven Prozessmodellierung gerecht wird.

Ausblick

Die Arbeit mit Änderungsoperationen ist ein sehr generisches Konzept, das beliebig erweitert werden kann. Im Folgenden werden einige Ansätze zur Erweiterung vorgeschlagen und konkrete Implementierungsvorschläge gegeben.

Profile

In vielen Fällen ist ein Prozessmodell von mehreren Modellen abhängig. Diese Multiabhängigkeit lässt sich konzeptuell am besten durch Profile ausdrücken. Somit könnte ein Modell, je nach Profil, mit mehreren Prozessmodellen verbunden sein. Die entsprechenden Datenstrukturen müssten nur noch mit einer `profile-id` parametrisiert werden.

Vorschlag einer Änderungsregion

Man könnte einen Mechanismus implementieren, der dem Modellierer eine mögliche Änderungsregion einer Übernahme vorschlägt. Ein paar Ansätze dafür wurden bereits im Kapitel 4 - Verwandte Arbeiten genannt. Im folgenden ein Vorschlag auf der Basis von Prozess-Struktur-Bäumen (process structure tree) [VVK09].

Ein Prozessstrukturbaum unterteilt einen Prozessgraphen in sogenannte SESE-Fragmente. Das sind „Single-entry-single-exit“ Fragmente, also Untergraphen eines Prozessgraphen, die nur eine eingehende Kante und eine ausgehende Kante besitzen. Abbildung 7.1 zeigt einen Beispielprozess und seine SESE-Fragmente und Abbildung 7.1 den dazugehörigen Prozess-Struktur-Baum.

Die Idee ist, die veränderten Aktivitäten zu nehmen und das SESE-Fragment der dazu verbundenen Aktivitäten zu bestimmen und entsprechend zu markieren. Listing 7.1 beschreibt einen möglichen Algorithmus für den Vorschlag einer Änderungsregion.

Algorithmus 7.1 Bestimmung der Änderungsregion anhand von SESE-Fragmenten

```
procedure ChangeRegion(ChangeOperation c)
    sese_fragment_source ← SESE-FRAGMENT(c[changed_elements])
                          // Bestimme das SESE-Fragment der veränderten Aktivitäten
    linked_elements ← ∅
    for all activities a ∈ sese_fragment_source do
        linked_elements ← linked_elements ∪ {b|(a, b) ∈ R}
                                          // Bestimme die verbundenen Aktivitäten
    end for
    sese_fragment_target ← SESE-FRAGMENT(linked_elements)
                          // Bestimme das SESE-Fragment der verbundenen Aktivitäten
    return sese_fragment_target
end procedure
```

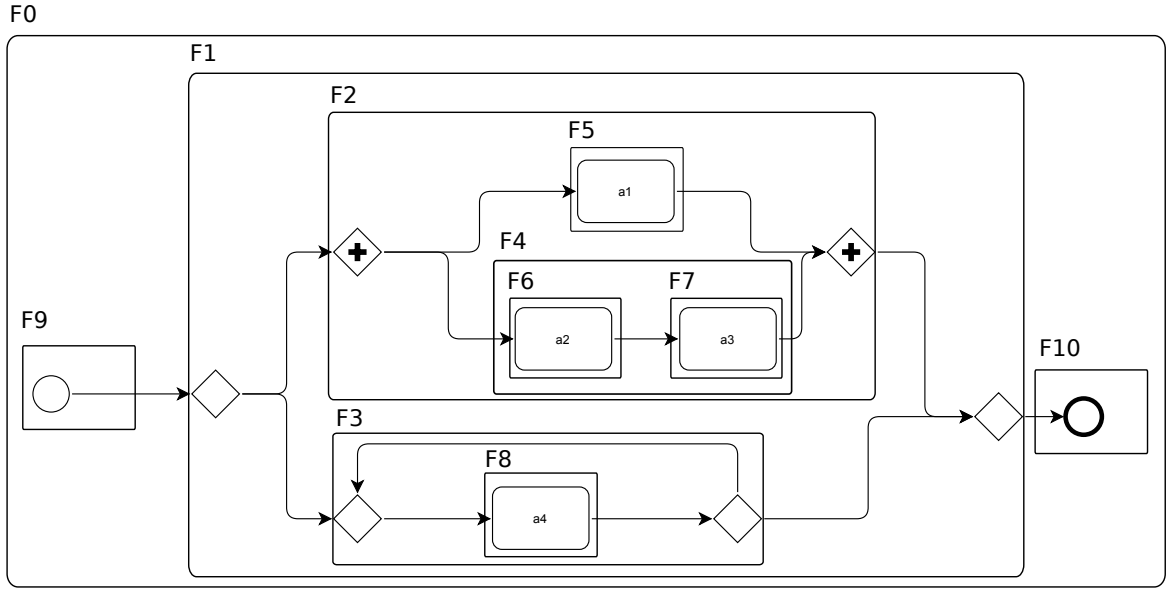

Abbildung 7.1: SESE-Fragmente eines Beispielprozesses

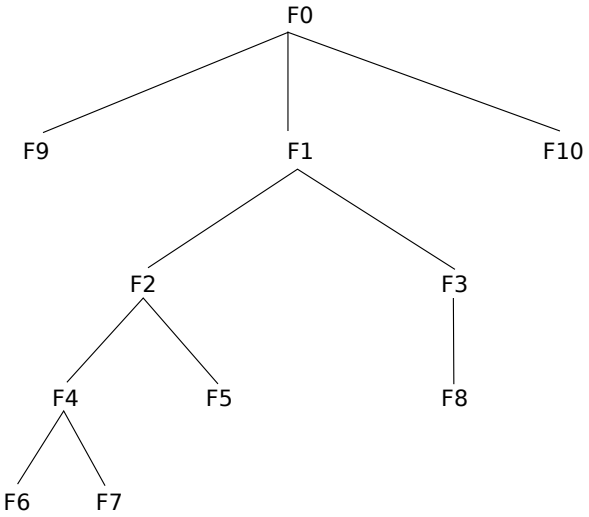

Abbildung 7.2: Prozess-Struktur-Baum des Prozesses aus Abbildung 7.1

Verfeinerungs- und Übernahmebedingungen

Wie schon in Abschnitt 5.7 vom Kapitel 5 - Modellierungskonzept angedeutet, besteht die Möglichkeit Bedingungen an die Verfeinerung eines Prozessmodells zu stellen. So ließen sich in der Nachbedingung einer Änderungsoperation beliebige Einschränkungen formulieren, die eingehalten werden müssen, damit die Änderungen als übernommen akzeptiert wird.

Möchte ein Modellierer beim Einfügen einer neuer Aktivität X, erzwingen dass jede Verfeinerung auch eine neue Aktivität hinzuzufügen hat, könnte er in der Änderungsoperation vermerken, dass sie nur durch eine weitere Einfüge-Operation übernommen werden kann.

Dazu erweitert man die Datenstruktur einer Änderungsoperation um ein Eigenschaftsfeld `adaption_operation` und prüft, ob die Änderung mit dieser Operation übernommen wurde.

Änderungen der Semantik

Das Konzept findet nur Anwendung auf der Menge der Aktivitäten eines Modells. Es gibt keine Änderungsoperationen für Anpassungen am Ablauf oder an den Bedingungen von Kontrollflusselementen eines Modells. Um diese Lücke zu schließen, könnte man weitere Operationen einführen. Eine andere Möglichkeit wäre die Einführung eines Nachrichtensystems mit dem Modellierer über den `Zweck` ihrer Änderung kommunizieren können.

Literaturverzeichnis

[ABL08] W. M. P. van der Aalst, K. Bisgaard Lassen. Translating unstructured workflow processes to readable BPEL: Theory and implementation. *Inf. Softw. Technol.*, 50:131–159, 2008. (Zitiert auf Seite 24)

[All95] T. Allweyer. Modellierung und Gestaltung adaptiver Geschäftsprozesse. IWi-Heft 115, Veröffentlichungen des Instituts für Wirtschaftsinformatik im Deutschen Forschungszentrum für Künstliche Intelligenz, 1995. URL http://www.uni-saarland.de/fileadmin/user_upload/Fachrichtungen/fr13_BWL/professuren/PDF/heft115.pdf. (Zitiert auf Seite 13)

[All05] T. Allweyer. *Geschäftsprozessmanagement. Strategie, Entwurf, Implementierung,Controlling*. Verlag Moritz Diesterweg, 2005. (Zitiert auf Seite 31)

[All07] T. Allweyer. Erzeugung detaillierter und ausführbarer Geschäftsprozessmodelle durch Modell-zu-Modell-Transformationen. In *EPK*, pp. 23–38. 2007. (Zitiert auf Seite 24)

[All10] T. Allweyer. *BPMN 2.0*. Books on Demand, 2010. (Zitiert auf den Seiten 15 und 35)

[BBR11] S. Buchwald, T. Bauer, M. Reichert. Bridging the Gap Between Business Process Models and Service Composition Specifications. In *Int'l Handbook on Service Life Cycle Tools and Technologies: Methods, Trends and Advances*. 2011. (Zitiert auf den Seiten 7, 26 und 27)

[BEK+06] S. Brockmans, M. Ehrig, A. Koschmider, A. Oberweis, R. Studer. Semantic Alignment Of Business Processes. In *IN: PROCEEDINGS OF THE EIGHTH INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS (ICEIS 2006*, pp. 191–196. INSTICC Press, 2006. (Zitiert auf den Seiten 7, 15, 27 und 28)

[BFR+] P. Bauler, E. Frogneux, B. Renwart, C. Thomase, I. S. Collaboration, C. Recherche. Usage of Model Driven Engineering in the context of Business Process Management. (Zitiert auf Seite 24)

[CT09] M. Chinosi, A. Trombetta. *2009 BPM and Workflow Handbook*, chapter A Design Methodology for BPMN, pp. 211–224. Future Strategies Inc., Book Division, 2009. (Zitiert auf Seite 30)

[DBB+02] E. Deborin, J. Basrai, T. Benedetti, R. Halchin, T. Mahfouz, N. Perera, B. Shamshabad, R. Spory, R. Turakhia. *Continuous Business Process Management with HOLOSOFX BPM Suite and IBM MQSeries Workflow*. IBM Corp., Riverton, NJ, USA, first edition, 2002. (Zitiert auf Seite 24)

[DDGBK09] R. Dijkman, M. Dumasy, L. García-Bañuelos, R. Käärik. Aligning Business Process Models. In *Proceedings of the 2009 IEEE International Enterprise Distributed Object Computing Conference (edoc 2009)*, EDOC '09, pp. 45–53. IEEE Computer Society, Washington, DC, USA, 2009. (Zitiert auf den Seiten 15 und 26)

[DDM08] B. van Dongen, R. Dijkman, J. Mendling. Measuring Similarity between Business Process Models. In Z. Bellahsène, M. Léonard, editors, *Advanced Information Systems Engineering*, volume 5074 of *Lecture Notes in Computer Science*, pp. 450–464. Springer Berlin / Heidelberg, 2008. (Zitiert auf Seite 27)

[Dec06] G. Decker. Bridging the Gap between Business Processes and existing IT Functionality. In *Proceedings of the 1st International Workshop on Design of Service-Oriented Applications (WDSOA)*, ICSOC, pp. 17–24. 2006. (Zitiert auf Seite 27)

[Dij] R. Dijkman. Feedback on Differences between Business Processes. (Zitiert auf Seite 27)

[Dij08] R. Dijkman. Diagnosing Differences between Business Process Models. In *Proceedings of the 6th International Conference on Business Process Management*, BPM '08, pp. 261–277. Springer-Verlag, Berlin, Heidelberg, 2008. (Zitiert auf Seite 27)

[Gad07] A. Gadatsch. *Grundkurs Geschäftsprozess-Management*. Vieweg, 5., erw. u. überarb. edition, 2007. (Zitiert auf den Seiten 11 und 15)

[Gla01] R. van Glabbeek. The Linear Time-Branching Time Spectrum I - The Semantics of Concrete, Sequential Processes. In *Handbook of Process Algebra, chapter 1*, pp. 3–99. Elsevier, 2001. (Zitiert auf Seite 27)

[GLKE10] C. Gerth, M. Luckey, J. M. Kuster, G. Engels. Detection of Semantically Equivalent Fragments for Business Process Model Change Management. In *Services Computing (SCC), 2010 IEEE International Conference on*, pp. 57 –64. 2010. doi: 10.1109/SCC.2010.38. (Zitiert auf Seite 27)

[GZ09] J. Gao, L. Zhang. On Measuring Semantic Similarity of Business Process Models. In *Interoperability for Enterprise Software and Applications China, 2009. IESA '09. International Conference on*, pp. 289 –293. 2009. doi:10.1109/I-ESA.2009.50. (Zitiert auf Seite 27)

[HZJ04] M. Henkel, J. Zdravkovic, P. Johannesson. Service-based processes: design for business and technology. In *Proceedings of the 2nd international conference on Service oriented computing*, ICSOC '04, pp. 21–29. ACM, New York, NY, USA, 2004. (Zitiert auf Seite 26)

[KGFE08] J. M. Küster, C. Gerth, A. Förster, G. Engels. Detecting and Resolving Process Model Differences in the Absence of a Change Log. In *Proceedings of the 6th International Conference on Business Process Management*, BPM '08, pp. 244–260. Springer-Verlag, Berlin, Heidelberg, 2008. (Zitiert auf Seite 26)

[KHK+03] J. Koehler, R. Hauser, S. Kapoor, F. Wu, S. Kumaran. A model-driven transformation method. In *Enterprise Distributed Object Computing Conference, 2003. Proceedings. Seventh IEEE International*, pp. 186 – 197. 2003. doi: 10.1109/EDOC.2003.1233848. (Zitiert auf Seite 24)

[KHK+08] J. Koehler, R. Hauser, J. Küster, K. Ryndina, J. Vanhatalo, M. Wahler. The Role of Visual Modeling and Model Transformations in Business-driven Development. *Electron. Notes Theor. Comput. Sci.*, 211:5–15, 2008. (Zitiert auf Seite 23)

[LRF10] J. Lemcke, T. Rahmani, A. Friesen. Semantic business process engineering. In *Proceedings of the 6th international conference on Semantic technologies for software engineering*, ReasoningWeb'10, pp. 161–181. Springer-Verlag, Berlin, Heidelberg, 2010. (Zitiert auf Seite 32)

[LS10] M. Lind, U. Seigerroth. Multi-Layered Process Modeling for Business and IT Alignment. *Hawaii International Conference on System Sciences*, 0:1–10, 2010. doi:http://doi.ieeecomputersociety.org/10.1109/HICSS.2010.280. (Zitiert auf den Seiten 7, 21 und 22)

[Lud02] J. Ludewig. Modelle im Software Engineering - eine Einführung und Kritik. In *Modellierung in der Praxis - Modellierung für die Praxis*, pp. 7–22. GI, 2002. (Zitiert auf Seite 13)

[MW06] A. K. A. D. Medeiros, A. J. M. M. Weijters. Process Equivalence: Comparing Two Process Models Based on Observed Behavior. In *International Conference on Business Process Management (BPM 2006), volume 4102 of Lecture Notes in Computer Science*, pp. 129–144. Springer, 2006. (Zitiert auf Seite 27)

[OMG10] OMG. BPMN 2.0 by Example. pp. 8–10, 2010. URL http://www.omg.org/spec/BPMN/2.0/examples/PDF. (Zitiert auf Seite 23)

[OMG11] OMG. BPMN 2.0 Spezifikation, 2011. URL http://www.omg.org/spec/BPMN/2.0/. (Zitiert auf den Seiten 7, 11, 22 und 23)

[Org07] Organization for the Advancement of Structured Information Standards (OASIS). *Web Services Business Process Execution Language (WS-BPEL) Version 2.0*, 2007. URL http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html. (Zitiert auf Seite 24)

[RBRB06] S. Rinderle, R. Bobrik, M. Reichert, T. Bauer. Business Process Visualization - Use Cases, Challenges, Solutions. In *ICEIS (3)*, pp. 204–211. 2006. (Zitiert auf Seite 14)

[RGL+09] Y. Ren, G. Gröner, J. Lemcke, T. Rahmani, A. Friesen, Y. Zhao, J. Z. Pan, S. Staab. Validating Process Refinement with Ontologies. In *Description Logics*. 2009. (Zitiert auf den Seiten 14 und 32)

[RMRW08] S. Rinderle-Ma, M. Reichert, B. Weber. On the Formal Semantics of Change Patterns in Process-Aware Information Systems. In Q. Li, S. Spaccapietra, E. Yu, A. Olivé, editors, *Conceptual Modeling - ER 2008*, volume 5231 of *Lecture Notes in Computer Science*, pp. 279–293. Springer Berlin / Heidelberg, 2008. (Zitiert auf Seite 29)

[RQZ07] C. Rupp, S. Queins, B. Zengler. *UML2 glasklar: Praxiswissen für die UML-Modellierung*. Hanser Verlag, 3. auflage edition, 2007. (Zitiert auf Seite 24)

[SEL+10] H. Schwarz, J. Ebert, J. Lemcke, T. Rahmani, S. Zivkovic. Using Expressive Traceability Relationships for Ensuring Consistent Process Model Refinement. In *Proceedings of the 15th IEEE International Conference on Engineering of Complex Computer Systems*. 2010. (Zitiert auf Seite 32)

[Sha10] R. Shapiro. Update on BPMN Release 2.0, 2010. URL http://www.wfmc.org/standards/Update2.0_final.ppt. (Zitiert auf den Seiten 7, 21, 22 und 48)

[Sil] B. Silver. BPMN Level 3 Method and Style - First Thoughts. URL http://www.brsilver.com/2010/05/27/bpmn-level-3-method-and-style-first-thoughts/. (Zitiert auf Seite 31)

[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. URL http://www.bpmnstyle.com/. (Zitiert auf den Seiten 21, 31 und 48)

[SO00] W. Sadiq, M. Orlowska. On Business Process Model Transformations. In A. Laender, S. Liddle, V. Storey, editors, *Conceptual Modeling — ER 2000*, volume 1920 of *Lecture Notes in Computer Science*, pp. 47–104. Springer Berlin / Heidelberg, 2000. (Zitiert auf Seite 24)

[Sta73] H. Stachowiak. *Allgemeine Modelltheorie*. Springer-Verlag, Wien[u.a.], 1973. (Zitiert auf Seite 13)

[Tsc07] W. Tscheschner. Oryx Dokumentation (Bachelor's thesis), 2007. URL http://oryx-editor.googlecode.com/files/BachelorthesisWilliTscheschner.pdf. (Zitiert auf den Seiten 7, 43 und 44)

[VVK09] J. Vanhatalo, H. Völzer, J. Koehler. The refined process structure tree. *Data & Knowledge Engineering*, 68(9):793 – 818, 2009. Sixth International Conference on Business Process Management (BPM 2008) - Five selected and extended papers. (Zitiert auf den Seiten 26 und 58)

[WBMW] M. Weidlich, A. Barros, J. Mendling, M. Weske. Vertical Alignment of Process Models - How can we get there? (Zitiert auf den Seiten 26 und 31)

[WKK+11] M. Weidmann, F. Kötter, M. Kintz, D. Schleicher, R. Mietzner, F. Leymann. Adaptive Business Process Modeling in the Internet of Services (ABIS). In P. of the Sixth International Conference on Internet, W. Applications, S. I. 2011, editors, *Adaptive Business Process Modeling in the Internet of Services (ABIS)*, pp. 1–8. Xpert Publishing Services, 2011. (Zitiert auf Seite 29)

[WRRM08] B. Weber, M. Reichert, S. Rinderle-Ma. Change patterns and change support features - Enhancing flexibility in process-aware information systems. *Data Knowl. Eng.*, 66:438–466, 2008. (Zitiert auf den Seiten 7, 29, 30 und 36)

[WSR09] B. Weber, S. Sadiq, M. Reichert. Beyond Rigidity - Dynamic Process Lifecycle Support. *Computer Science - Research and Development*, 23:47–65, 2009. (Zitiert auf Seite 28)

[WWM09] M. Weidlich, M. Weske, J. Mendling. Change Propagation in Process Models Using Behavioural Profiles. In *Services Computing, 2009. SCC '09. IEEE International Conference on*, pp. 33 –40. 2009. (Zitiert auf Seite 31)

[WWW] Oryx: Research. URL http://bpt.hpi.uni-potsdam.de/Oryx/Research. (Zitiert auf Seite 43)

[WWW05] BPMN Core Elements, 2005. URL http://www.omg.org/bpmn/Samples/Elements/Core_BPMN_Elements.htm. (Zitiert auf den Seiten 7, 15, 16 und 48)

[ZP09] J. Zhu, H. K. Pung. Process Matching: A Structural Approach for Business Process Search. In *Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009. COMPUTATIONWORLD '09. Computation World:*, pp. 227 –232. 2009. (Zitiert auf Seite 27)

Alle URLs wurden zuletzt am 03.08.2011 geprüft.

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Modood Ahmad Alvi)